

HCSS User's Reference Manual

Herschel Data Processing

**Version 3.0, Document Number: HERSCHEL-HSC-DOC-0935
25 August 2010**



HCSS User's Reference Manual: Herschel Data Processing

Table of Contents

I. Categorized view of Commands	xii
II. How to use this manual	xv
II.1. Related documentation	xv
III. Dictionary	xvi
1. Numeric	1
1.1. Introduction	1
1.2. Quick start	1
1.3. Import statements	1
1.4. Constants	2
1.5. Arrays	2
1.6. Toolboxes	14
2. DP Commands	21
2.1. Introduction	21
2.2. ABS	23
2.3. AbstractImageManipulationTask	24
2.4. AbstractMappingTask	25
2.5. AbstractModel	26
2.6. AddSpectrum	27
2.7. ALL	30
2.8. AllPresent	31
2.9. AmoebaFitter	33
2.10. AnnotationToolbox	35
2.11. AnnularSkyAperturePhotometryExplorer	37
2.12. AnnularSkyAperturePhotometryPanel	40
2.13. AnnularSkyAperturePhotometryProduct	42
2.14. AnnularSkyAperturePhotometryTask	45
2.15. ANY	47
2.16. AnyPresent	49
2.17. AperturePhotometryExplorer	51
2.18. AperturePhotometryPanel	56
2.19. AperturePhotometryProduct	64
2.20. AperturePhotometryTask	72
2.21. ARCCOS	74
2.22. ARCSIN	75
2.23. ARCTAN	76
2.24. ArctanModel	77
2.25. ArithmeticSpectrumTask	78
2.26. ArrayAssistant	81
2.27. asciiTableReader	82
2.28. asciiTableWriter	86
2.29. AttribQuery	88
2.30. AutoCorrelationTask	89
2.31. AutomaticContourTask	90
2.32. AutomaticContourTaskSignatureComponent	92
2.33. AverageSpectrum	94
2.34. bg	98
2.35. Bilinear	100
2.36. BinCentres	107
2.37. BinomialModel	109
2.38. Bool1d	110
2.39. Bool2d	111
2.40. Bool3d	112
2.41. Bool4d	113
2.42. Bool5d	114
2.43. BoxCarFilter	115

2.44. BoxCarSmoothingTask	117
2.45. Byte1d	118
2.46. Byte2d	119
2.47. Byte3d	120
2.48. Byte4d	121
2.49. Byte5d	122
2.50. CachedPool	123
2.51. CEIL	125
2.52. ChannelMapPlotting	126
2.53. CholeskyDecomposition	130
2.54. CircleHistogramExplorer	132
2.55. CircleHistogramPanel	135
2.56. CircleHistogramProduct	137
2.57. CircleHistogramTask	140
2.58. CircularIntegrationTask	142
2.59. clamp	143
2.60. ClampTaskSignatureComponent	145
2.61. clear	148
2.62. Complex1d	150
2.63. Complex2d	151
2.64. Complex3d	152
2.65. Complex4d	153
2.66. Complex5d	154
2.67. CONCATENATE	155
2.68. Condense	156
2.69. ConjugateGradientFitter	160
2.70. ConnectorBox	161
2.71. Contour	162
2.72. ContourLevel	164
2.73. ContourPlotting	166
2.74. ContourTask	170
2.75. ConvertImageUnitTask	171
2.76. ConvertImageUnitTaskSignatureComponent	172
2.77. Convolution	175
2.78. Correlate	177
2.79. CorrelateMatrix	179
2.80. COS	180
2.81. COSH	182
2.82. CreateRgbImageTask	183
2.83. CreateRgbImageTaskSignatureComponent	185
2.84. crop	187
2.85. CrossCorrelationTask	189
2.86. Cube3Dviewer	190
2.87. CubeCompareDisplay	196
2.88. CubeSpectrumAnalysis	199
2.89. CubeSpectrumAnalysisToolbox	201
2.90. CubicSplineInterpolator	219
2.91. CubicSplinesModel	221
2.92. cutLevels	222
2.93. CutLevelsTaskSignatureComponent	224
2.94. DataFlow	226
2.95. DataFlowManager	227
2.96. DbFactory	228
2.97. DbPool	229
2.98. DETERMINANT	232
2.99. DFT2dTask	233
2.100. Display	234
2.101. displayLog	277

2.102. DivideSpectrum	279
2.103. Double1d	283
2.104. Double2d	284
2.105. Double3d	285
2.106. Double4d	286
2.107. Double5d	287
2.108. EigenvalueDecomposition	288
2.109. EllipseHistogramExplorer	289
2.110. EllipseHistogramPanel	292
2.111. EllipseHistogramProduct	294
2.112. EllipseHistogramTask	297
2.113. Erfc	299
2.114. Erf	300
2.115. EXP10	301
2.116. EXP	302
2.117. ExpModel	303
2.118. ExpN	304
2.119. exportPalToUfDir	305
2.120. ExportSpectrumToASCII	307
2.121. ExtractFreqRanges	309
2.122. ExtractMultiplePixelSpectrumPanel	312
2.123. ExtractRegionPixelSpectrumTask	314
2.124. ExtractSinglePixelSpectrumPanel	316
2.125. ExtractSinglePixelSpectrumTask	318
2.126. FFT2dTask	320
2.127. FFT	321
2.128. FitFringeData	323
2.129. FitFringe	325
2.130. FitsArchive	328
2.131. fitsReader	342
2.132. FitterFunction	344
2.133. Fitter	346
2.134. FitterTask	347
2.135. FixedMask	352
2.136. FixedSkyAperturePhotometryExplorer	355
2.137. FixedSkyAperturePhotometryPanel	358
2.138. FixedSkyAperturePhotometryProduct	360
2.139. FixedSkyAperturePhotometryTask	363
2.140. FIX	365
2.141. Flag	367
2.142. FlagPixels	374
2.143. FlagSaturatedPixelsCubeTask	377
2.144. FlagSaturatedPixelsTask	378
2.145. Float1d	379
2.146. Float2d	380
2.147. Float3d	381
2.148. Float4d	382
2.149. Float5d	383
2.150. FLOOR	384
2.151. FoldSpectrum	385
2.152. FullQuery	387
2.153. GAMMALN	388
2.154. GammaP	389
2.155. GammaQ	390
2.156. Gauss2DModel	391
2.157. Gauss2DRotModel	392
2.158. GaussFitTask	393
2.159. GaussianFilter	398

2.160. GaussianSmoothingTask	400
2.161. GaussModel	401
2.162. getObservation	402
2.163. HAMMING	404
2.164. HANNING	407
2.165. HarmonicModel	410
2.166. HduHeaders	411
2.167. help	413
2.168. Histogram	415
2.169. Histogram	417
2.170. HistogramPanel	422
2.171. HistogramTask	429
2.172. HttpClientFactory	430
2.173. IFFT	431
2.174. ImageAbsTask	433
2.175. ImageAddTask	434
2.176. ImageAddTaskSignatureComponent	435
2.177. ImageAnalysis	437
2.178. ImageAnalysisToolbox	439
2.179. ImageArithmeticsTask	448
2.180. ImageArithmeticsTaskSignatureComponent	449
2.181. ImageAxis	452
2.182. ImageCeilTask	462
2.183. ImageContour	463
2.184. ImageConvolutionTask	466
2.185. ImageConvolutionTaskSignatureComponent	467
2.186. ImageDivideTask	469
2.187. ImageDivideTaskSignatureComponent	470
2.188. ImageExp10Task	472
2.189. ImageExpNTask	473
2.190. ImageExpTask	474
2.191. ImageFloorTask	475
2.192. ImageHistogramExplorer	476
2.193. ImageHistogramProduct	480
2.194. ImageHistogramTask	483
2.195. ImageHistogramTaskSignatureComponent	484
2.196. ImageLog10Task	487
2.197. ImageLogNTask	488
2.198. ImageLogTask	489
2.199. ImageModuloTask	490
2.200. ImageModuloTaskSignatureComponent	491
2.201. ImageMultiplyTask	493
2.202. ImageMultiplyTaskSignatureComponent	494
2.203. ImagePowerTask	496
2.204. ImageRoundTask	497
2.205. ImageSaverTask	498
2.206. ImageSqrtTask	499
2.207. ImageSquareTask	500
2.208. ImageSubtractTask	501
2.209. ImageSubtractTaskSignatureComponent	502
2.210. importCube	504
2.211. importImage	505
2.212. importRgbImage	506
2.213. importSpectralCube	507
2.214. importUfDirToPal	508
2.215. info	510
2.216. InputImageNSignatureComponent	512
2.217. InputImageSignatureComponent	515

2.218. Int1d	518
2.219. Int2d	519
2.220. Int3d	520
2.221. Int4d	521
2.222. Int5d	522
2.223. IntegratedMapDisplay	523
2.224. IntegrateMapFromCubeTask	525
2.225. Integrator	527
2.226. IntensityCalculator	529
2.227. IntensityMapGui	534
2.228. interruptable	535
2.229. InverseDFT2dTask	537
2.230. InverseFFT2dTask	538
2.231. INVERSE	539
2.232. IS_FINITE	540
2.233. IS_INFINITE	541
2.234. IS_NAN	542
2.235. JavaQuery	543
2.236. KURTOSIS	544
2.237. LayerStruct	546
2.238. LevenbergMarquardtFitter	549
2.239. LinearInterpolator	550
2.240. LineIntensityMapTask	552
2.241. ListContext	555
2.242. localStoreCopier	557
2.243. localStoreWriter	559
2.244. LOG10	560
2.245. LOG	561
2.246. LogN	562
2.247. Long1d	564
2.248. Long2d	565
2.249. Long3d	566
2.250. Long4d	567
2.251. Long5d	568
2.252. LorentzModel	569
2.253. LUdecomposition	570
2.254. ManualContourPanel	572
2.255. ManualContourTask	576
2.256. ManyToOneSpectrumTask	577
2.257. MapContext	579
2.258. MatrixMultiply	581
2.259. MatrixSolve	583
2.260. MAX	584
2.261. MEAN	586
2.262. MeanSmoothingTask	587
2.263. MedianAbsoluteDeviation	588
2.264. MEDIAN	589
2.265. MedianSmoothingTask	591
2.266. MetaQuery	592
2.267. MIN	593
2.268. ModelFitTask	595
2.269. MosaicTask	599
2.270. MultiplySpectrum	600
2.271. NAN_FILTER	604
2.272. NearestNeighborInterpolator	605
2.273. NearestNeighbourProjectionTask	607
2.274. Normalize	611
2.275. NotPresent	614

2.276. NumberedDataset	616
2.277. ObservationContext	617
2.278. OpDayGenerator	619
2.279. openFile	620
2.280. openTask	621
2.281. openVariable	622
2.282. OverPlotter	623
2.283. PackedMask	627
2.284. pacs2HipeDouble3d	629
2.285. PadeModel	630
2.286. pause	631
2.287. pointHistoryDisplay	632
2.288. Polygon	634
2.289. PolygonHistogramExplorer	636
2.290. PolygonHistogramPanel	639
2.291. PolygonHistogramProduct	641
2.292. PolygonHistogramTask	644
2.293. PolygonIntersect	646
2.294. Polynomial	647
2.295. PolynomialModel	649
2.296. PolySurfaceModel	650
2.297. poolDataReader	651
2.298. poolDataWriter	654
2.299. PositionList	657
2.300. PowerLawModel	659
2.301. PowerModel	660
2.302. PowerSpectrum	661
2.303. PowerSpectrum	663
2.304. Pow	664
2.305. PreviousInterpolator	665
2.306. PrfGaussian	667
2.307. ProcessDistributor	668
2.308. PRODUCT	670
2.309. ProductRef	672
2.310. ProductStorage	673
2.311. ProfileExplorer	680
2.312. Profile	684
2.313. ProfilePanel	687
2.314. ProfilePlotting	693
2.315. ProfileTask	697
2.316. QRDecomposition	699
2.317. Query	701
2.318. RadialVelocityTask	702
2.319. RandomGauss	706
2.320. RandomPoisson	707
2.321. RandomUniform	708
2.322. RangeExtractionGui	710
2.323. RangeExtractionTask	711
2.324. REBIN	713
2.325. RectangleHistogramExplorer	716
2.326. RectangleHistogramPanel	719
2.327. RectangleHistogramProduct	721
2.328. RectangleHistogramTask	724
2.329. RectangularSkyAperturePhotometryExplorer	726
2.330. RectangularSkyAperturePhotometryPanel	729
2.331. RectangularSkyAperturePhotometryProduct	732
2.332. RectangularSkyAperturePhotometryTask	736
2.333. Regrid	739




2.334. REPEAT	747
2.335. ReplaceFreqRanges	748
2.336. ResampleFrequency	750
2.337. RESHAPE	753
2.338. restore	755
2.339. resume	757
2.340. REVERSE	758
2.341. RgbSimpleImage	759
2.342. Rotate	764
2.343. rotate	767
2.344. RotateTaskSignatureComponent	770
2.345. ROUND	772
2.346. RowByRowAverageSpectrum	774
2.347. save	775
2.348. saveObservation	777
2.349. scale	779
2.350. ScaleTaskSignatureComponent	782
2.351. SelectSpectrum	784
2.352. SerialArchive	787
2.353. SerialClientPool	791
2.354. SHIFT	793
2.355. Short1d	795
2.356. Short2d	796
2.357. Short3d	797
2.358. Short4d	798
2.359. Short5d	799
2.360. SIGCLIP	800
2.361. SIGNUM	803
2.362. simpleAsciiTableReader	804
2.363. SimpleCube	806
2.364. simpleFitsReader	818
2.365. simpleFitsWriter	820
2.366. SimpleImageExplorer	822
2.367. SimpleImage	826
2.368. SimpleStack	840
2.369. SincModel	850
2.370. SineAmpModel	851
2.371. SineModel	852
2.372. SingularValueDecompositionFitter	853
2.373. SingularValueDecomposition	854
2.374. SIN	856
2.375. SINH	858
2.376. SkewGaussModel	859
2.377. SKEWNESS	860
2.378. SkyAperturePhotometryExplorer	862
2.379. SkyAperturePhotometryProduct	864
2.380. SkyAperturePhotometryTask	867
2.381. SlicedCube	869
2.382. SlicedImage	883
2.383. SmoothBaseline	898
2.384. SmoothingTask	900
2.385. SmoothSpectrum	901
2.386. SORT	903
2.387. SourceExtractorDaophotTask	904
2.388. SourceExtractorSussextractorTask	908
2.389. SourceFitTask	912
2.390. SourceFittingExplorer	916
2.391. SourceFittingPanel	920

2.392. SourceFittingProduct	924
2.393. SourceFittingTask	931
2.394. SpectralProjectionAlgorithm	933
2.395. Spectrum1d	934
2.396. Spectrum2d	936
2.397. SpectrumAvgPlotting	938
2.398. SpectrumPlotting	939
2.399. SpectrumStatistics	943
2.400. SpectrumTask	946
2.401. SpireFlags	947
2.402. SQRT	948
2.403. SQUARE	949
2.404. Sso	950
2.405. STDDEV	951
2.406. StitchSpectrum	952
2.407. String1d	955
2.408. SubtractSpectrum	956
2.409. SUM	960
2.410. TablePlotter	962
2.411. TAN	971
2.412. TANH	973
2.413. TaskWrapper	974
2.414. tiledImage	975
2.415. Transform2dTask	976
2.416. translate	977
2.417. TranslateTaskSignatureComponent	979
2.418. TRANSPOSE	981
2.419. transpose	982
2.420. TransposeTaskSignatureComponent	984
2.421. TruncatedLinearInterpolator	986
2.422. UNIQ_SORTED	987
2.423. UNIQ	989
2.424. untar	991
2.425. VARIANCE	992
2.426. VelocityPosMapComputeTask	993
2.427. VelocityPosMapPlotting	995
2.428. VoigtModel	999
2.429. WcsExplorer	1000
2.430. Wcs	1004
2.431. WeightedMean	1030


Categorized view of Commands

This chapter provides a categorized view of all built-in DPfunctions, tasks and objects.











Datasets

-  [NumberedDataset](#) - NumberedDataset is an alternative for the non-existing VectorDataset.
-  [Spectrum1d](#) - Spectrum1d is an extension of AbstractSpectrumDataset which in turn
-  [Spectrum2d](#) - Spectrum2d is an extension of AbstractSpectrumDataset which in turn



Display

-  [ImageAxis](#) - Axes for image display.




Image

-  [AnnotationToolbox](#) - A toolbox to draw annotations.
-  [Display](#) - An Image display for DP. A class to display images.
-  [Flag](#) - A class to describe Flags for images (SimpleImage, SimpleCube, SimpleStack, SlicedImage, SlicedCube and SlicedStack).
-  [RgbSimpleImage](#) - A Product describing three color images.
-  [SimpleCube](#) - A Product describing cubes.
-  [SimpleImage](#) - A Product describing images.
-  [SimpleStack](#) - A SimpleStack is a stack of Images.
-  [SlicedCube](#) - A SlicedCube is a large cube which is stored in slices to make the memory usage lower.
-  [SlicedImage](#) - A SlicedImage is a large image which is stored in slices to make the memory usage lower.
-  [Wcs](#) - A class to create a Wcs.




Input-Output

-  [getObservation](#) - Load an observation into your session.
-  [saveObservation](#) - Save an observation into a local pool.


PAL

-  [DbFactory](#) - An implementation of a ProductPool which saves data to a versant database.
-  [DbPool](#) - An implementation of a ProductPool which saves data to a versant database (to be exact, any
-  [HttpClientFactory](#) - Factory class to create HttpClientPool instances.





class


-  [FitsArchive](#) - A class for reading and writing FITS files.
-  [HduHeaders](#) - A class for working with FITS HDUs. HDU data is loaded only when requested.
-  [SerialArchive](#) - A class for writing and reading serialized objects.

developer


-  [interruptable](#) - Demonstrate the use of the Ctrl-s on top a task


 **generic task**


-  [FitterTask](#) - generic module: FitterTask
-  [GaussFitTask](#) - generic module: GaussFitTask
-  [ModelFitTask](#) - generic module: FitterTask
-  [SourceFitTask](#) - generic module: SourceFitTask

 **numeric**




-  [REBIN](#) - *


 **numerics**


 **interp**

-  [Bilinear](#) - Bilinear interpolation takes 2-D data (i.e. an image), and returns values at arbitrary positions by linearly combining the pixels around the position of interest. Positions are relative to pixel indices, where the center of the top-left pixel is (0,0). No extrapolation is allowed;


 **task**

-  [bg](#) - Execute jython commands in the background
-  [clear](#) - Clear variables from the jython session.
-  [displayLog](#) - Open a window displaying log messages
-  [exportPalToUfDir](#) - Exports observations related to an URN from a pool to a user friendly directory structure
-  [fitsReader](#) - The FitsReaderTask task.
-  [help](#) - The help task for interactive help.
-  [importUfDirToPal](#) - Imports an observation context from an user friendly directory structure into a pool.
-  [localStoreCopier](#) - Creates a (possibly compressed) copy of a Local Store. This copy can be distributed
-  [localStoreWriter](#) - Saves a product in a Local Store, either already defined, or a new
-  [NearestNeighbourProjectionTask](#) - NearestNeighbourProjectionTask
-  [openFile](#) - opens a variable in a viewer
-  [openTask](#) - opens a task in its viewer
-  [openVariable](#) - opens a variable in a viewer
-  [pause](#) - Pause the execution of jython code
-  [restore](#) - Restore variables from a file to a jython session
-  [resume](#) - Resume the execution of jython code
-  [save](#) - Save variables from the jython session to a file
-  [simpleAsciiTableReader](#) - The simpleAsciiTableReader task.
-  [simpleFitsReader](#) - The SimpleFitsReaderTask task.
-  [simpleFitsWriter](#) - Saves a product in a FITS file
-  [SourceExtractorDaophotTask](#) - This task extracts point sources from an HCSS SimpleImage image using DAOPHOT.

 [SourceExtractorSussextractorTask](#) - This task extracts point sources from an HCSS SimpleImage image using SUSSEXtractor.

 [untar](#) - Untars a (relative path) tar file in a user-chosen directory

 **cube**


 [importCube](#) - The ImportCube task for Cubes.


 [importSpectralCube](#) - The ImportSpectralCube task for SpectralSimpleCubes.


 **general**


 [info](#) - A task for inspecting the content of a variable in the jython session.

 **image**


 [clamp](#) - Clamping images and cubes.


 [crop](#) - A Task to crop images and cubes.


 [cutLevels](#) - A Task to calculate the cut levels of an image.

 [ImageSaverTask](#) - A Task to save images.


 [importImage](#) - A Task to import images into an Image object.


 [importRgbImage](#) - A Task to import color images into an RgbImage object.

 [rotate](#) - A Task that rotates images.


 [scale](#) - A Task to scale images.


 [tiledImage](#) - A Task to convert an image to a TiledImage.

 [translate](#) - A Task that translates an image.


 [transpose](#) - A Task that transposes an image.

 **toolbox**

 [RadialVelocityTask](#) - The RadialVelocity Task.


 [Sso](#) - To obtain an Horizons object you need an Ephemeris object and a directory with SSO files


 **utility task**

 [asciiTableReader](#) - AsciiTableReaderTask

 [asciiTableWriter](#) - Task for writing ASCII files.

 [pointHistoryDisplay](#) - Task for displaying graphs of pointing information

 [poolDataReader](#) - Task for reading a product from a pool.

 [poolDataWriter](#) - Task for writing a product into a pool.

How to use this manual

The *User's Reference Manual* contains information about all the main tasks and classes that you can use within your scripts. There are four version of this manual: the *HCSS* version describes the functions provided by the core Herschel software, and is always shipped with HIPE; the three other versions describe functions provided by HIFI, PACS and SPIRE software. You may or may not have these additional manuals, depending on the software you installed.

This manual includes the following sections:

- [Categorized view of Commands](#). All the functions described in the manual, organised by category.
- [Dictionary](#). A list of definitions of words and acronyms you are likely to encounter in documentation on the Herschel satellite.
- [Chapter 1](#). This section describes the *Numeric* library, one of the cornerstones of the HCSS software. You can find more information about this library in the *Scripting and Data Mining* guide: [Chapter 3](#).
- [Section 2.1](#). This section lists all the available commands in alphabetical order. Each command has a description, usage instructions and examples.

II.1. Related documentation

The aim of this manual is to provide reference information for all the user-relevant aspects of the HCSS system. Despite our efforts, you may find that some routines are missing, or have incomplete or inaccurate descriptions. In this case you are encouraged to consult the *Javadoc* developer's reference documentation. You can access the Javadoc by clicking on *HCSS Developer's Reference Manual (API)* in the table of contents of the HIPE Help System.

Guidance on how to use the Javadoc is provided in the *Scripting and Data Mining* guide: [Chapter 5](#).

Some Javadoc pages may have links to more in-depth developer documentation. Be aware that these are *not* fully fledged help documents and are most useful to system developers or advanced users only.

Inspecting the source code

If you are an advanced user versed in Java and Jython, you might want to have a look at the source code of a task or class. You can include source code in your HIPE installation in the following ways:

- If you are installing a developer build via the Continuous Installation System, the `--src=yes` option will install source code. This is enabled by default if you use the `--developer` option.

With the `--unpack=yes` option, the source code will be unpacked into a `src` subdirectory in your HIPE installation. With the `--unpack=no` option, source files for each module will be kept as ZIP files in the repository (usually under `$HOME/.hcss.d/repository` in UNIX systems and under `%HOMEPATH%/.hcss.d/repository` in Windows).

- If you are using an installer, select the checkbox next to the question *Would you like to have the source code installed?* This is only available if you choose the *Advanced* installation. The source code will be in the `src` subdirectory of your HIPE installation.

Dictionary

This section contains a list of definitions of words and acronyms you are likely to encounter in documentation on the Herschel satellite.

Dictionary

AAS	[Acronym] Altitude Anomaly Sensors
ABCL	[Acronym] As-Built Configuration List
ACA	[Acronym] Altitude Control Axis
ACC	[Acronym] Attitude Control Computer
ACM	[Acronym] Attitude Control and Measurement
ACMS	[Acronym] Attitude Control and Measurement System
ACS	[Acronym] Auto-Correlation Spectrometer
<code>activateMasks</code>	[Pipeline] PACS task to activate or deactivate specified masks before running a pipeline task
A/D	[Acronym] Analogue to Digital Converter
AD	[Acronym] Applicable Document
ADC	[Acronym] Analogue to Digital Converter
<code>addUTC</code>	[Pipeline] PACS Level 0 task to convert from onboard time to UTC
ADP	[Acronym] Acceptance Data Package
AIT	[Acronym] Assembly, Integration and Test
AIV	[Acronym] Assembly, Integration, and Verification
AM	[Acronym] Avionics Model
AMA	[Acronym] Absolute Measurement Accuracy
AME	[Acronym] Attitude Measurement Error
AO	[Acronym] Announcement of Opportunity
AOCS	[Acronym] Attitude and Orbit Control System
AOR	[Acronym] Astronomical Observation Request
AOS	[Acronym] Acousto Optic Spectrometer
AOT	[Acronym] Astronomical Observing Template
APD	[Acronym] Absolute Pointing Drift
APE	[Acronym] Absolute Pointing Error
API	[Acronym] Applications Programmer Interface
API	[Generic HIPE; Acronym] Application Programmer Interface. It is an interface that a software program or library implements

	in order to allow other software to interact with it. In the "API" documentation, the whats and hows of the java tasks and classes in HIPE are provided. These will be a little difficult for those unused to writing programs to understand. The documentation is also known as the javadocs or Developer's Reference Manual.
APID	[Acronym] Application Programmer Identifier. The APID number is how the pipelines identifies spectrometers
AR	[Acronym] Acceptance Review
ARE	[Acronym] Absolute Rate Error
associateSkyPosition	[Pipeline] SPIRE photometry Level 0.5 task to add the pointing product to the timelines. Common to jiggle and scan map pipelines
argument	[I/O; Data types; Other] Or parameter, are the variables you send to a task when you call that task
attribute	[I/O; Data types; Other] Is a name for a specific field in a class. A field that denotes a particular characteristic of the class, much like a property of a class. For example, an important attribute of an observation context is the observation ID
auxiliary	[I/O; Data types; Other] Additional products containing information necessary to process raw Herschel data, but which the general user will never need to look at
BACUS	[Acronym] Bolometer Array Calibration Unit for SPIRE
BBID	[Generic HIPE; Acronym] Building block ID = bbtype*65536 + bbnumbers
bbnumber	[Generic HIPE] Building block number
BE	[Acronym] Back End
BEM	[Acronym] Back End Module (LFI)
BER	[Acronym] Bit Error Rate
block	[Generic HIPE] A limited stream of data, e.g. part of an observation, that are logically related to each other (e.g. see calibration block)
BlockTable	[I/O; Data types; Other] PACS. A table showing the organisation of your observation data (your pipeline-processed or being processed product) into logical blocks, e.g. by raster position, wavelength range, nod position... Is also a class of product
boolean	[Generic HIPE] Can be True/False (1/0)
bps	[Acronym] bits per second
BW	[Acronym] Bandwidth
CA	[Acronym] Calibration Analysis (software)
CachedPool	[I/O; Data types; Other] A type of pool (see pool), containing cached products and meta data retrieved from remote pools

<code>calcBsmAngles</code>	[Pipeline] SPIRE photometry Level 0.5 task to convert BSM telemetry in a Y angle and Z angle timeline. Common to jiggle and scan map pipelines
<code>calcBsmFlags</code>	[Pipeline] SPIRE photometry Level 0.5 task to convert BSM telemetry in a Y angle and Z angle timeline. Jiggle pipeline
<code>calibration block</code>	[Generic HIPE] The block, the segment, of your observation when the internal calibration source(s) was observed
<code>calibration product</code>	[Generic HIPE] A product from the calibration store
<code>calibration store</code>	[Generic HIPE] The store of calibration data, necessary to process your data and which comes with HIPE and with your HSA data
<code>calibration tree</code>	[Generic HIPE] The calibration information, held in the calibration store, necessary to reduce Herschel data. It comes with HIPE and with your HSA data and is necessary for pipeline processing
<code>calstore</code>	Generic HIPE. Shorthand for the store of calibration data, which comes with HIPE and with your HSA data
CAP	[Acronym] calibration analysis procedure (a script)
CASSIS	[I/O; Data types; Other] Spectral fitting software (http://cassis.cesr.fr/)
CDD	[Acronym] Configuration Data Document
CEU	[Acronym] Cryo Electronics Unit
<code>CheckDataStructure</code>	[Pipeline] HIFI Level 0.5 task that checks that the datasets included in the <code>HifitimelineProduct</code> have a unique <code>bbnumber</code>
<code>CheckFreqGrid</code>	[Pipeline] HIFI Level 0.5 task that provides information about the frequency ranges observed in the HTP and forms groups of datasets with common LO settings
<code>CheckPhases</code>	[Pipeline] HIFI Level 0.5 task that checks that the Chopper, LO frequency and buffer follow specific patterns prescribed by the observing modes
<code>class</code>	[I/O; Data types; Other] The (java) class of a product defines the type of object it is. In object-oriented programming a class is a construct that is used as a blueprint to create objects, so all objects of a particular class will have the same organisation and definition. Also, <code>CLASS</code> is the name of an (externa) software package for continuum and spectral line analysis
<code>cleanPlateauFrames</code>	[Pipeline] PACS Level 0 task to flag data points at the beginning and end of each chopper movement. Photometry. End of Level 0 pipeline processing
COA	[Acronym] Common Optics Assembly
Co-I	[Acronym] Co-Investigator
COM	[Acronym] Centre of Mass

console	[Generic HIPE] A panel of the HIPE GUI where you can write commands
constructor	[I/O; Data types; Other] In object-oriented programming a constructor in a class is a special block of statements that are called when an object is created declared or constructed)
Context	[I/O: Data types; Other] A Context is a special type of product that contains references to other products stored. Think of it as a bag of other products. This enables a means of building complex data structures. There are two way that these products can be organised in a Context, as a List where they are accessed by index or as a Map where they are accessed as key—value pairs (similar to a python dictionary)
convertChopper2Angle	[Pipeline] PACS Level 0 task to convert chopper instrument positions to sky angle
convertSignal2StandardCap	[Pipeline] PACS Level 0.5 task to convert all data down to the values they would have had if taken at the lowest capacitance setting. Subsequent tasks require this. Spectroscopy
ConvertFrequencyTask	[Pipeline] HIFI Level 1 pipeline task. Converts the IF frequency scale to the sideband frequencies defined by $f_{usb} = f_{LO} + f_{IF}$ and $f_{lsb} = f_{LO} - f_{IF}$. At the same time, the units are changed from MHz to GHz. Can also be used to transform back to the IF or a velocity scale
COP	[Acronym] Commissioning Operations Plan or Commissioning phase (CoP)
corrBolTimeResponse	[Pipeline] SPIRE photometry Level 0.5 task to do the bolometer time response correction. Common to jiggle and scan map pipelines
CP	[Acronym] Calibration Pointing
CQM	[Acronym] Cryogenic Qualification Model
CRE	[Acronym] Cryogenic Readout Electronics (in some documents: Cold Readout Electronics)
createSpirePointing	[Pipeline] SPIRE photometry Level 0.5 task to run the Pointing Task to get the instrument pointing timeline. Common to jiggle and scan map pipelines
CRS	[Acronym] Coarse Rate Sensors
CS	[Acronym] (internal) calibration source
CSDT	[Acronym] Common Software Development Team
CST	[Acronym] Community Support Tools
Cube	[I/O; Data types; Other] a class of product at Level 1 to 2, of 3 dimensions, usually RA and Dec and time/wavelength or frequency
CUS	[Acronym] Common Uplink System
cutPhotDetTimelines	[Pipeline] SPIRE photometry ccan map Level 0.5 task to cut the timeline back to scan line range. End of Level 0.5 pipeline

CVV	[Acronym] Cryostat Vacuum Vessel
D/A	[Acronym] Digital to Analogue converter
DAC	[Acronym] Digital to Analogue Converter
DACS	[Acronym] Digital Autocorrelator Spectrometer
DAE	[Acronym] Data Acquisition Electronics (LFI)
DAPSAS	[Acronym] Data Processing and Science Analysis Software
DBMS	[Acronym] Database Management System
dBmW/MHz	[Acronym] Noise power level in dB relative to 1 mW in a 1 MHz bandwidth
DbPool	[I/O; Data types; Other] A type of pool (see pool) for accessing products from a remote database (e.g. the Versant database)
DBS	[Acronym] dual beam switch
DBU	[Acronym] Digital Bus Unit, the digital interface provided by the spacecraft
DC	[Acronym] Direct Current. Also, DC/DC: Direct Current voltage converter
DC/DC	[Acronym] Direct Current voltage converter
DDS	[Acronym] Data Distribution System
deactivateMasks	[Pipeline] PACS task to deactivate specified masks before running a pipeline task
DEC	[Acronym] Detector Control
DECMEC	[Acronym] electronic board, 2 of them: detector and mechanism controller
deglitchTimeline	[Pipeline] SPIRE photometry Level 0.5 task to do the deglitching for scan maps. Common to jiggle and scan map pipelines
demodulate	[Pipeline] SPIRE photometry Level 0.5 task to perform demodulation on jiggle pipeline data
denodding	[Pipeline] SPIRE photometry Level 0.5 task run to remove nodding from jiggle data. Jiggle pipeline
detectCalibrationBlock	[Pipeline] PACS Level 0 task to identify the calibration blocks and fill the CALSOURCE entry in the status table
Developer's Reference Manual	[Generic HIPE] The documentation that details the whats and hows of java tasks and function in HIPE. These will be a little difficult for those unused to writing programs to understand. Also known as the javadocs or API
DHS	[Acronym] Data Handling System
DHSS	[Acronym] Data Handling Subsystem
dictionary	[I/O; Data types; Other] A type of data array (jython)

dither	[I/O; Data types; Other] Moving sky position a small amount and taking another exposure
DMA	[Acronym] Direct Memory Access
dmchead	[Pipeline] PACS. The Dec Mec (detector mechanism) header. It contains the position and status of the instrument sampled at a high frequency. Values therein are in detector units.
DML	[Acronym] Declared Material List
DMS	[Acronym] Document Management System
DoAntennaTemp	[Pipeline] HIFI Level 1 task to translates to the antenna temperature scale
DoChannelWeights	[Pipeline] HIFI Level 0.5 task to compute the (channel-dependent) weights and fills them into the datasets of type 'science'
DoCleanUp	[Pipeline] HIFI Level 1 task to remove data that is no longer needed after running the Level1 pipeline
DOF	[Acronym] Degree of Freedom
DoFluxHotCold	[Pipeline] HIFI Level 0.5 task that applies the bandpass spectra to the science spectra for the intensity calibration. End of Level 1 pipeline
DoFreqGrid	[Pipeline] HIFI Level 1 task to compute a frequency scale which will be used for resampling
DoHkCheck	[Pipeline] HIFI Level 0.5 task that checks the House Keeping contained in a HifiCalibrationDataset
doHrsCorrSP	[Pipeline] HIFI Level 0 HRS task which corrects HRS spectra from IF non-linearity errors
doHrsCutBandEdges	[Pipeline] HIFI Level 0 HRS task which cuts the edges of the HRS sub-bands, according to the bandpass of the filter. End of HRS pipeline
doHrsFFT	[Pipeline] HIFI Level 0 HRS task which applies a FFT processing on the correlation functions of HRS
doHrsFreq	[Pipeline] HIFI Level 0 HRS task which computes the frequency of the sub-bands
doHrsNorm	[Pipeline] HIFI Level 0 HRS task which normalises the raw correlation functions of HRS
doHrsOffsetPow	[Pipeline] HIFI Level 0 HRS task which computes the offset and power of HRS
doHrsPowCorr	[Pipeline] HIFI Level 0 HRS task which corrects the non-Pipeline.linearity error of the power of HRS.
doHrsQDCFast	[Pipeline] HIFI Level 0 HRS task which corrects roughly the quantisation distortion of the correlation functions
doHrsQDCFull	[Pipeline] HIFI Level 0 HRS task which corrects the quantisation distortion of the correlation functions

doHrsSmooth	[Pipeline] HIFI Level 0 HRS task which applies a Hanning smoothing on the spectra
doHrsSubbands	[Pipeline] HIFI Level 0 HRS task which extracts the HRS subbands from the HRS readout
doHrsSymm	[Pipeline] HIFI Level 0 HRS task which symmetries the correlation function of HRS to prepare the FFT
doHrsWindow	[Pipeline] HIFI Level 0 HRS task which applies a Hanning or Hamming windowing to the correlation functions of HRS
DoOffSubtract	[Pipeline] HIFI Level 0.5 task to compute the differences between ON and OFF positions
DoPointingTask	[Pipeline] HIFI Level 0.5 task that applies the Pointing Product to the HTP to give position of the spectrum
DoRadialVelocity	[Pipeline] HIFI Level 1 task to correct the local oscillator frequency for the radial velocity of the spacecraft
DoRefSubtract	[Pipeline] HIFI Level 0.5 task that is used for the reference subtraction
DoSidebandGain	[Pipeline] HIFI Level 1 task that applies the sideband gain coefficients to the flux values
DoSpurs	[Pipeline] HIFI Level 1 task that removes spurious signals in the scans
DoSubStitch	[Pipeline] HIFI Level 1 task to adjust the subbands relative to each other, possibly removing the overlap
DoWbsBadPixels	[Pipeline] HIFI Level 0 WBS task that applies the masking pixel list
DoWbsDark	[Pipeline] HIFI Level 0 WBS task to subtract the dark values for all scans in a HifiSpectrumDataset
DoWbsFreq	[Pipeline] HIFI Level 0 WBS task that creates a frequency table for each scan
DoWbsNonLin	[Pipeline] HIFI Level 0 WBS task to perform nonlinearity correction for even and odd WBS pixels
DoWbsScanCount	[Pipeline] HIFI Level 0 WBS task to normalise the integration time of each scan to 10 milliseconds
DoWbsSubbands	[Pipeline] HIFI Level 0 WBS task that separates the output from CCDs into the four subbands for the WBS for all HifiSpectrumDatasets contained in the HifiTimelineProduct
DoWbsZero	[Pipeline] HIFI Level 0 WBS task that subtracts appropriate zero spectra for each time step
DP	[Acronym] Data Processing (usually what we call the HIPE data processing language)
DPL	[Acronym] Declared Process List
DPOP	[Acronym] Daily Prime Operational Phase

DPU	[Acronym] Data Processing Unit
DRCU	[Acronym] Detector Readout and Control Unit
DRM	[Acronym] Developer's Reference Manual
DS	[Acronym] Digital Serial
DSB	[Acronym] Double Side Band
DSP	[Acronym] Digital Signal Processor
DTCP	[Acronym] Daily Telecommunications Period
EDAC	[Acronym] Error Detection And Correction
<code>elecCrossCorrection</code>	[Pipeline] SPIRE photometry Level 0.5 task to do the electrical crosstalk correction. Common to jiggle and scan map pipelines
<code>editor</code>	[Generic HIPE] A panel of HIPE in which you can load, write and run scripts and in which other GUI-based tasks will open
EOM	[Acronym] End of Mission
EOP	[Acronym] Early Orbit Phase
EP	[Acronym] Entrance Pupil
EPLM	[Acronym] Extended Payload Module
ESAC	[Acronym] European Space Astronomy Centre
ESD	[Acronym] Electrostatic Discharge
ESOC	[Acronym] European Space Operations Centre
ESTEC	[Acronym] European Space Technology and Research Centre
ESTRACK	[Acronym] European Space Tracking Station Network
EU	[Acronym] Electrical Unit
FCU	[Acronym] Focal Plane Control Unit (HIFI)
f/D	[Acronym] Ratio of focal length to Diameter
FEM	[Acronym] Front End Module (LFI)
FEU	[Acronym] Front End Unit (LFI)
FFT	[Acronym] Fast Fourier Transform
FH	[Acronym] Feed Horn (LFI)
<code>filterSlew</code>	[Pipeline] PACS Level 1 photometry task to remove the data taken during satellite slews
<code>findBlocks</code>	[Pipeline] PACS Level 0 task to organise your data into "blocks" according to an internal logical. Result is put in the BlockTable of your product
<code>fitRamps</code>	[Pipeline] PACS Level 0 task to fit the slope of the averaged or raw ramps, turning a Ramps product into a Frames product. Spectroscopy

flag	[Generic HIPE] A flag in the context of the pipeline is a layer in a data product where data points, or whole pixels, have been identified as (potentially) bad; aka mask. See also quality flag
flagChopMoveFrames	[Pipeline] PACS Level 0 task to mask individual data points that were taken while the chopper was moving. Spectroscopy
flagGratMove	[Pipeline] PACS Level 0 task to mask individual data point that were taken while the grating was moving. Spectroscopy
FM	[Acronym] Flight Model
FoV	[Acronym] Field of View
FP	[Acronym] Fabry-Perot
FPA	[Acronym] Focal Plane Assembly
FPU	[Acronym] Focal Plane Unit
FRD	[Acronym] Formatted Raw Data
frame	[I/O; Data types; Other] A GUI, or a window in a GUI; see also Frames
Frames	[I/O; Data types; Other] Data Frames (or frames) is a generic word used for the datasets that comes from Herschel. Frames is also PACS class of data product that goes through the pipeline
FS	[Acronym] Flight Spare Model
FSS	[Acronym] Fine Sun Sensor
FTB	[Acronym] Focal Plane JFET and/or RF filter box (SPIRE)
FTP	[Acronym] File Transfer Protocol
FTS	[Acronym] Fourier Transform Spectrometer
FWHM	[Acronym] Full Width Half Maximum
GaAs	[Acronym] Gallium Arsenide Semiconductor
GeGa	[Acronym] Germanium Gallium Detector
getCalTree	[Pipeline] PACS task to get the calibration tree, necessary to do any pipeline processing
getObservation	[I/O; Data types; Other] PACS task to get your Observation-Context from pool or from the HSA
glitch	[Generic HIPE] A hiccup in the signal, due e.g. to a cosmic ray. Unlike for optical data, these glitches may affect the response of the immediately subsequent readouts
GO	[Acronym] Geostationary Orbit or Guest Observer
G/S	[Acronym] Ground Station
G/T	[Acronym] Gain to Temperature Ratio.
GT	[Acronym] Guaranteed Time

GTO	[Acronym] Geostationary Transfer Orbit or Guaranteed Time Observer
GUI	[Acronym] Graphical User Interface
GYR	[Acronym] Gyroscope
HAIO	[Acronym] Herschel Archive Interoperability
HBES	[Acronym] HIFI Back-End Sub-system
HBES-1	[Acronym] HBES IF processor
HBES-2	[Acronym] HBES high resolution spectrometer
HBES-3	[Acronym] HBES wide band spectrometer
HCNE	[Acronym] Herschel Confusion Noise Estimator
HCSS	[Acronym] Herschel Common Science System
HDB	[Acronym] HSA Data Base
HDD	[Acronym] HSA Data Distribution
HDP	[I/O; Data types; Other] Herschel Data Processing (usually what we call our data processing language)
HDU	[Acronym] Header Data Unit (FITS)
HEB	[Acronym] Hot Electron Bolometer
HEMT	[Acronym] High Electron Mobility Transistor
HEO	[Acronym] Highly Eccentric Orbit
HET	[Acronym] Heterodyne instrument of the FIRST model payload
HFCU	[Acronym] HIFI Focal Plane Control Unit
HFPU	[Acronym] HIFI Focal Plane Unit
HGA	[Acronym] High-Gain Antenna
HICU	[Acronym] HIFI Instrument Control Unit
HIFI	[Acronym] Heterodyne Instrument for the Far Infrared
HifiPipelineTask	[Pipeline] Task to run all (or any) of the pipeline algorithms for all (or any) of the HIFI spectrometers
HIPE	[Acronym] Herschel Interactive [Data] Processing Environment
HK	[Acronym] House Keeping (data)
H/K	[Acronym] House Keeping (data)
HLCU	[Acronym] HIFI Local Oscillator Control Unit
HLOU	[Acronym] HIFI Local Oscillator Unit
HLS	[Acronym] High-Level Software (LFI)

HOB	[Acronym] Herschel Optical Bench
HOTAC	[Acronym] Herschel Observing Time Allocation Committee
housekeeping	[Generic HIPE] Data about what satellite was doing, its state, during your observation. The general user will never need to look at the housekeeping data
HPBW	[Acronym] Half-Power Beam Width
HPMCS	[Acronym] Herschel/Planck Mission Control System
HRI	[Acronym] High-Resolution IF processor (HIFI)
HRP	[Acronym] High Resolution IF Processor
HRS	[Acronym] High-Resolution Spectrometer (HIFI)
HSA	[Acronym] Herschel Science Archive
HsaReadPool	[I/O; Data types; Other] A type of pool (see pool), to access the Herschel Science Archive
HSC	[Acronym] Herschel Science Centre
HSIA	[Acronym] Hardware/Software Interaction Analysis
HSK	[Acronym] House Keeping
HTP	[Acronym] HifiTimelineProduct, a MapContext containing HifiSpectrumDatasets
HttpClientPool	[I/O; Data types; Other] A type of pool (see pool) but networked (for remote access)
HUI	[Acronym] HSA User Interface
H/W	[Acronym] Hardware
IA	[Acronym] Interactive Analysis
ICC	[Acronym] Instrument Control Centre
ID	[Acronym] Identification
I/F	[Acronym] Interface. Also IF: Intermediate Frequency
IF	[Acronym] Intermediate Frequency
IFP	[Acronym] Intermediate Frequency Processor
IFS	[Acronym] Integral field spectroscopy/spectrograph
IFSS	[Acronym] Intermediate Frequency and Spectrometer System
IFU	[Acronym] integral field unit
ILT	[Acronym] Instrument Level Test
InP	[Acronym] Indium Phosphide semiconductor material
I/O	[Acronym] Input/Output

IPAC	[Acronym] (NASA/JPL) Infrared Processing and Analysis Center
IST	[Acronym] Instrument System Test
javadoc	[Generic HIPE] The documentation that details the whats and hows of java tasks and function in HIPE. These will be a little difficult for those unused to writing programs to understand. Also known as the Developer's Reference Manual or API
JIDE	[Acronym] a previous version of HIPE (jython interactive data-processing environment)
jiggleAverage	[Pipeline] SPIRE photometry Level 0.5 task average on the jiggle position for a number (1-4) of DemodPointingProducts. Jiggle pipeline
Label	[Generic HIPE] A bitword recorded with every PACS frame, indicating where we are in the synchronised mechanism movement sequence (chopper plateau, grating movement, e.t.c.)
LAN	[Acronym] Local Area Network
Level 0	[I/O; Data types; Other] Herschel data can have up to 5 levels, depending on the amount of (mainly pipeline) processing that has been performed. There is a common convention for all Herschel instruments. Level 0 has had no pipeline processing done, and the data are organised in a time sequence (and perhaps other parameters)
Level 0.5	[I/O; Data types; Other] Herschel data can have up to 5 levels, depending on the amount of (mainly pipeline) processing that has been performed. There is a common convention for all Herschel instruments. Level 0.5 has had all basic pipeline processing done so the product can be inspected
Level 1	[I/O; Data types; Other] Herschel data can have up to 5 levels, depending on the amount of (mainly pipeline) processing that has been performed. There is a common convention for all Herschel instruments. Level 1 has had all instrument-correction type pipeline processing done and the data converted to physical units; they may now need to be inspected by the astronomer
Level 2	[I/O; Data types; Other] Herschel data can have up to 5 levels, depending on the amount of (mainly pipeline) processing that has been performed. There is a common convention for all Herschel instruments. Level 2 data should be of science quality
Level 3	[I/O; Data types; Other] Herschel data can have up to 5 levels, depending on the amount of (mainly pipeline) processing that has been performed. There is a common convention for all Herschel instruments. It is hoped that Level 3 data will be astronomical results derived from the data by the user and such results maybe input into the Herschel archive (e.g. source catalogues). Suitable for VO (Virtual Observatory) access
LGA	[Acronym] Low-Gain Antenna
list	[I/O; Data types; Other] An array of data (jython)
list context	[I/O; Data types; Other] see ListContext

ListContext	[I/O; Data types; Other] A type of Context which is a list of individual products (some of which can themselves be lists or other Contexts); see also Context
LOBT	[Acronym] Local On-Board Time
LocalPool	[I/O; Data types; Other] PACS. A way to define where you want your pool to be so you can save a product to it
local store	[I/O; Data types; Other] The local store (by default in [your home directory].hcss/lstore) is a database in the form of a directory structure which contains pools of products in the form of FITS files. These pools are themselves subdirectories of the local store. See also storage, pool, PAL
LocalStore	[I/O; Data types; Other] The local store (by default in [your home directory].hcss/lstore) is a database in the form of a directory structure which contains pools of products in the form of FITS files. These pools are themselves subdirectories of the local store. See also storage, pool, PAL
LOS	[Acronym] Line of Sight
LOU	[Acronym] Local Oscillator Unit
lpfResponseCorrection	[Pipeline] SPIRE photometry Level 0.5 task to do the electrical Low Pass Filter response correction. Common to jiggle and scan map pipelines
LSB	[Acronym] Lower Side Band
LSU	[Acronym] Local Oscillator Synthesiser Unit (HIFI)
LTA	[Acronym] Long Term Archive
LWU	[Acronym] Local Oscillator Waveguide Unit (HIFI)
M3	[Acronym] Third Mirror in the optical train starting with the telescope primary
makeTodArray	[Pipeline] PACS Level 1 photometry task to build a time-ordered data stream for input into MadMap and add to the meta header for the output skymap
MapContext	[I/O; Data types; Other] A MapContext is a Context for products grouped into containers with access to each by a key (similar to a python dictionary). See also Context
mask	[Generic HIPE] A layer in a data product where data points, or whole pixels, have been identified as (potentially) bad. Not the same as quality flags
MCC	[Acronym] Mission Control Centre
M-CDR	[Acronym] Mission Level CDR
MCM	[Acronym] Multi Chip Module
MEM	[Acronym] Maximum Entropy Method
Meta data	[I/O; Data types; Other] The data about an observation that are held in the Meta header (which is much like a FITS header)

Meta header	[I/O; Data types; Other] Information about an observation that is contained in a "header", much like a FITS header.
method	[I/O; Data types; Other] In object-oriented programming a method is a group of (software) instructions that is given a unique name and can be called up at any point by simply quoting the name. In other languages a method is called a function, subroutine or procedure. Example: > myTask.getSomething(); the getSomething() is a method for myTask and it returns an answer that depends on what is (or is not) in the ()
MIP	[Acronym] Mission Implementation Plan
MkFluxHotCold	[Pipeline] HIFI Level 0.5 task to compute the bandpass and receiver temperature for the intensity calibration of the spectra
MkFreqGrid	[Pipeline] HIFI Level 1 pipeline task. At the end of the generic pipeline, the flux of all the resulting scans are resampled to a common frequency grid which this tasks computes
MkOffSmooth	[Pipeline] HIFI Level 0.5 task to calibrate a baseline by processing the measurements of an OFF position in sky
MkSidebandGain	[Pipeline] HIFI Level 1 task that prepares a helper object that provides the means to compute the IF- and LO-Pipeline.frequency dependent sideband gains ratios
MkSpur	[Pipeline] HIFI Level 0 WBS task to identify spurs in WBS HifiTimelineProducts. End of WBS pipeline
MkWbsBadPixels	[Pipeline] HIFI Level 0 WBS task that checks for saturated pixels
MkWbsFluxAtten	[Pipeline] HIFI Level 0 WBS task that analyses the attenuator settings
MkWbsFreq	[Pipeline] HIFI Level 0 WBS task that derives frequency scale from comb spectra
MkWbsZero	[Pipeline] HIFI Level 0 WBS task that checks the zeros and compute an interpolation function to represent zeros for each time
MLI	[Acronym] Multi-Layer Insulation
MOC	[Acronym] Mission Operations Centre
MoI	[Acronym] Moment of Inertia
MOIS	[Acronym] Mission Operations Information System
NAIF	[Acronym] Navigation Ancillary Information Facility
naiveAppMapper	[Pipeline] Level 1 SPIRE basic mapper for jiggle maps. End of Level 1 pipeline
naiveScanMapper	[Pipeline] Level 1 SPIRE task for basic map creation for scan maps. Scan map pipeline
namespace	[Generic HIPE] An abstract container providing context for the items it holds; a context for identifiers (unique expressions written in code)

NaN	[Acronym] Not a Number
navigator	[Generic HIPE] A panel of the HIPE GUI where you can navigate through your disc and import some data into HIPE
NDIU	[Acronym] Network Data Interface Unit
NEFD	[Acronym] Noise Equivalent Flux Density
NEP	[Acronym] Noise Equivalent Power
NFS	[Acronym] Network File System
NHSC	[Acronym] NASA Herschel Science Centre
nodAverage	[Pipeline] SPIRE photometry Level 0.5 task averaging denodded jiggle data, producing AveragedPhotPointingProduct. Jiggle pipeline
OBCP	[Acronym] on board control procedure
OBDH	[Acronym] On-board Data Handling (System)
OBS	[Acronym] On-board Software
observation context	[I/O; Data types; Other] The Level 0 Herschel data that you will get from the HSA. It contains at least the Level 0 data, and all levels above that have been pipeline-processed by the HSA. It is a wrapper for everything you need to process and understand your observation. The ObservationContext is also a class, and (not to confuse you or anything) it is also a type of MapContext. The ObservationContext can contain data of Levels 0 to 2 and can contain other Contexts, eg Auxiliary and Calibration
ObservationContext	[I/O; Data types; Other] The Level 0 Herschel data that you will get from the HSA. It contains at least the Level 0 data, and all levels above that have been pipeline-processed by the HSA. It is a wrapper for everything you need to process and understand your observation. The ObservationContext is also a class, and (not to confuse you or anything) it is also a type of MapContext. The ObservationContext can contain data of Levels 0 to 2 and can contain other Contexts, eg Auxiliary and Calibration
obsid	[I/O; Data types; Other] Observation identification (number)
OBSM	[Acronym] On-board Software Monitoring System
OBSW	[Acronym] On-board Software
OBT	[Acronym] On-Board Time
OD	[Acronym] Operational Day
ODB	[Acronym] Operational Database
ODD	[Acronym] On-demand Processed Data
ODP	[Acronym] On-demand Processing
ODR	[Acronym] On-demand Processing Report
OFD	[Acronym] On the Fly processed Data or Operations Facilities Document

OFPP	[Acronym] On the Fly Processing
OFR	[Acronym] On the Fly Processing Report
OGSE	[Acronym] Optical Ground Support Equipment
OIRD	[Acronym] Operations Interface Requirements Document
operational day	[I/O; Data types; Other] the Herschel day unit
OTAC	[Acronym] Observing Time Allocation Committee
OTF	[Acronym] On-Target Flag
OTF	[Acronym] On the Fly
OU	[Acronym] Optics Unit
outline	[Generic HIPE] A panel of the HIPE GUI where an outline of a highlighted item in the variables panel is presented
packages	[Generic HIPE] A logical way of organising tasks and functionalities that have been written for you to use in HIPE. Also a panel of the HIPE GUI where all packages that have been imported into HIPE are displayed
packet sequence	[Generic HIPE] A sequence of packets of the rawest of raw Herschel data (telemetry, tm, files are made of up packet sequences)
PACS	[Acronym] Photodetector Array Camera and Spectrometer
PAL	[Acronym] Product Access Layer
PDB	[Acronym] Proposal Data Base
PDE	[Acronym] Pointing Drift Error
PER	[Acronym] Proposal Evaluation Report
perspective	[Generic HIPE] A perspective is a presentation of the HIPE system that is made available to the user though a set of "views", where views are panels providing particular capabilities. Each view is a separate panel in the HIPE GUI and together they make a perspective. The main toolbar lists all perspectives you can have
photAddInstantPointing	[Pipeline] PACS Level 0 task to do the first step of the astrometric calibration, associating raster point counter, nod counter or sky line scan counter to the individual frames within an observation. Photometry
photAssignRaDec	[Pipeline] PACS Level 0.5 task to assign RA and Dec to every pixel in the image. Photometry
photAvgDith	[Pipeline] PACS Level 0.5 task to mean-combine the chops, after photDiffChop has been run. Photometry
photAvgNod	[Pipeline] PACS Level 0.5 photometry task to average nod signals per nod cycle
photAvgPlateau	[Pipeline] PACS Level 0.5 task to average all signals on a chopper plateau (chopper not moving time). Photometry

photCombineNod	[Pipeline] PACS Level 0.5 task to combine repeated nod cycles. Photometry
photConvDigit2Volts	[Pipeline] PACS Level 0 task to convert units to Volts. Photometry
photCorrectCrosstalk	[Pipeline] PACS Level 0 task to correct for electronic crosstalk. Photometry
photDiffCal	[Pipeline] PACS Level 0 task to subtract the off and on calibration block data and average the results. Various statistics are calculated. Photometry
photDiffChop	[Pipeline] PACS Level 0.5 task to subtract the off- from the corresponding on-.chops. Photometry
photDiffNod	[Pipeline] PACS Level 0.5 task to subtract the A and B nods, per cycle. Photometry
photDriftCorrection	[Pipeline] PACS Level 0.5 task to apply a correction for drift in the flatfield over the observation. Photometry. End of Level 0.5 pipeline processing
photFlagBadPixels	[Pipeline] PACS Level 0 task to mask pixels that have been pre-identified by the PACS team as bad. Photometry
photFlagSaturation	[Pipeline] PACS Level 0 task to mask individual data points for being saturated. Photometry
photFluxConversion	[Pipeline] SPIRE photometry Level 0.5 task to do the flux conversion. Common to jiggle and scan map pipelines
photGetPointings4PointSource	[Pipeline] PACS Level 0.5 task to get the pointings for the virtual aperture; part of the astrometric calibration. Photometry
photHighPassFilter	[Pipeline] PACS Level 1 photometry task to remove the 1/f noise
photMakeDithPos	[Pipeline] PACS Level 0.5 task to check whether a dither pattern exists, and if so to identify the positions. Photometry
photMakeRasPosCount	[Pipeline] PACS Level 0.5 task to add raster position to the status table. Photometry
photMMTDeglitching	[Pipeline] PACS Level 0 task to detect, mask and remove the effects of cosmic rays (glitches). Photometry
photOptCrossCorrection	[Pipeline] SPIRE photometry Level 0.5 task to do the optical crosstalk correction. Common to jiggle and scan map pipelines
photProject	[Pipeline] PACS Level 1 photometry task to create a map out of images, by adding the images. End of Level 1 pipeline
photProjectPointSource	[Pipeline] PACS Level 1 photometry task to create an image covering the total footprint of a point source of an observation. End of Level 1 pipeline processing
photRespFlatFieldCorrection	[Pipeline] Level 0.5 task to apply the flatfield correction and convert signal to a flux density. Photometry
photShiftDith	[Pipeline] PACS Level 1 photometry task to shift the dither offsets of an observation and then add the images

PHS	[Acronym] Proposal Handling Subsystem
PI	[Acronym] Principal Investigator
P/L	[Acronym] Payload
plastic	[Generic HIPE] The technology used for interaction with the VO (virtual observatory)
PLM	[Acronym] Payload Module
POF	[Acronym] Planned Observations File
pointing product	[Pipeline] PACS. The product that contains the instrument pointing over the entire observation
pool	[I/O; Data types; Other] A pool is a Herschel database on your disc, on an external disc, or online. By default pools are sub-directories off of your local store ([your home directory]/.hcss/lstore). Data therein are in FITS format. In a pool you will usually place data that are related in some way. There are several types of pools, the general user will be most used to Product-Pools. See also local store, storage, PAL
POS	[Acronym] Planned Observation Sequence
product	[I/O; Data types; Other] Herschel data are carried about in structures called products, which are like software onions: they are made up of layers, each of which has its own description and history of the contents, and on each of which specific actions can be taken. Peel the layers to get at the actual data, be that one number or arrays of numbers. Generally Herschel data products include meta data keywords, tables with actual data and the history of the generation of the product
product access layer	[I/O; Data types; Other] PAL. This is a manager to allow you to set up, explore and get data from storage areas. The PAL is also a perspective of HIPE
ProductRef	[I/O; Data types; Other] A reference to a product. Not the actual product but a "link" to it. You can have more than one product "stored" in a ProductRef
product reference	[I/O; Data types; Other] A reference to a product. Not the actual product but a "link" to it. You can have more than one product "stored" in a ProductRef
ProductPool	[I/O; Data types; Other] A type of pool (see pool), containing products
product storage	[I/O; Data types; Other] See storage, because product storage is a "storage" of products. ProductStorage is the name of the java class you use to define a storage
ProductStorage	[I/O; Data types; Other] See storage, because product storage is a "storage" of products, and ProductStorage is the name of the task you can use to define the storage
product viewer	[Generic HIPE] Viewers for looking at products. You can access them from the command line or by clicking on a product in the variables panel/editor panel

PSF	[Acronym] Planning Skeleton File or Point Spread Function
PT	[Acronym] Product Tree
PV	[Acronym] Performance Verification
PVOP	[Acronym] Performance Verification Operations Plan
QA	[Acronym] Quality Assurance
QAR	[Acronym] Quality Assessment Report
QCP	[Acronym] Quality Control Pipeline
QLA	[Acronym] Quick Look Assessment (software)
QM	[Acronym] Qualification Model
QPD	[Acronym] Quality Processed Data
QTR	[Acronym] Qualification Test Review
quality flag	[Generic HIPE] An indication of how well a pipeline task ran
Ramps	[I/O; Data types; Other] A PACS class of product used in the pipeline of Level 0 to 0.5. It has 4 dimensions
raster	[I/O; Data types; Other] Moving sky position by small increments over a larger field to observe a larger area in more dense detail
RCS	[Acronym] Reaction Control System
reference	[I/O; Data types; Other] A pointer to something. For example, when you create an array in HIPE "sey = [4,5,6]", then "sey" is a pointer to [4,5,6]. "sey" is not actually [4,5,6], it just references it
RF	[Acronym] Radio Frequency
RFI	[Acronym] Radio Frequency Interference
RFW	[Acronym] Request for Waiver
RMS	[Acronym] root-mean squared
RPE	[Acronym] Relative Pointing Error
rsrfCal	[Pipeline] PACS Level 0.5 task to correct a spectroscopy Frames product for the relative spectral response function. Absolute correct is done with specRespCal.
RSS	[Acronym] root-sum squared
runMadMap	[Pipeline] PACS Level 1 photometry task to run the java MadMap module. This module uses the maximum-likelihood technique to build a map from an input time-ordered data stream. End of Level 1 pipeline processing
RWA	[Acronym] Reaction Wheel Assembly
SA	[Acronym] Solar Array or Scientific Analysis

SAA	[Acronym] Solar Aspect Angle
SAS	[Acronym] Sun Acquisition Sensors
saveObservation	[I/O; Data types; Other] PACS task to save your Observation-Context to a pool
S/C	[Acronym] Spacecraft
secondDeglitching	[Pipeline] SPIRE photometry Level 0.5 task to do a second level deglitching for jiggle pipeline
SED	[Acronym] Spectral Energy Distribution
SerialClientPool	[I/O; Data types; Other] A type of pool (see pool), to read/write or access a remote pool
SIN	[Acronym] Straylight Induced Noise
S/N	[Acronym] Signal to Noise Ratio
SOVT	[Acronym] System Operation Validation Test
spaxel	[Generic HIPE] A spatial pixel, i.e. the spatial unit of a cube
SPC	[Acronym] Science Programme Committee
specAddInstantPointing	[Pipeline] PACS Level 0 task to add the RA and Dec to the central spaxel. Spectroscopy
specAddNod	[Pipeline] PACS Level 0.5 task to add together corresponding nods. Spectroscopy
specAssignRaDec	[Pipeline] PACS Level 0 task to take the RA and Dec that have been assigned to the central spaxel (specAddInstantPointing) and calculate the pointing for all other spaxels. Spectroscopy
specConvDigits2VoltsPerSec	[Pipeline] PACS Level 0 task to convert the units to Volts/s. For a spectroscopy Frames product
specConvDigits2VoltsRamps	[Pipeline] PACS Level 0 task to convert the units to Volts. For a spectroscopy Ramps product
specCorrectHerschelVelocity	[Pipeline] PACS Level 0 task to correct the wavelength grid for the velocity of Herschel during your observation. Spectroscopy
specCorrectSignalNonLinearities	[Pipeline] PACS Level 0.5 task to correct the Frames data points for the fact that the true slope of the raw ramps (that the frames came from) are not linear. Spectroscopy
specDiffChop	[Pipeline] PACS Level 0.5 task to subtract the off chop data from the corresponding on chop data. Spectroscopy
specDiffCs	[Pipeline] PACS Level 0.5 task to calculate the dark and response from the calibration block, for each pixel. Spectroscopy
specExtendStatus	[Pipeline] PACS Level 0 task to add information to the Status table, based on the previous pipeline processing that you have done. Spectroscopy
specFitSignalDrift	[Pipeline] PACS Level 0.5 task to calculate the drift in the response of the pixels during an observation. Spectroscopy

specFlagBadPixelsFrames	[Pipeline] PACS Level 0 task to add a mask to the bad pixels identified by the PACS team. Spectroscopy
specFlagGlitch-FramesQTest	[Pipeline] PACS Level 0.5 task to add a mask to individual data points that are glitches. Spectroscopy
specFlagOutliers	[Pipeline] PACS Level 1 task to flag individual data points in a PacsCube for being outliers (most likely glitches). Spectroscopy
specFlagSaturationFrames	[Pipeline] PACS Level 0 task to mask individual data points for saturation. Works on a Frames product. Spectroscopy
specFlagSaturationRamps	[Pipeline] PACS Level 0 task to mask individual data points for saturation. Works on a Ramps product. Spectroscopy
specFrames2PacsCube	[Pipeline] PACS task to convert a spectroscopy Frames product to a PacsCube product. End of Level 0.5 pipeline processing
specProject	[Pipeline] PACS task to take the RebinnedCube and project each spaxel and each data point onto a regular RA/Dec grid on the sky. Spectroscopy and end of Level 2 pipeline processing
specRespCal	[Pipeline] PACS Level 0.5 task to correct the data to an absolute flux scale. Spectroscopy
Spectrum1d	[I/O; Data types; Other] Is a class of product that is for holding 1-dimensional spectral data
Spectrum2d	[I/O; Data types; Other] Is a class of product that is for holding 2-dimensional spectral data
spectrum container	[I/O; Data types; Other] SpectrumContainer is a class of product. A container of spectra
SpectrumContainer	[I/O; Data types; Other] Is a class of product. A container of spectra
specWaveRebin	[Pipeline] PACS Level 1 task to convert the 5x5 PacsCube with several spectra in each spaxel to a 5x5 RebinnedCube with one spectrum in each spaxel. Spectroscopy. End of Level 1 pipeline processing
SPF	[Acronym] Single Point Failure
SPG	[Acronym] Software/Software Product Generation
SPIRE	[Acronym] Spectral and Photometric Imaging Receiver
SpireListContext	[Pipeline] Level 1 SPIRE task to create a SpireListContext to be used as input for map making. Scan map pipeline
SPR	[Acronym] Software Problem Report
SPU	[Acronym] Signal Processing Unit
SREM	[Acronym] Standard Radiation Environment Monitor
SRPE	[Acronym] Spatial Relative Pointing Error
S/S	[Acronym] Subsystem

SSB	[Acronym] Single Side Band
SSO	[Acronym] Solar System Object
SSR	[Acronym] Solid State Recorders
Status	[I/O; Data types; Other] PACS. Information about the instrument status, held in tabular format and added to as you pipeline process your observation
storage	[I/O; Data types; Other] Consider a storage a database storing data in an structure that Herschel observations require. A storage is the variable defined in HIPE pointing to where the data is on disc, rather than the name of the directory on disc that actually contains the data. See also pool, local store, PAL
STR	[Acronym] Star Trackers
SUM	[Acronym] Satellite User Manual
Summary Table	[I/O; Data types; Other] HIFI. a table for each spectrometer at each level in the ObservationContext, detailing the components of the observation
SVM	[Acronym] Service Module
SVT	[Acronym] System Validation Test
S/W	[Acronym] Software
TA	[Acronym] Trend Analysis (software) or Telescope Assembly
TableDataset	[I/O; Data types; Other] A dataset held in a of tabular structure. Is also a class of product
task	[Generic HIPE] A self-contained HIPE script that does something to the input data. Also a panel of HIPE in which tasks are listed
TC	[Acronym] Telecommand
temperatureDriftCorrection	[Pipeline] SPIRE photometry Level 0.5 task to do the temperature drift correction. Common to jiggle and scan map pipelines
TM	[Acronym] Telemetry. A TM file contains telemetry data, and has a tm suffix
ToO	[Acronym] Target of Opportunity
toolbox	[Generic HIPE] A set of routines for use in data processing and which you can use if writing your own scripts in HIPE
tuple	[I/O; Data types; Other] A type of data array (jython)
UM	[Acronym] User Manual
URD	[Acronym] User Requirements Document
URM	[Acronym] User Reference Manual
URN	[Acronym] Uniform Resource Name (a string identifier to uniquely identify any product stored in any pool)

URR	[Acronym] User Requirements Review
USB	[Acronym] Upper Side Band
useRemoveBaseline	[Pipeline] Level 1 SPIRE task to set a boolean Flag to switch on and off the baseline removal. Scan map pipeline
variables	[Generic HIPE] A named location in memory that holds a value/s that can be modified. Also, the HIPE GUI variables panel is where all products, arrays, or otherwise defined or set variables are listed. You can do things to the variables by clicking on them
VC	[Acronym] Virtual Channel
view	[Generic HIPE] The individual panels you see in the HIPE GUI are "views". These each allow you to view and thus search through something, or allow you to type commands into HIPE. The main toolbar lists all views you can have
VO	[Acronym] Virtual Observatory
waveCalc	[Pipeline] PACS Level 0 task to calculate the wavelength grid from the grating positions for an observation. Spectroscopy.
wavelengthGrid	[Pipeline] PACS Level 1 task to calculate the wavelength grid of a PacsCube. Some optimisation possible. Spectroscopy
WBC	[Acronym] Wide-Band Spectrometer Control (HIFI)
WBI	[Acronym] Wide-Band Spectrometer IF processor (HIFI)
WBO	[Acronym] Wide-Band Spectrometer AOS (HIFI)
WBS	[Acronym] Wide-Band Spectrometer
WBSPU	[Acronym] Wide Bandwidth Spectrometer Processing Unit
WCS	[Acronym] world coordinate system
WFE	[Acronym] WaveFront Error
WG	[Acronym] Wave Guide
workbench	[Generic HIPE] The main GUI of HIPE in where you will work. Full workbench has more panels that the workbench, but all panels can also be accessed from the main toolbar
XML	[Acronym] Extensible Markup Language

Chapter 1. Numeric

1.1. Introduction

This guide is written with Jython users in mind and has a strong focus on Jython syntax. Nevertheless the information found here may be quite useful for Java developers as well.

Quick start	This may help you starting up.
Import statements	Describes various ways to get access to the functionality in this library.
Constants	Describes a couple of useful constants you can import.
Arrays	All you need to know about arrays that are defined within this library.
Toolboxes	All the functions and procedures within this library are grouped in toolboxes.

1.2. Quick start

The following code serves as a quick introduction that may help using the library.

```
# Get everything from the library:
from herschel.ia.numeric.all import *

# Define an 2d double array, and populate it:
x=Double2d(2,3)
x[0,:]=[1,2,3]
x[1,:]=[4,5,6]
print x

# Compute the result of a simple addition:
y=x+1
print y

# Similarly, but do this by altering x itself:
x+=1
print x

# Using a function:
y=SQUARE(x)
print y
```

1.3. Import statements

This section is not applicable if you access the library components from within a HIPE session, because in these cases all the components are automatically available.

A Jython session does not give automatic access to numeric arrays and/or functions respectively. For that you have to import them into the name-space of your Jython session.

Alternatively you can get finer control over what you import into your session. Because of the layout of the library, you can choose to import arrays definitions only, or even a single array definition.

For those users that are familiar with Java, Jython provides a way to avoid name-space pollution by importing a single name into your session:

```
import herschel
x=herschel.ia.numeric.Double1d([0.,1/3.,0.5,1])
```

```
f=herschel.ia.numeric.toolbox.basic.Basic.SIN
y=f(x)
```

In fact, the import statement above gives you access to the complete herschel code without potential name clashes. Alternatively you could do something like:

```
import herschel.ia.numeric
x=numeric.Double1d([0.,1/3.,0.5,1])
f=numeric.toolbox.basic.Basic.SIN
y=f(x)
```

Importing everything

As a user you may want to import all the arrays and functions of the numeric library in one go. This may be rather convenient as it avoids importing all the bits and pieces yourself.

The easiest way to get access to the library is to import every array and function definition into your name-space using the `all` module:

```
from herschel.ia.numeric.all import *
```

1.4. Constants

The `java.lang.Math` class contains two constants representing π , the ratio of the circumference of a circle to its diameter, and e , the base of the natural logarithms. Although these constants do not belong to the Numeric library, they are described here for being a useful tool in many numerical algorithms.

These constants are the best approximations achievable with real numbers, so you are advised to use them instead of defining these quantities yourself. The following code shows how to import the two constants:

```
HIPE> from java.lang.Math import PI
HIPE> print PI
HIPE> 3.141592653589793
HIPE> from java.lang.Math import E
HIPE> print E
HIPE> 2.718281828459045
```

1.5. Arrays

This chapter describes how to create and access arrays.

Definitions	Description of all supported array types and dimensions.
Creation	Creation and initialization of arrays.
Properties	General information about your array.
Conversions	Implicit and explicit conversions..
Operators	Unary and binary operators that can be applied to arrays.
Methods	Additional mechanisms to access or manipulate your array, such as <code>where</code> , performing functions in-line.

Definitions

The library defines the following arrays:

Type	rank of the array				
	1	2	3	4	5
String	String1d	-	-		
Boolean	Bool1d	Bool2d	Bool3d	Bool4d	Bool5d
Byte (8-bit)	Byte1d	Byte2d	Byte3d	Byte4d	Byte5d
Short (16-bit)	Short1d	Short2d	Short3d	Short4d	Short5d
Integer (32-bit)	Int1d	Int2d	Int3d	Int4d	Int5d
Long (64-bit)	Long1d	Long2d	Long3d	Long4d	Long5d
Float (32-bit)	Float1d	Float2d	Float3d	Float4d	Float5d
Double (64-bit)	Double1d	Double2d	Double3d	Double4d	Double5d
Complex (128-bit)	Complex1d	Complex2d	Complex3d	Complex4d	Complex5d

Creation

There are various ways to create an array:

Creating a numeric array of a particular shape

You can create an array by specifying the size of each dimension and optionally a scalar value that is used to initialize all the elements. In general the syntax looks like:

```
x=ArrayPrefix1d(length[,value])
x=ArrayPrefix2d(m,n[,value])
x=ArrayPrefix3d(m,n,p[,value])
```

where `ArrayPrefix1d` is something like `Int1d`.

Examples:

```
x=Bool1d(1)           # [false]
x=Int1d(3)            # [0,0,0]
x=Int1d(3,4)         # [4,4,4]
x=Double2d(3,4)      # [ [0.0,0.0,0.0], [0.0,0.0,0.0], [0.0,0.0,0.0] -]
x=Complex3d(2,3,4,4-3j)
```



Warning

Caveat for users that come from IDL and FORTRAN: The Jython scripting language (and Java for that matter) is using the *last* dimension as the fastest running index! Therefore, the IDL construct `DBLARR(2,3,4)` is equivalent to the Jython construct `Double3d(4,3,2)`.

Creating a 1d array as a range

Using the following syntax:

```
x=ArrayPrefix1d.range(n)
```

, you can create a one-dimensional array of n elements, containing the values $[0, 1, \dots, n-1]$.

Examples:

```
x=Int1d.range(3)      # [0,1,2]
x=Double1d.range(4)   # [0.0,1.0,2.0,3.0]
x=Complex1d.range(2)  # [0.0+0.0j,1.0+0.0j]
```

Note, this does not work for Bool1d and String1d

Creating a numeric array from a Jython array

You can create a numeric array from a Jython array. Doing so, the library checks whether the Jython array was [jagged](#) or not.

```
jx=[1,2,3]           # a Jython array
x =Int1d(jx)        # a numeric array of rank 1
x =Int1d([1,2,3])   # idem, but in one go

jx=[ [1,2,3], [4,5,6] -] # a Jython array of arrays
x =Double2d(jx)     # a numeric array of rank 2
x =Double2d([ [1,2,3], [4,5,6] -])
```

Making a copy of a numeric array

The assignment $y=x$ will usually mean that the variable names x and y refer to the same values.

For example:

```
x=Int1d.range(3)
y=x
print y      # [0,1,2]

x[0]=-1
print y      # [-1,1,2], as x and y refer to the same contents!!
```

If you would like to create a copy, you have to do that explicitly:

```
x=Int1d.range(3)
y=x.copy()
print y      # [0,1,2]

x[0]=-1
print y      # [0,1,2]
```

Note that this behaviour is not specific to this library, but it is part of the Jython language!

You may want to consult the following link for more insights about [sharing data](#) in numeric arrays.

Converting to another type

You can convert a numeric array to another numeric array of the same shape but of different type:

```
x=Int1d([3,4,5])
y=Double1d(x)
z=Complex1d(x)
x=Int1d(y)
```

Properties

Properties (sometimes called attributes or fields, depending on what computer languages you are familiar with) are named items that tell something about your variable.

An array has several *read-only* properties that can be accessed using the following syntax:

```
variable.property
```

This chapter explains the common properties of an array, using the following example variable:

```
x=Int3d(3,5,4)
```

Class

You can ask the array for its [Definitions](#):

```
print x.__class__# herschel.ia.numeric.Int3d
```

Note the underscores. Actually, this property is not specific to arrays. In fact it can be used on any variable in your Jython session, as this property is defined as part of the Jython language!

Size

The size gives you the number of all elements within the array

```
print x.size      # 60
```

Rank

The rank is the number of dimensions of an array

```
print x.rank      # 3
```

Dimensions

The dimensions property tells you something about the number of elements along each dimension.

```
print x.dimensions      # [3,5,4]
print x.dimensions[0]   # 3
```

Conversions

Often you are dealing with an array of a specific type, but your function expects another type. In those cases the array needs to be converted.

Below we discuss the conversion behaviours:

Implicit Conversion

An implicit conversion is something the *system* is doing for you. A function may need a Double1d as input, but it can also handle an Int1d. It just silently converts it to what is needed.

Examples:

```
x=Double1d.range(3) # [0.0,1.0,2.0]
x[0]=1              # implicitly convert -'1' to -'1.0'
y=Byte1d.range(3)  # [0,1,2]
x+=y                # implicitly convert y to the type of x
```

Explicit Conversion

An explicit conversion is something *you* force upon the system. This may be because a particular function is expecting an Float1d array, but your data is stored in an Int1d array.

With an explicit conversions you can also truncate a floating point array into an integer array; this is called narrowing. The opposite is called widening.

Examples:

```
f=Float1d.range(3)+0.5 # [0.5,1.5,2.5]
d=Double1d(f)          # widening!
```

```
i=Int1d(d)           # narrowing! [0,1,2]
z=Complex1d(d)      # [0.5+0j,1.5+0j,2.5+0j]
```

Operators

The numeric library provides an extensive set of operators that allow you to manipulate the arrays in various ways.

Access

You can access elements in an array by using the square brackets (`[access-part]`) notation. The *access-part* is a comma-separated list of access items, where each item can be an index or a range/slice:

index	The indices run from 0 to n-1, where n is the length of a particular dimension.						
selection	An arbitrary number of indices for a particular dimension, e.g.: <code>Selection([1,3,7,200])</code>						
slice	A slice is specified as: <code>begin:end[:step]</code> , where: <table> <tr> <td>begin</td> <td>starting index. If not specified, it assumes the first element; if negative, it counts backwards from the length of the dimension in question.</td> </tr> <tr> <td>end</td> <td>ending index. The specified index is <i>excluded</i>. If not specified, it takes the length of the dimension in question; if negative, it counts backwards from the length of the dimension in question.</td> </tr> <tr> <td>step</td> <td>step size. Must be a positive number. If not specified, it assumes a default step size of one.</td> </tr> </table> <p>Note, that <code>[:]</code>,<code>[::]</code>,<code>[end:]</code>,<code>[start:]</code>, <code>[::step]</code> are all valid statements, as each field is optional.</p>	begin	starting index. If not specified, it assumes the first element; if negative, it counts backwards from the length of the dimension in question.	end	ending index. The specified index is <i>excluded</i> . If not specified, it takes the length of the dimension in question; if negative, it counts backwards from the length of the dimension in question.	step	step size. Must be a positive number. If not specified, it assumes a default step size of one.
begin	starting index. If not specified, it assumes the first element; if negative, it counts backwards from the length of the dimension in question.						
end	ending index. The specified index is <i>excluded</i> . If not specified, it takes the length of the dimension in question; if negative, it counts backwards from the length of the dimension in question.						
step	step size. Must be a positive number. If not specified, it assumes a default step size of one.						

To illustrate the above we will use the following variables:

```
vector # array data of rank 1
array  # array data of rank 2
cube   # array data of rank 3
```

Rank 1 Arrays

Syntax:

```
# elemental access
vector[i]=value
value=vector[i]

# slices
vector[i:j]=value # value is {scalar/data1d/jython-1d-sequence}
value=vector[i:j] # returns a data1d of the same type as vector
```

Example:

```
x=Int1d([1,2,3,4,5,6])
#      -| -| -| -| -| -|
# index: 0 1 2 3 4 5
x[1]           # 2 as indexing always starts at 0
x[:]          # [1,2,3,4,5,6]
x[1:]         # [2,3,4,5,6]
x[:-1]       # [1,2,3,4,5]
x[2:5]       # [3,4,5]
```

```
x[::2] # [2,4,6]
```

Rank 2 Arrays

Syntax:

```
# elemental access
array[i,j]=value # value is a scalar
value=array[i,j]

# slices
array[i0:i1,j0:j1]=value # value is {scalar/data2d/jython-2d-sequence}
value=array[i0:i1,j0:j1] # returns a data2d of the same type as array

# section
array[i,j0:j1]=value # value is {scalar/data1d/jython-1d-sequence}
array[i0:i1,j]=value
value=array[i0:i1,j] # returns a data1d of the same type as array
value=array[i,j0:j1]
```

Example:

```
x=Int2d([[1,2,3,4],[5,6,7,8],[9,10,11,12]])
x[1,1] # 6
x[1,:] # [5,6,7,8]
x[:,1] # [2,6,10]
x[:,1:-1] # [ [2,3],[6,7],[10,11] -]
```

Rank 3 Arrays

Syntax:

```
# elemental access
cube[i,j,k]=value # value is a scalar
value=cube[i,j,k]

# slices
cube[i0:i1,j0:j1,k0:k1]=value # value is {scalar/data3d/jython-3d-sequence}
value=cube[i0:i1,j0:j1,k0:k1] # returns a data3d of the same type as cube

# section
cube[i,j0:j1,k0:k1]=value # value is {scalar/data2d/jython-2d-sequence}
cube[i0:i1,j,k0:k1]=value
cube[i0:i1,j0:j1,k]=value # returns a data2d of the same type as cube
cube[i,j0:j1,k0:k1]=value
cube[i0:i1,j,k0:k1]=value
cube[i0:i1,j0:j1,k]=value

cube[i,j,k0:k1]=value # value is {scalar/data1d/jython-1d-sequence}
cube[i,j0:j1,k]=value
cube[i0:i1,j,k]=value
value=cube[i,j,k0:k1] # returns a data1d of the same type as cube
value=cube[i,j0:j1,k]
value=cube[i0:i1,j,k]
```

Arithmetic Operators

You can use the following list of operators on arrays that are of numeric type (Byte arrays ... Complex arrays):

OP	Operation	Example	Before	After
*	multiplication	x*2	4	8
/	division	x/2	4	2
+	addition	x+2	4	6

-	subtraction	$x-2$	4	2
**	exponentiation	$x**2$	4	16
%	modulus	$x\%2$	5	1

where **OP** denotes the operator symbol. All arithmetic operators can be used with the following syntax:

```
# classic
arrayNd = arrayNd OP scalar
arrayNd = arrayNd OP arrayNd

# in-line
arrayNd OP= scalar
arrayNd OP= arrayNd
```

where arrayNd is the name of an array variable of N dimensions. The first two lines create a new array. The new array has the same type as the widest type of the two variables in the right-hand-side expression:

```
a=Int1d([1,2,3])
b=Float1d([4,5,6])
c=a*1.          # c is a Double1d
c=a+b          # c is a Float1d
```

The form OP=, we sometimes refer to as an *in-line operator*, as they do not produce a new result, but apply the operation on the left-hand-side. Though this form is functionally equivalent to its counterpart ($z=x$ OP y), there are some subtleties.

The in-line form of the operator does not create an array, but tries to change the left-hand-side. This saves memory and increases speed of the operation.

Note, that the type of the right-hand-side can not be wider than the left-hand-side:

```
x=Int1d([1,2,3])
y=Float1d([4,5,6])
x+=1.          # error: the right-hand-side is a floating point scalar!
x+=y          # error: the right-hand-side is a floating point array!
x*=2          # OK: [2,4,6]
y+=x          # OK: [6.0,9.0,12.0]
```

Logical Operators

Logical operators combine logical expressions, and they are defined for all boolean arrays.

If one defines two variables:

```
x=Bool1d([1,0,1,0])
y=Bool1d([1,0,0,1])
```

then the following logical operators are applicable:

OP	Operator	Example	Result
&	and	$x \& y$	[true,false,false,false]
	inclusive or	$x y$	[true,false,true,true]
^	exclusive or	$x \wedge y$	[false,false,true,true]

Relational Operators

Relational operators compare two arrays on an element-by-element basis, and they are defined for all numeric ordered arrays (thus not for complex arrays).

If one defines two variables:

```
x=Int1d([1,2,3])
y=Double1d([3,2,1])
```

then the following relational operators are applicable:

OP	Operator	Example	Result
<	less than	$x < y$	[true,true,false]
<=	less than or equals	$x <= y$	[true,true,false]
==	equals	$x == y$	[false,true,false]
!=	not equals	$x != y$	[true,false,true]
>=	greater than or equals	$x >= y$	[false,true,true]
>	greater than	$x > y$	[false,false,true]

Methods

This section describes various special methods of all defined arrays.

[Apply and Perform Methods](#)

Methods that allow you to execute a function onto an array either by creating a new result or modifying the input array itself.

[Where](#)

Where methods act like query mechanisms that may help you to filter your data.

[Operator methods](#)

Alternative approach for using operators such as +, -, ... and potential benefits.

[Array reducers](#)

Reducing arrays along a specified dimension

Apply and Perform Methods

In general a Jython User will use the following Jython syntax to apply functions to an array:

```
x=Double1d(...)
y=SIN(SQUARE(x))
```

Method: apply()

Using the Java syntax in Jython provides exactly the same results:

```
x=Double1d(...)
y=x.apply(SQUARE).apply(SIN)
```

Method: perform()

Note that in the examples above two array copies are made. An alternative that mutates the array itself:

```
x=Double1d(...)
x.perform(SQUARE).perform(SIN)
```

Combining

If you want to keep the original data intact, a combination of apply and perform might do the trick:

```
x=Double1d(...)
y=x.apply(SQUARE).perform(SIN)
```

When to use what

In the last example, the SQUARE function is applied to x in order to create a new array; then the SIN function is performed onto the newly created array.

The combination of apply and perform methods allow you to avoid unnecessary creation and initialization of temporary arrays. That is, it allows for optimization of your code

In general, we recommend to prototype your code with the normal syntax. Once satisfied with your algorithm, you may want to tune it using the methods described above.

Where

The `where` method allows you to query and filter your array data. The mechanisms and how to use it are described in this chapter.

Functionality

`where` methods return a Selection object that contains an indexing mechanism into an array. The basic syntax is:

```
q=x.where( predicate -)
y=x[q]           # getting selected elements
x[q]=v          # setting selected elements
```

where:

- x is the array this function is operating on,
- **predicate** - the applied logical function,
- q is a Selection object containing the subscripts of those elements matching specified predicate;
- y - a one dimensional array holding only those elements in x that are defined at subscripts defined in q ;
- v - is a value or an array of values of the same type as x . These values are assigned to those elements in x that are defined at subscripts defined in q .

Selection object

All `where` methods return a selection. A selection object is a container of *long index* subscripts, where a multi-dimensional array is treated as if it would be a one-dimensional array.

Taking a two-dimensional array as an example, the mapping is as follows:

Table 1.1. Mapping of long indices to normal indices in a $m \times n$ array.

	j=0	1	...	n-1
i=0	0	1	...	n-1
1	1*n	1*n+1	...	1*n+n-1
:	:	:	:	:
m-1	(m-1)*n	(m-1)*n+1	...	(m-1)*n+n-1

For example the subscripts ($i=2, j=2$) in a $(m \times n)=(3 \times 4)$ array correspond to the long index= $2*4+2=10$.

Basic Usage

The `where` function works on arrays of any rank, except for an input argument that is a Jython *lambda* expression: this kind of functionality is limited to arrays of rank one.

Below follows the various ways of using where functionality:

Using a mask as your predicate

As described [above](#) the predicate specifies the condition(s) that are evaluated for the array the where method is operating on.

One way of using this method is by providing a mask as a predicate. You can create this mask manually or as a result of a logical expression. The mask must have the same shape as the array it is operating on:

```
x=Double1d( [1,2,3,4,5,6,7,8,9,10] -)

# Applying a manually created mask
q=x.where(Bool1d( [1,0,1,0,1,0,1,0,1,0] -))
print q          # [0,2,4,6,8]
print x[q]       # [1,3,5,7,9]

# Applying a mask resulting from a logical expression:
q=x.where( (x>2).and(x<8) -)          # alternative syntax: (x>2)&(<8)
print q                               # [2,3,4,5,6]
print x[q]                            # [3,4,5,6,7]

# Same, but now on a 2x5 array:
x=Double2d( [ [1,2,3,4,5],[6,7,8,9,10] -] -)
q=x.where( (x>2).and(x<8) -)          # alternative syntax: (x>2)&(<8)
print q                               # [2,3,4,5,6]
print x[q]                            # [3,4,5,6,7]

# Assignment
x[q]=44
print x
# [[1,2,44,44,44],[44,44,8,9,10]]
x[q]=Double1d([33,22,11,22,33])
print x                               # [[1,2,33,22,11],[22,33,8,9,10]]
```

Using Predicate Functions

The where method also works in conjunction with array predicate objects, such as the function IS_FINITE (is finite number).

```
x=Double1d( [1,2,Double.NaN,3,4] -)
q=x.where(IS_FINITE)
print x[q]          # [1,2,3,4]
```

Lambda Predicate Expressions

Lambda expressions are Jython constructs to apply a certain logic to a list of elements:

```
x=Double1d( [1,2,3,4,5,6,7,8,9,10] -)
q=x.where(lambda i: i>2 and i<8)
print x[q]          # [3,4,5,6,7]
```

When to use what

Predicate function and boolean arrays have a better performance than lambda expressions.

Advanced Usage

A selection created on one object can be applied to another object as well. This is demonstrated in the sections below.

Applying a selection along a dimension

Suppose a set of numeric arrays are somehow related, for example -using the objects defined below- the value a1d[i] is related to a2d[i,:] and a3d[:,i,:].

Then we create a selection by e.g. applying the where function on a1d, and apply that selection to a2d and a3d as well as follows:

```
# ---- array definition ----
# 4 elements
a1d=Double1d([1,5,Double.NaN,2])
# 4x3 elements
a2d=Int2d([ [1,2,3],[4,5,6],[7,8,9],[10,11,12] -])

# 2x4x2 elements
a3d=Byte3d([ [[1,2],[2,3],[3,4],[4,5]], [[5,4],[4,3],[3,2],[2,1]] -])

# ---- creating a selection manually ----
mask=Bool1d([1,0,0,1])
q=a1d.where(mask)
print -"Selected indices:",q # [0,3]

# ---- filtering your data ----
print a1d[q]           # [1.0,2.0]
print a2d[q,: ]       # [ [1,2,3],[10,11,12] -]
print a3d[:,q,:]      # [ [[1,2],[4,5]], [[5,4],[2,1]] -]

# ---- creating a selection using predicate function ----
q=a1d.where(IS_FINITE)
print -"Selected indices:",q # [0,1,3]

# ---- filtering your data ----
print a1d[q]           # [1.0,5.0,2.0]
print a2d[q,: ]       # [ [1,2,3],[4,5,6],[10,11,12] -]
print a3d[:,q,:]      # [ [[1,2],[2,3],[4,5]], [[5,4],[4,3],[2,1]] -]
```

Applying a selection on an array of different shape

The example above showed how to apply a selection along a particular dimension. This section is making use of the *long indexing* features of arrays as described in section [Jython Usage->Arrays->Operators->Access](#).

In short the idea is that a Selection *can* be considered as a set of long indices. For example the indices $[i, j]=[1, 2]$ in a 4x3 array corresponds to the long index $li=1*3+2=5$. Thus:

```
# ---- definitions ----
a1d=Int1d( [4,1,3,9,2,7,4,5] -)
a2d=Double2d([ [1,2,3,4], [4,3,2,1] -])

# ---- query ----
q=a1d.where( (a1d > 2).and(a1d < 8).and(a1d != 4) -)
print q
# [2,5,7]

# ---- query used as long index ----
print a1d[q] # [3,7,5]
print a2d[q] # [3.0,3.0,1.0]
```

Manipulating and creating Selections

The where method creates a Selection for you as specified by the predicate argument of this method. It is possible however to create and/or manipulate the indices yourself by converting a selection to and from an Int1d array as follows:

```
# create a selection from an integer array
q=Selection([1,2,6]) # holding indices 1,2 and 6

# extracting the indices from a selection as an Int1d array
x=q.toInt1d()
x[2]=7

# recreate the modified selection from a Int1d array
q=Selection(x)
```

One possible usage is these conversions may be a function that manipulates indices, and the best way to manipulate them is as if they were elements in an integer array (including all the access possibilities and applicable functions). For example if one would like to change the order of of the selection:

```
x=Int1d([1,2,3,4,5])
q=x.where(x > 2) # indices: [2,3,4]
print x[q]      # values: [3,4,5]

q=Selection(REVERSE(q.toInt1d()))
print x[q]      # values: [5,4,3]
```

Operator methods

The numeric library provides syntax constructs that allow for abbreviated style (see [the section called “Operators”](#)) of writing binary and unary operators, which is not possible in Java. As the library is written for Jython and Java users, these operators are available in the form of methods as well.

There are subtle differences:

- All operators that could be written as inline operators are implemented as methods that mutate the array it operates on.
- All operator methods are explicit, that is no conversions apply. For example adding an array to say an Int1d array only works if the array that is added is an Int1d array as well.

Operator methods

Arithmetic operators

All inline operators such as +=, -=, *= have equivalent methods named add, subtract, divide and so on.

Relational operators

All operators such as ==, !=, < and so on have equivalent methods named eq, ne, lt and so on.

Logical operators

All logical operators such as &, !, | and ^ have equivalent methods named and, not, or and xor respectively. The latter methods mutate the logical array it is operating on:

```
a.and(b).or(c)          # The contents of -'a' are altered!
r=a.copy().and(b).or(c) # The contents of -'a' are altered!
```

When to use what

Operator methods that work inline are as fast as their abbreviated syntax (e.g. +=). The real benefit comes from a style of writing:

```
y=SQRT( SQUARE(SIN(x)/n) + 1 -)
y=x.apply(SIN).divide(n).perform(SQUARE).add(1).perform(SQRT)
```

The code snippet above has reduced the creation of temporary arrays from four down to zero!

Array reducers

Numeric arrays offer the methods `reduce` and `reduceDimX`, where X goes from 0 to the array dimension minus one. For instance, a Double3d array has the methods `reduceDim0`, `reduceDim1` and `reduceDim2`.

These methods take as argument an *array reductor*, that is, a function that reduces the number of dimensions of the array. The available array reducers are ALL, ANY, MIN, MAX, PRODUCT and SUM.

Let us build a `Double2d` array as example:

```
HIPE> a = Double2d([[2, 4, 5], [3, 2, 8], [6, 8, 3]])
HIPE> print a
[
[2.0,4.0,5.0],
[3.0,2.0,8.0],
[6.0,8.0,3.0]
]
```

Using for instance the `SUM` function, the `reduce` method turns the array to a scalar:

```
HIPE> print a.reduce(SUM)
41.0 # Sum of all elements
```

This is the same as writing `print SUM(a)`.

The other two available reduction methods are the following:

```
HIPE> print a.reduceDim0(SUM)
[11.0,14.0,16.0] # Sum of each column
HIPE> print SUM(a, 0) # Equivalent syntax
[11.0,14.0,16.0]
HIPE> print a.reduceDim1(SUM)
[11.0,13.0,17.0] # Sum of each row
HIPE> print SUM(a, 1) # Equivalent syntax
[11.0,13.0,17.0]
```

Note the apparently unexpected behaviour of functions like `MEDIAN`:

```
HIPE> print MEDIAN(a, 1) # Works as expected
[4.0,3.0,6.0]
HIPE> print a.reduceDim1(MEDIAN) # Causes an error
TypeError: reduceDim1(): 1st arg can't be coerced to
herschel.ia.numeric.toolbox.ArrayReductor
```

The reason is that the functions accepted by the `reduce` methods build their result gradually as they go through the elements of the array. This is not possible for functions like `MEDIAN`, which require previous knowledge of all the elements of the array.

1.6. Toolboxes

Introduction

Toolboxes are add-ons that extend the functionality of numeric arrays. All the functions and procedures within this library are grouped in toolboxes.

Each toolbox has its own set of specialized functions, documentation and example code. If you click on one of the toolboxes on the left, you will get more information about the functionality it provides.

Currently available toolboxes are:

The Basic Toolbox	Basic functions like <code>SIN</code> , <code>SQUARE</code> , <code>MIN</code> , ...
The Fit Toolbox	Provides functionality for fitting.
The Filter Toolbox	Provides filter functions and utilities useful for processing time-series and images.
The Integration Toolbox	Provides numeric integration methods, for both continuous functions and discrete data.

The Interpolator Toolbox	Provides functions and utilities for interpolation and extrapolation.
The Mask Handling Toolbox	Provides facilities for handling masks that are stored in numeric library arrays.
The Matrix Toolbox	Set of functions and utilities to manipulate matrices.
The Random Functions Toolbox	Set of functions for populating arrays with random elements.
The XFORM Toolbox	Provides Transformational functions such as FFT.

The Basic Toolbox

This toolbox provides basic functions in the following categories:

- [General Functions](#)
- [Trigonometric Functions](#)
- [Nearest Integer Functions](#)
- [Reduction Functions](#)
- [Statistical Functions](#)

name-space: `herschel.ia.numeric.toolbox.basic`

General Functions

- [SIGCLIP](#)
- [REVERSE](#)
- [Rotate](#)
- [SQUARE](#)
- [ExpN](#)
- [Polynomial](#)
- [AnyPresent](#)
- [LOG10](#)
- [IS_INFINITE](#)
- [NAN_FILTER](#)
- [LogN](#)
- [IS_FINITE](#)
- [SQRT](#)
- [RESHAPE](#)
- [SHIFT](#)
- [CONCATENATE](#)

- [SIGNUM](#)
- [EXP10](#)
- [EXP](#)
- [NotPresent](#)
- [ABS](#)
- [LOG](#)
- [MedianAbsoluteDeviation](#)
- [UNIQ](#)
- [IS_NAN](#)
- [UNIQ_SORTED](#)
- [Pow](#)
- [SORT](#)

Trigonometric Functions

- [COSH](#)
- [TAN](#)
- [ARCCOS](#)
- [SIN](#)
- [ARCTAN](#)
- [ARCSIN](#)
- [COS](#)
- [TANH](#)
- [SINH](#)

Nearest Integer Functions

- [ROUND](#)
- [CEIL](#)
- [FIX](#)
- [FLOOR](#)

Reduction Functions

This family of functions reduces a sequence of values a single value by combining elements via a supplied function. Additionally, this mechanism can be used to reduce an array of rank N to an N-1 array.

The general syntax of reduction functions is as follows:

```
y=f(x [,dim])
```

where: x is the input array of any rank, dim an optional dimension and y is the result of the computation. If the optional dimension is not specified, the array is reduced to a single scalar value.

If the dimension is specified the function computes the result by reducing the array elements along that dimension into a single value; the resulting array has a rank which is equivalent to $x.rank-1$.

For example, consider an arbitrary reduction function f and the following two-dimensional array

```
-| 1 2 -|
x=| 3 4 -|
```

Then:

```
f(x) = f([1,2,3,4])
f(x,0) = [ f(1,3), f(2,4) -]
f(x,1) = [ f(1,2), f(3,4) -]
```

- [MIN](#)
- [PRODUCT](#)
- [SUM](#)
- [AllPresent](#)
- [ALL](#)
- [ANY](#)
- [MAX](#)

Statistical Functions

The following statistical functions accept integral as well floating-point arrays; the former implicitly transformed to a double array.

You have to filter out non-finite elements (such as NaN elements) as these functions produce sensible results only if all elements have finite values:

```
x=DoubleId([Double.NaN,1,Double.NaN,3,2,3,4,Double.NaN,4])
print MEDIAN(x) # ERROR: the median is not NaN
print MEDIAN(x[x.where(IS_FINITE)]) # 3.0
```

- [Correlate](#)
- [STDDEV](#)
- [GAMMALN](#)
- [VARIANCE](#)
- [CorrelateMatrix](#)
- [SKEWNESS](#)
- [GammaP](#)
- [MEAN](#)
- [GammaQ](#)

- [KURTOSIS](#)
- [Erfc](#)
- [MEDIAN](#)
- [Erf](#)

The Filter Toolbox

This toolbox provides filter functions.

```
name-space: herschel.ia.numeric.toolbox.filter
```

- [GaussianFilter](#)
- [Convolution](#)
- [BoxCarFilter](#)

The Fit Toolbox

The fitter toolbox provides a generalized way to fit models to data and to estimate the best values of the model parameters. An introduction to the use of fit package can be found [here](#).

Before proceeding it is wise to familiarize yourself with the [background information](#) about the Fitter classes.

To get started we present [worked examples](#) on almost all aspects of of the package.

```
name-space: herschel.ia.numeric.toolbox.fit
```

The Integration Toolbox

The integration toolbox provides a generalized way to compute integrals, through different numeric methods.

```
name-space: herschel.ia.numeric.toolbox.integr
```

- [FitterFunction](#)
- [Integrator](#)

The Interpolator Toolbox

This toolbox provides interpolation functions.

```
name-space: herschel.ia.numeric.toolbox.interp
```

- [NearestNeighborInterpolator](#)
- [PreviousInterpolator](#)
- [CubicSplineInterpolator](#)
- [LinearInterpolator](#)
- [Regrid](#)

The Mask Handling Toolbox

This toolbox provides facilities for mask handling.

```
name-space: herschel.ia.numeric.toolbox.mask
```

- [PackedMask](#)
- [FixedMask](#)

The Matrix Toolbox

This toolbox provides matrix functions.

```
name-space: herschel.ia.numeric.toolbox.matrix
```

- [LUDecomposition](#)
- [SingularValueDecomposition](#)
- [INVERSE](#)
- [TRANSPOSE](#)
- [QRDecomposition](#)
- [EigenvalueDecomposition](#)
- [MatrixMultiply](#)
- [DETERMINANT](#)
- [MatrixSolve](#)
- [CholeskyDecomposition](#)

The Random Functions Toolbox

This toolbox provides random functions.

```
name-space: herschel.ia.numeric.toolbox.random
```

- [RandomPoisson](#)
- [RandomGauss](#)
- [RandomUniform](#)

The XFORM Toolbox

This toolbox provides transformation functions and utilities.

```
name-space: herschel.ia.numeric.toolbox.xform
```

- [IFFT](#)
- [FFT](#)
- [HAMMING](#)

- [HANNING](#)
- [PowerSpectrum](#)

Chapter 2. DP Commands

2.1. Introduction

This chapter is a complete listing of all built-in DP functions, task and objects, collectively referred to as commands.

How to use this chapter

All of Dp's commands are documented alphabetically in this chapter. A first section gives a general overview what a command is used for and how to use it. This includes a general description and some examples. The "API summary" section provides a quick overview of the available constructors, methods and properties of a command. It is intended as quick reference for users that just need to remember a certain functionality. The "API details" section provides more detailed information about the constructors, methods and properties. Furthermore it provides links to other references and information about the command's history.

Overview

A table gives the full name of the command, indicates if it is written in Java or Jython and if it is just an ordinary command or if it follows the HCSS Task specifications. The import statement in this table can be copied to a Jython console.

Description

General description of the command.

Examples

Most commands include one or more examples that demonstrates how the command is used. Most of the examples are just one or two lines of DP code that can be entered at the Jython prompt. Others are code fragments or routines designed to serve as an examples for your own programs.

Limitations and Miscellaneous

The "Limitations" and "Miscellaneous" sections describes the boundaries of applicability and other specialties of the command.

API Summary

The API Summary provides a quick summary of the commands constructor, method, and properties. Note that most of the arguments are positional parameters that must be supplied in the order indicated by the command's syntax. However arguments can be often supplied by specifying their names. If its the case then they can provided in any order.

Constructor

Constructors are used to create a command.

Method

Methods are functions of a command that can be executed.

Properties

Properties are fields of a command that can be read or written by a command. INPUT properties contain values that are required by a command, OUTPUT properties contain the result of a command, INOUT provide both functionalities at the same time.


See also

The "See also" section provides links to related commands, to other manuals, or to external resources in the Internet.

History

The "History" section describes the changes occurred to the command.

2.2. ABS

Full Name:	herschel.ia.numeric.toolbox.basic.Abs
Alias:	ABS
Type:	Java Class - 
Import:	from herschel.ia.numeric.toolbox.basic import Abs

Description

Gives the absolute value of a number or a numeric array. For complex numbers, ABS(x) gives the modulus.

Example

Example 1: Apply ABS on a Int1d
<pre>x=Int1d([-1,0,1] -) print ABS(-123), ABS(x) # 123 [1,0,1]</pre>

API Summary


Jython Syntax
<y>=ABS (<x>)
Properties
any number and numeric type x [INPUT, MANDATORY, default=no default value]
any number and numeric type y [OUTPUT, MANDATORY, default=no default value]

API details

Properties

any number and numeric type x [INPUT, MANDATORY, default=no default value]
Accepts any number and numeric type.
any number and numeric type y [OUTPUT, MANDATORY, default=no default value]
For complex numbers, ABS(z) gives the modulus.

2.3. AbstractImageManipulationTask

Full Name:	herschel.ia.toolbox.image.AbstractImageManipulationTask
Type:	Java Task - 
Import:	from herschel.ia.toolbox.image import AbstractImageManipulationTask

Description

An abstract Task for image manipulation.

An abstract Task for image manipulation.

API Summary


Property
Image <code>image</code> [INPUT, MANDATORY, default=No default value]

API details

Property

<code>Image image [INPUT, MANDATORY, default=No default value]</code>
The input image.

2.4. AbstractMappingTask

Full Name:	herschel.ia.toolbox.image.AbstractMappingTask
Type:	Java Task - 
Import:	from herschel.ia.toolbox.image import AbstractMappingTask

Description

An abstract Task for mapping.

An abstract Task where the overlap of an input pixel with (smaller) output pixels must be determined.

API Summary


Property
boolean oversample [INPUT, OPTIONAL, default=Default value : true]

API details

Property

<code>boolean oversample [INPUT, OPTIONAL, default=Default value : true]</code>
Indication whether oversampling should be done.

2.5. AbstractModel

Full Name:	herschel.ia.numeric.toolbox.fit.AbstractModel
Type:	Java Class - 
Import:	from herschel.ia.numeric.toolbox.fit import AbstractModel

Description

AbstractModel implements the common parts of (compound) model functions

which can be handled by the various Fitter classes.

An introduction to the use of fit package can be found [here](#).

At least once have a look at the [background](#) on the organization and structure of the package. The fit/demo directory contains worked [examples](#) on almost all aspects of of the package.

Example


Example 1: x = Double1d.range(10)

```

poly = PolynomialModel( 2 -)          # quadratic model
poly.setParameters( Double1d([3,2,1]) -) # set the parameters for the model
y = poly( x -)                        # evaluate the model at x
p0 = poly[0]                          # 3: the first parameter
gauss = GaussModel()                  # gaussian model
gauss += PolynomialModel( 0 -)        # gaussian on a constant background
print gauss.getNumberOfParameters()   # 4
lolim = Double1d([0,-10,0,-5])        # lower limits for the parameters
hilim = Double1d([10,10,2, 5])        # high limits for parameters
gauss.setLimits( lolim, hilim -)      # set limits. Does not work with all
Fitters
index = Int1d([0,3])
value = Double1d([4.5,0])
gauss.keepFixed( index, value -)      # keep index parameters fixed at value

```


2.6. AddSpectrum

Full Name:	herschel.ia.toolbox.spectrum.AddSpectrum
Type:	Java Task - 
Import:	from herschel.ia.toolbox.spectrum import AddSpectrum

Description

Task for adding a scalar to the flux data included in a spectrum container or for adding two spectrum containers on a spectrum-by-spectrum basis (computed as an average).

For the scalar mode, 'ds' and 'param' need to be specified - for the pair-wise mode, 'ds1' and 'ds2' need to be set. In the pairwise mode, the task iterates through the two containers and computes the pair-operation. In case the size of the two containers is different, the iteration stops at the minimum size. Hence, the remaining spectra in the larger container are ignored. In case one of the datasets only contains a single spectrum, the task always refers to this single spectrum while iterating over all the spectra in the other dataset.

Different selection schemes are available for selecting the point spectra or the segments to be processed. The most simple scheme is to specify lists of point spectrum indices (`selection=[1, 3, 4, 2]`). In this case, the operation (scalar- or pair-operation) is just taken over the point spectra with corresponding indices. Instead of specifying a list of indices (which requires the knowledge of these) other recipes to specify a selection are available. Typical recipes test attribute values to match certain values or to be included in a given interval. These recipes we refer to as selection models. The different options to specify selections can be combined. Here, an AND logic is adopted.

Similarly, you can specify what segments to consider. In the most simple case, you can just specify the indices of segments to be considered (for the scalar operation `segments=[1, 3, 4]` and for the pair operation `segments1=[1, 3, 4]`, `segments2=[3, 6, 5]`). In more advanced situations you can use a `SegmentSelection` object. Note that the pairs of segments to be processed need to be consistent - otherwise the processing will fail.

You can specify whether the original datasets should be overwritten with the 'overwrite' flag. By default no data is overwritten. In the pair-wise mode, it is not always possible to overwrite the data - in particular if non-trivial segment selections are specified.

For the pair-wise mode, there are further options that can be selected by the 'variant' (e.g. whether to use flags and weights).

Example

Example 1: from Jide:

```
from herschel.ia.toolbox.spectrum import AddSpectrum
add = AddSpectrum()
spectraOut = add(ds1=spectral1, ds2=spectra2)
spectraOut = add(ds=spectra, param=2.1)
spectraOut = add(ds=spectra, selection={"bbtype": [6031]}, param=2.1)
spectraOut = add(ds=spectra, selection=[0,1,2,3], param=2.1)
spectraOut = add(ds=spectra, selection={"Chopper": ([-4.4, 5.9], 0.2), "-bbtype": [6031]}, param=2.1)
spectraOut = add(ds=spectra, selection={"LoFrequency": (4000.0, 5000.0)}, param=2.1)
spectraOut = add(ds=spectra, selection={"bbtype": [6031]}, segments=[2,3], param=2.1)
```

API Summary

Properties
SpectrumContainer ds [INPUT, OPTIONAL, default=no default value.]
Double param [INPUT, OPTIONAL, default=no default value.]
Object selection [INPUT, OPTIONAL, default=None.]
PyDictionary Map< > selection_lookup [INPUT, OPTIONAL, default=no default value.]
PyList selection_index [INPUT, OPTIONAL, default=No default value.]
Boolean overwrite [INPUT, OPTIONAL, default=False.]
SpectrumContainer ds1 [INPUT, OPTIONAL, default=no default value.]
SpectrumContainer ds2 [INPUT, OPTIONAL, default=no default value.]
Object segments [INPUT, OPTIONAL, default=no default value.]
Object segments1 [INPUT, OPTIONAL, default=no default value.]
Object segments2 [INPUT, OPTIONAL, default=no default value.]
String variant [INPUT, OPTIONAL, default=no default value.]
SpectrumContainer result [OUTPUT, OPTIONAL, default=no default value.]

API details

Properties

SpectrumContainer ds [INPUT, OPTIONAL, default=no default value.]
Input container to be processed by the task in case the task is used as a scalar operation.
Double param [INPUT, OPTIONAL, default=no default value.]
The scalar parameter to be considered when the task is used as scalar operation.
Object selection [INPUT, OPTIONAL, default=None.]
Specify what point spectra the operation should be applied to. Different ways to specify these selections are possible: <ul style="list-style-type: none"> • Specify a list of indices (in jython) of the point spectra for which the add should be applied. • Use a dictionary (in jython) to specify a selection model (lookup for attributes, filter attributes to be included in ranges or intervals). • Pass a string representation of a jython list or a jython dictionary. The string will be parsed and the processing delegated to the two cases above. • Pass any java instance that implements the SelectionModel interface (for the advanced user).
PyDictionary Map< > selection_lookup [INPUT, OPTIONAL, default=no default value.]
Specify a PyDictionary with keys given by the attribute name to look up and a list of admissible values. Behind the scenes, a DiscreteValueSelectionModel is instantiated.

PyList selection_index [INPUT, OPTIONAL, default=No default value.]

Specify a PyList with the indices of the point spectra to be considered.

Boolean overwrite [INPUT, OPTIONAL, default=False.]

Specify whether the input data container can be reused - the values found therein is overwritten.

SpectrumContainer ds1 [INPUT, OPTIONAL, default=no default value.]

First input container for pair-wise operations.

SpectrumContainer ds2 [INPUT, OPTIONAL, default=no default value.]

Second input container for pair-wise operations.

Object segments [INPUT, OPTIONAL, default=no default value.]

Specify what segments the operation should be applied to. There are two options available:

- Either pass an instance of a SegmentSelection which gives the information on what segments for each point spectrum included in the container.
- Specify a PyList of segment indices.

Object segments1 [INPUT, OPTIONAL, default=no default value.]

Specify the segment selection to be associated with 'ds1'.

Object segments2 [INPUT, OPTIONAL, default=no default value.]

Specify the segment selection to be associated with 'ds2'.

String variant [INPUT, OPTIONAL, default=no default value.]

Specify the variant of processing mode you would like to run (including flags / weights / ...). HCSS defaults include: "flux" / "flux-wave" / "flux-wave-weight" / "flux-flag-wave" / "flux-weight-flag-wave"

SpectrumContainer result [OUTPUT, OPTIONAL, default=no default value]

Result object containing the results of the operation applied.


See also

- [ArithmeticSpectrumTask](#)

History

- 2007-08-17 - meli: Initial.
- 2008-03-16 - meli: major refactoring of the toolbox.
- 2008-10-03 - meli: segments, javadoc

2.7. ALL

Full Name:	herschel.ia.numeric.toolbox.basic.All
Alias:	ALL
Type:	Java Class - 
Import:	from herschel.ia.numeric.toolbox.basic import All

Description

Determines whether all elements in the array evaluate to true.

The input array must be of integral type (e.g. bytes,integers) and the result is a boolean array.

Example

Example 1: Apply ALL on a Int1d: check if all the array items, in the specified dimension, are lower than '3'

```
x=Int2d( [ [1,2], [1,4], [1,6] -])
print ALL(x<3)      #all items => 0 (false)
print ALL(x<3, 0) #dim 0 => [ ALL(Int1d([1,1,1]) < 3) -, ALL(Int1d([2,4,6]) <
3) -] = [true, false]
print ALL(x<3, 1) #dim 1 => [ ALL(Int1d([1,2]) < 3) -, ALL(Int1d([1,4]) < 3),
ALL(Int1d([1,6]) < 3) = [true, false, false]
```

API Summary

Jython Syntax

```
<y>=ALL(<array_expression>[,<dimension>])
```

Properties

any expression with an integral type **x** [INPUT, MANDATORY, default=no default value]

integer **dimension** [INPUT, OPTIONAL, default=all the array items]

a boolean array **y** [OUTPUT, MANDATORY, default=no default value]

API details

Properties

any expression with an integral type **x** [INPUT, MANDATORY, default=no default value]

The input array must be of integral type (e.g. bytes,integers) See example.


integer **dimension** [INPUT, OPTIONAL, default=all the array items]

The dimension where to apply the expression.

a boolean array **y** [OUTPUT, MANDATORY, default=no default value]

The result is a boolean array

2.8. AllPresent

Full Name:	herschel.ia.numeric.toolbox.basic.AllPresent
Type:	Java Class - 
Import:	from herschel.ia.numeric.toolbox.basic import AllPresent

Description

Tests whether all of the bits in specified mask are present in the elements of the input array.

The input array must be of integral type (e.g. bytes,integers) and the result is a boolean array.

Example

Example 1: Apply AllPresent on a Int1d: test if a binary 'and' comparison, item by item, with '3' (as mask) is equals to '3'

```
x=Int1d([0,1,2,3])
print AllPresent(3)(x)
#=> [ (0 & 3) == 3, (1 & 3) == 3, (2 & 3) == 3, (3 & 3) == 3 -] =
[false,false,false,true]
print x.apply(AllPresent(3)) #Another way for using AllPresent
```

API Summary

Jython Syntax

```
<y>=AllPresent(<bitmask>)(<x>)
```

Properties

any integral type **x** [INPUT, MANDATORY, default=no default value]

any number **bitmask** [INPUT, MANDATORY, default=no default value]

a boolean array **y** [OUTPUT, MANDATORY, default=no default value]

API details

Properties

any integral type **x** [INPUT, MANDATORY, default=no default value]

The input array must be of integral type (e.g. bytes,integers)

any number **bitmask** [INPUT, MANDATORY, default=no default value]

Accepts any number.


a boolean array **y** [OUTPUT, MANDATORY, default=no default value]

The result is a boolean array

See also

- [AnyPresent](#)
- [NotPresent](#)

2.9. AmoebaFitter

Full Name:	herschel.ia.numeric.toolbox.fit.AmoebaFitter
Type:	Java Class - 
Import:	from herschel.ia.numeric.toolbox.fit import AmoebaFitter

Description

Simulated annealing simplex finding minimum.

AmoebaFitter can be used in two modes: with simulated annealing on or off. The simulated annealing mode is invoked by setting the temperature to some value. By default it is off: temperature at zero.

When the temperature is set to (or left at) zero, AmoebaFitter acts as a simple Nelder-Mead downhill simplex method. With two advantages and one disadvantage. The pro's are that it is reasonably fast and that it does not need partial derivatives. The con is that it will fall into the first local minimum it encounters. No guarantee that this minimum has anything to do with the absolute minimum. This T=0 modus can only be used in mono-modal problems.

In the other modus, when the temperature is set at some value the simplex sometimes takes a uphill step, depending on the temperature at that moment. Steps downhill are always taken. In that way it is possible to climb out of local minima to find better ones. Meanwhile the temperature is steadily lowered, concentrating the search on the by now hopefully found absolute minimum. Of course this takes much more iterations and still there is **no guarantee** that the best value is found. But a better chance. The initial temperature which suggests itself is of the order of the difference between $\#^2$ at a random point and $\#^2$ at the minimum. AmoebaFitter can be used with limits set to one or more of the parameters.

An introduction to the use of fit package can be found [here](#).

At least once have a look at the [background](#) on the organization and structure of the package.

The fit/demo directory contains worked [examples](#) on almost all aspects of of the package.

Example

Example 1: AmoebaFitter (for non-linear models)

```
# assume x and y are Double1d data arrays.
sine = SineModel( - ) # sinusiodal
lolim = Double1d([10,-1,-1])
hilim = Double1d([100,1,1])
sine.setLimits( lolim, hilim - ) # set limits on the model parameters
amfit = AmoebaFitter( x, sine - )
amfit.setTemperature( 100 - ) # set a temperature to escape local
minima
param = amfit.fit( y - )
stdev = amfit.getStandardDeviation() # stdevs on the parameters
chisq = amfit.getChiSquared()
scale = amfit.getScale() # noise scale
yfit = amfit.getResult() # fitted values
yfit = sine( x - ) # fitted values (same as previous)
yband = amfit.monteCarloError() # 1 sigma confidence region
# for diagnostics (or just for fun)
amfit = AmoebaFitter( x, sine - )
amfit.setTemperature( 100 - ) # set a temperature to escape local
minima
amfit.setVerbose( 10 - ) # report every 10th iteration
plotter = IterationPlotter() # from herschel.ia.toolbox.fit
amfit.setPlotter( plotter, 20 - ) # make a plot every 20th iteration
param = amfit.fit( y - )
```


Limitations

1. AmoebaFitter is **not** guaranteed to find the global minimum.
2. The calculation of the evidence is an Gaussian approximation which is only exact for linear models with a fixed scale.

Miscellaneous

In case of problems look at the [trouble shooting](#) section.

2.10. AnnotationToolbox

Full Name:	herschel.ia.gui.image.gui.AnnotationToolbox
Type:	Java Class - 
Import:	from herschel.ia.gui.image.gui import AnnotationToolbox
Category:	Image

Description

A toolbox to draw annotations.

AnnotationToolbox construct a toolbox where the use can select which annotation to display. Using the mouse, the annotation can be put on the image. The jython command to reconstruct the annotation is also shown.

API Summary

Constructors
AnnotationToolbox(Display d) Constructor for the annotation toolbox.
AnnotationToolbox(Display d, boolean useAsComponent) Constructs the annotation toolbox.
Methods
getComponent() Returns the component that contains the annotation toolbox.
close() Closes the annotation toolbox.
saveJythonCode() Saves the jython code to a python script.

API details

Constructors

AnnotationToolbox(Display d) Constructor for the annotation toolbox. Constructs the annotation toolbox. Argument Display d [INPUT, MANDATORY]
AnnotationToolbox(Display d, boolean useAsComponent) Constructs the annotation toolbox. Constructs the annotation toolbox. If useAsComponent is true, the annotation toolbox is component based.

AnnotationToolbox([Display](#) d, boolean useAsComponent)

Arguments

[Display](#) d [INPUT, MANDATORY]

boolean **useAsComponent** [INPUT, MANDATORY]

Methods

getComponent()

Returns the component that contains the annotation toolbox.

Returns the component that contains the annotation toolbox.

close()

Closes the annotation toolbox.


Closes the annotation toolbox.

saveJythonCode()

Saves the jython code to a python script.

Saves the jython code to reconstruct the annotations to a python script.

2.11. AnnularSkyAperturePhotometryExplorer

Full Name:	herschel.ia.gui.image.AnnularSkyAperturePhotometryExplorer
Type:	Java Class - 
Import:	from herschel.ia.gui.image import AnnularSkyAperturePhotometryExplorer

Description

An explorer for AnnularSkyAperturePhotometryProducts.

An explorer for AnnularSkyAperturePhotometryProducts.

API Summary

Constructors
AnnularSkyAperturePhotometryExplorer() The construction of a new AnnularSkyAperturePhotometryExplorer.
AnnularSkyAperturePhotometryExplorer(Object object) The construction of a new AnnularSkyAperturePhotometryExplorer associated with the given object.
Methods
boolean canHandle(Class className) Checks whether this AnnularSkyAperturePhotometryExplorer can handle objects of the given class.
AnnularSkyAperturePhotometryProduct getObject() Returns the object.
Class getVariableType() Returns the expected variable type.
JScrollPane getParameterTable() Returns the parameter table.
JPanel getPlotPanel() Returns the plot panel.
JPanel getSkyIntensityPlotPanel() Returns the plot panel.
JComponent getSkyIntensityPlot() Returns the sky intensity plot.

API details

Constructors

AnnularSkyAperturePhotometryExplorer()
The construction of a new AnnularSkyAperturePhotometryExplorer.

AnnularSkyAperturePhotometryExplorer()

The construction of a new AnnularSkyAperturePhotometryExplorer.

AnnularSkyAperturePhotometryExplorer([Object](#) object)

The construction of a new AnnularSkyAperturePhotometryExplorer associated with the given object.

The construction of a new AnnularSkyAperturePhotometryExplorer associated with the given object.

Argument

[Object](#) **object** [INPUT, MANDATORY]

Methods

boolean canHandle([Class](#) className)

Checks whether this AnnularSkyAperturePhotometryExplorer can handle objects of the given class.

Returns true if this AnnularSkyAperturePhotometryExplorer can handle object of the given class; false otherwise.

Argument

[Class](#) **className** [INPUT, MANDATORY]

Return

boolean

Returns true if this AnnularSkyAperturePhotometry can handle objects of the given class; false otherwise.

[AnnularSkyAperturePhotometryProduct](#) getObject()

Returns the object.

Returns the object for this AnnularSkyAperturePhotometryExplorer.

Return

[AnnularSkyAperturePhotometryProduct](#)

Returns the object for this AnnularSkyAperturePhotometryExplorer.

[Class](#) getVariableType()

Returns the expected variable type.

Returns the expected variable type for this AnnularSkyAperturePhotometryExplorer.

Return

[Class](#)


Returns the expected variable type for this AnnularSkyAperturePhotometryExplorer.

[JScrollPane](#) getParameterTable()

Returns the parameter table.

JScrollPane <code>getParameterTable()</code>
Constructs and returns the parameter table for this <code>AnnularSkyAperturePhotometryExplorer</code> .
Return
JScrollPane
Returns the parameter table for this <code>AnnularSkyAperturePhotometryExplorer</code> .
JPanel <code>getPlotPanel()</code>
Returns the plot panel.
Constructs and returns the plot panel for this <code>AnnularSkyAperturePhotometryExplorer</code> .
Return
JPanel
Returns the plot panel for this <code>AnnularSkyAperturePhotometryExplorer</code> .
JPanel <code>getSkyIntensityPlotPanel()</code>
Returns the plot panel.
Constructs and returns the plot panel for this <code>AnnularSkyAperturePhotometryExplorer</code> .
Return
JPanel
Returns the plot panel for this <code>AnnularSkyAperturePhotometryExplorer</code> .
JComponent <code>getSkyIntensityPlot()</code>
Returns the sky intensity plot.
Constructs and returns the sky intensity plot for this <code>AnnularSkyAperturePhotometryExplorer</code> .
Return
JComponent
Returns the sky intensity plot for this <code>AnnularSkyAperturePhotometryExplorer</code> .

2.12. AnnularSkyAperturePhotometryPanel

Full Name:	herschel.ia.toolbox.image.gui.AnnularSkyAperturePhotometryPanel
Type:	Java Class - 
Import:	from herschel.ia.toolbox.image.gui import AnnularSkyAperturePhotometryPanel

Description

A panel for the AnnularSkyAperturePhotometryTask.

A panel to serve as GUI for the AnnularSkyAperturePhotometryTask.

API Summary

Constructor
AnnularSkyAperturePhotometryPanel() The construction of a new AnnularSkyAperturePhotometryPanel.
Methods
JPanel getAperturesPanel() Returns the panel where to specify the parameters for the apertures.
JComponent getSkyApertureComponent() Returns the component where to specify the inner and outer radius of the annular sky aperture.
drawFigures() Draws the circles bounding the apertures on the image.
moveConfirmedFigures() Allows the user to move the circles bounding the apertures.
clearSkyAperture() Clears the annular sky aperture.

API details

Constructor

<code>AnnularSkyAperturePhotometryPanel()</code>
The construction of a new AnnularSkyAperturePhotometryPanel.
The construction of a new AnnularSkyAperturePhotometryPanel.

Methods

<code>JPanel getAperturesPanel()</code>
Returns the panel where to specify the parameters for the apertures.
Returns the panel where to specify the parameters for the apertures for this AnnularSkyAperturePhotometryPanel.
Return

JPanel `getAperturesPanel()`**JPanel**

Returns the panel where to specify the parameters for the apertures for this `AnnularSkyAperturePhotometryPanel`.

JComponent `getSkyApertureComponent()`

Returns the component where to specify the inner and outer radius of the annular sky aperture.

Returns the component where to specify the inner and outer radius of the annular sky aperture for this `AnnularSkyAperturePhotometryPanel`.

Return**JComponent**

Returns the component where to specify the inner and outer radius of the annular sky aperture for this `AnnularSkyAperturePhotometryPanel`.

drawFigures()

Draws the circles bounding the apertures on the image.

Draws the circles on the image for this `AnnularSkyAperturePhotometryPanel`.

moveConfirmedFigures()

Allows the user to move the circles bounding the apertures.


Allows the user to move the circles bounding the apertures for this `AnnularSkyAperturePhotometryPanel`.

clearSkyAperture()

Clears the annular sky aperture.

Clears the annular sky aperture for this `AnnularSkyAperturePhotometryPanel`.

2.13. AnnularSkyAperturePhotometryProduct

Full Name:	herschel.ia.dataset.image.AnnularSkyAperturePhotometryProduct
Type:	Java Class - 
Import:	from herschel.ia.dataset.image import AnnularSkyAperturePhotometryProduct

Description

A class to deal with the results of aperture photometry with a circular target aperture and an annular sky aperture.

API Summary

Constructor
AnnularSkyAperturePhotometryProduct() The constructor of a new AnnularSkyAperturePhotometryProduct.
Methods
setRadii(double innerPixels, double outerPixels) Sets the inner and outer radius in pixels for this
setRadii(double innerPixels, double innerArcsec, double outerPixels, double outerArcsec) Sets the inner and outer radius for this
double getInnerRadiusPixels() Returns the inner radius for this AnnularSkyAperturePhotometryProduct in pixels.
double getInnerRadiusArcsec() Returns the inner radius for this AnnularSkyAperturePhotometryProduct
double getOuterRadiusPixels() Returns the outer radius for this AnnularSkyAperturePhotometryProduct in pixels.
double getOuterRadiusArcsec() Returns the outer radius for this AnnularSkyAperturePhotometryProduct in arcsec.
TableDataset getSkyIntensityPlot() Returns the sky intensity plot for this
DoubleId getSkyRadius() Returns the sky radius in pixels for the sky intensity
DoubleId getSkyIntensity() Returns the sky intensity for the sky intensity plot for this

Miscellaneous

Aperture specific information is stored in meta data, all other data (results of calculations) are stored in TableDatasets.

API details

Constructor

AnnularSkyAperturePhotometryProduct()

The constructor of a new AnnularSkyAperturePhotometryProduct.

Methods

setRadii(double innerPixels, double outerPixels)

Sets the inner and outer radius in pixels for this

AnnularSkyAperturePhotometryProduct to the given numbers of pixels.

Arguments

double **innerPixels** [INPUT, MANDATORY]

double **outerPixels** [INPUT, MANDATORY]

setRadii(double innerPixels, double innerArcsec, double outerPixels, double outerArcsec)

Sets the inner and outer radius for this

AnnularSkyAperturePhotometryProduct to the given numbers of pixels and arcsec.

Arguments

double **innerPixels** [INPUT, MANDATORY]

double **innerArcsec** [INPUT, MANDATORY]

double **outerPixels** [INPUT, MANDATORY]

double **outerArcsec** [INPUT, MANDATORY]

double getInnerRadiusPixels()

Returns the inner radius for this AnnularSkyAperturePhotometryProduct in pixels.

Return

double

Returns the inner radius for this AnnularSkyAperturePhotometryProduct in pixels.

double getInnerRadiusArcsec()

Returns the inner radius for this AnnularSkyAperturePhotometryProduct

in arcsec.

Return

double

Returns the inner radius for this AnnularSkyAperturePhotometryProduct in arcsec.


double getOuterRadiusPixels()

Returns the outer radius for this AnnularSkyAperturePhotometryProduct in pixels.

Return

double <code>getOuterRadiusPixels()</code>
double Returns the outer radius for this <code>AnnularSkyAperturePhotometryProduct</code> in pixels.
double <code>getOuterRadiusArcsec()</code>
Returns the outer radius for this <code>AnnularSkyAperturePhotometryProduct</code> in arcsec. Return double Returns the outer radius for this <code>AnnularSkyAperturePhotometryProduct</code> in arcsec.
TableDataset <code>getSkyIntensityPlot()</code>
Returns the sky intensity plot for this <code>AnnularSkyAperturePhotometryProduct</code> . Return TableDataset Returns the sky intensity plot for this <code>AnnularSkyAperturePhotometryProduct</code> .
<u>DoubleId</u> <code>getSkyRadius()</code>
Returns the sky radius in pixels for the sky intensity plot for this <code>AnnularSkyAperturePhotometryProduct</code> . Return <u>DoubleId</u> Returns the sky radius in pixels for the sky intensity plot for this <code>AnnularSkyAperturePhotometryProduct</code> .
<u>DoubleId</u> <code>getSkyIntensity()</code>
Returns the sky intensity for the sky intensity plot for this <code>AnnularSkyAperturePhotometryProduct</code> . Return <u>DoubleId</u> Returns the sky intensity for the sky intensity plot for this <code>AnnularSkyAperturePhotometryProduct</code> .

2.14. AnnularSkyAperturePhotometryTask

Full Name:	herschel.ia.toolbox.image.AnnularSkyAperturePhotometryTask
Type:	Java Task - 
Import:	from herschel.ia.toolbox.image import AnnularSkyAperturePhotometryTask

Description

A Task for aperture photometry.

A Task for aperture photometry, using a circular target aperture and an annular sky aperture.

API Summary

Properties
Image image [INPUT, MANDATORY, default=No default value]
Double centerX [INPUT, OPTIONAL, default=Default value : NaN]
Double centerY [INPUT, OPTIONAL, default=Default value : NaN]
String centerRA [INPUT, OPTIONAL, default=Default value : "centerRA"]
String centerDEC [INPUT, OPTIONAL, default=Default value : "centerDec"]
Double radiusPixels [INPUT, OPTIONAL, default=Default value : NaN]
Double radiusArcsec [INPUT, OPTIONAL, default=Default value : NaN]
Integer algorithm [INPUT, MANDATORY, default=Default value : 4]
AnnularSkyAperturePhotometryProduct result [OUTPUT, MANDATORY, default=No default value]
Double innerPixels [INPUT, OPTIONAL, default=Default value : NaN]
Double innerArcsec [INPUT, OPTIONAL, default=Default value : NaN]
Double outerPixels [INPUT, OPTIONAL, default=Default value : NaN]
Double outerArcsec [INPUT, OPTIONAL, default=Default value : NaN]

API details

Properties

Image image [INPUT, MANDATORY, default=No default value]
The image.
Double centerX [INPUT, OPTIONAL, default=Default value : NaN]
The x-pixel-coordinate of the target center.
Double centerY [INPUT, OPTIONAL, default=Default value : NaN]
The y-pixel-coordinate of the target center.
String centerRA [INPUT, OPTIONAL, default=Default value : "centerRA"]
The right ascension of the target center.

String centerDEC [INPUT, OPTIONAL, default=Default value : "centerDec"]

The declination of the target center.

Double radiusPixels [INPUT, OPTIONAL, default=Default value : NaN]

The target radius (i.e. the radius of the circular target aperture) in pixels.

Double radiusArcsec [INPUT, OPTIONAL, default=Default value : NaN]

Integer algorithm [INPUT, MANDATORY, default=Default value : 4]

The sky estimation algorithm.

AnnularSkyAperturePhotometryProduct result [OUTPUT, MANDATORY, default=No default value]

The result.

Double innerPixels [INPUT, OPTIONAL, default=Default value : NaN]

The inner radius of the annular sky aperture in pixels.

Double innerArcsec [INPUT, OPTIONAL, default=Default value : NaN]

The inner radius of the annular sky aperture in arcsec.


Double outerPixels [INPUT, OPTIONAL, default=Default value : NaN]

The outer radius of the annular sky aperture in pixels.

Double outerArcsec [INPUT, OPTIONAL, default=Default value : NaN]

The outer radius of the annular sky aperture in arcsec.

2.15. ANY

Full Name:	herschel.ia.numeric.toolbox.basic.Any
Alias:	ANY
Type:	Java Class - 
Import:	from herschel.ia.numeric.toolbox.basic import Any

Description

Determines whether any element in the array evaluates to true.

The input array must be of integral type (e.g. bytes,integers) and the result is a boolean array.

Example

Example 1: Apply ANY on a Int2d: check if any array item, in the specified dimension, is lower than '3'

```
x=Int2d( [ [1,2], [3,4], [5,6] -])
print ANY(x<3)      #any item => 1 (true)
print ANY(x<3, 0) #dim 0 => [ ANY(Int1d([1,3,5]) < 3) -, ANY(Int1d([2,4,6]) <
3) -] = [true, true]
print ANY(x<3, 1) #dim 1 => [ ANY(Int1d([1,2]) < 3) -, ANY(Int1d([3,4]) < 3),
ANY(Int1d([5,6]) < 3) = [true, false, false]
```

API Summary

Jython Syntax

```
<y>=ALL(<array_expression>[, <dimension>])
```

Properties

[any expression with an integral type **x** \[INPUT, MANDATORY, default=no default value\]](#)

[integer **dimension** \[INPUT, OPTIONAL, default=all the array items\]](#)

[a boolean array **y** \[OUTPUT, MANDATORY, default=no default value\]](#)

API details

Properties

any expression with an integral type **x [INPUT, MANDATORY, default=no default value]**

The input array must be of integral type (e.g. bytes,integers) See example.

integer **dimension [INPUT, OPTIONAL, default=all the array items]**

The dimension where to apply the expression.


a boolean array **y [OUTPUT, MANDATORY, default=no default value]**

The result is a boolean array

See also

- [ALL](#)

2.16. AnyPresent

Full Name:	herschel.ia.numeric.toolbox.basic.AnyPresent
Type:	Java Class - 
Import:	from herschel.ia.numeric.toolbox.basic import AnyPresent

Description

Tests whether any of the bits in specified mask are present in the elements of the input array.

The input array must be of integral type (e.g. bytes,integers) and the result is a boolean array.

Example

Example 1: Apply AnyPresent on a Int1d: test if a binary 'and' comparison, item by item, with '3' (as mask) is not equals to '0'

```
x=Int1d([0,1,2,3])
print AnyPresent(3)(x)
#=> [ (0 & 3) != 0, (1 & 3) != 0, (2 & 3) != 0, (3 & 3) != 0 -]
= [false,true,true,true]
print x.apply(AnyPresent(3)) #Another way for using AnyPresent
```

API Summary

Jython Syntax

```
<y>=AnyPresent(<bitmask>)(<x>)
```

Properties

any integral type **x** [INPUT, MANDATORY, default=no default value]

any number **bitmask** [INPUT, MANDATORY, default=no default value]

a boolean array **y** [OUTPUT, MANDATORY, default=no default value]

API details

Properties

any integral type x [INPUT, MANDATORY, default=no default value]

The input array must be of integral type (e.g. bytes,integers)

any number bitmask [INPUT, MANDATORY, default=no default value]

Accepts any number.


a boolean array y [OUTPUT, MANDATORY, default=no default value]

The result is a boolean array

See also

- [AllPresent](#)
- [NotPresent](#)

2.17. AperturePhotometryExplorer

Full Name:	herschel.ia.gui.image.AperturePhotometryExplorer
Type:	Java Class - 
Import:	from herschel.ia.gui.image import AperturePhotometryExplorer

Description

An explorer for AperturePhotometryProducts.

An explorer for AperturePhotometryProducts.

API Summary

Constructors	
AperturePhotometryExplorer()	The construction of a new AperturePhotometryExplorer.
AperturePhotometryExplorer(Object object)	The construction of a new AperturePhotometryExplorer associated with the given object.
Methods	
String getName()	Returns the name.
String getDescription()	Returns the description.
boolean canHandle(Class className)	Checks whether this AperturePhotometryExplorer can handle objects of the given class.
setObject(Object object)	Sets the object.
AperturePhotometryProduct getObject()	Returns the object.
Class getVariableType()	Returns the expected variable type.
JComponent getComponent()	Returns the component that is responsible for displaying the object.
JPanel getDataPanel()	Returns the data panel.
JScrollPane getParameterTable()	Returns the parameter table.
JScrollPane getResultsTable()	Returns the results table.
JPanel getPlotPanel()	Returns the plot panel.
JPanel getCurveOfGrowthPanel()	Returns the curve of growth panel.

Methods
JComponent getCurveOfGrowth() Returns the curve of growth.
addDataObjectListener(DataObjectListener listener) Adds the given listener.
removeDataObjectListener(DataObjectListener listener) Removes the given listener.

API details

Constructors

AperturePhotometryExplorer() The construction of a new AperturePhotometryExplorer. The construction of a new AperturePhotometryExplorer.
AperturePhotometryExplorer(Object object) The construction of a new AperturePhotometryExplorer associated with the given object. The construction of a new AperturePhotometryExplorer associated with the given object. Argument Object object [INPUT, MANDATORY]

Methods

String getName() Returns the name. Returns the name for this AperturePhotometryExplorer. Return String Returns the name for this AperturePhotometryExplorer.
String getDescription() Returns the description. Returns the description for this AperturePhotometryExplorer. Return String Returns the description for this AperturePhotometryExplorer.
boolean canHandle(Class className) Checks whether this AperturePhotometryExplorer can handle objects of the given class. Returns true if this AperturePhotometryExplorer can handle objects of the given class; false otherwise.

boolean canHandle(Class className)
Argument Class className [INPUT, MANDATORY]
Return boolean Returns true of this AperturePhotometryExplorer can handle objects of the given class; false otherwise.

setObject(Object object)
Sets the object. Sets the object for this AperturePhotometryExplorer to the given object.
Argument Object object [INPUT, MANDATORY]

AperturePhotometryProduct getObject()
Returns the object. Returns the object for this AperturePhotometryExplorer.
Return AperturePhotometryProduct Returns the object for this AperturePhotometryExplorer.

Class getVariableType()
Returns the expected variable type. Returns the expected variable type for this AperturePhotometryExplorer.
Return Class Returns the expected variable type for this AperturePhotometryExplorer.

JComponent getComponent()
Returns the component that is responsible for displaying the object. Returns the component that is responsible for displaying the data object for this AperturePhotometryExplorer.
Return JComponent Returns the component that is responsible for displaying the data object for this AperturePhotometryExplorer.

JPanel getDataPanel()
Returns the data panel.

[JPanel](#) getDataPanel()

Returns the data panel for this AperturePhotometryExplorer.

Return

[JPanel](#)

Returns the data panel for this AperturePhotometryExplorer.

[JScrollPane](#) getParameterTable()

Returns the parameter table.

Constructs and returns the parameter table for this AperturePhotometryExplorer.

Return

[JScrollPane](#)

Returns the parameter table for this AperturePhotometryExplorer.

[JScrollPane](#) getResultsTable()

Returns the results table.

Constructs and returns the results table for this AperturePhotometryExplorer.

Return

[JScrollPane](#)

Returns the results table for this AperturePhotometryExplorer.

[JPanel](#) getPlotPanel()

Returns the plot panel.

Constructs and returns the plot panel for this AperturePhotometryExplorer.

Return

[JPanel](#)

Returns the plot panel for this AperturePhotometryExplorer.

[JPanel](#) getCurveOfGrowthPanel()

Returns the curve of growth panel.

Constructs and returns the curve of growth panel for this AperturePhotometryExplorer.

Return

[JPanel](#)

Returns the curve of growthpanel for this AperturePhotometryExplorer.

[JComponent](#) getCurveOfGrowth()

Returns the curve of growth.

Constructs and returns the curve of growth for this AperturePhotometryExplorer.

JComponent <code>getCurveOfGrowth()</code>
Return JComponent Returns the curve of growth for this AperturePhotometryExplorer.
addDataObjectListener(DataObjectListener listener)
Adds the given listener. Adds the given listener to this AperturePhotometryExplorer to receive data object events from it. Argument DataObjectListener listener [INPUT, MANDATORY]
removeDataObjectListener(DataObjectListener listener)
Removes the given listener. Removes the given listener for this AperturePhotometryExplorer, so that it no longer receives data object events by this explorer. Argument DataObjectListener listener [INPUT, MANDATORY]

2.18. AperturePhotometryPanel

Full Name:	herschel.ia.toolbox.image.gui.AperturePhotometryPanel
Type:	Java Class - 
Import:	from herschel.ia.toolbox.image.gui import AperturePhotometryPanel

Description

A panel for aperture photometry.

A panel to serve as GUI for the AperturePhotometryTask.

API Summary

Constructor	
AperturePhotometryPanel()	The construction of a new AperturePhotometryPanel.
Methods	
JPanel getImagePanel()	Returns a display for the image.
JPanel getTargetCenterPanel()	Returns the panel where to specify the target center.
adjustTargetCenter()	Allows the user to drage the circle indicating the target center.
boolean isInImage(MouseEvent me)	Checks whether the given mouse event occurred inside the image.
boolean isInImage(double pixX, double pixY)	Checks whether the given pixel coordinates lie inside the image.
JPanel getAperturesPanel()	Returns the panel where to specify the parameters for the apertures for this AperturePhotometryPanel.
JPanel getTargetAperturePanel()	Returns the panel where to specify the target radius.
JComponent getSkyApertureComponent()	Returns the panel where to specify the parameters for the annular/rectangular sky aperture.
JPanel getAlgorithmPanel()	Returns the panel where to specify the kind of pixels to use and the sky estimation algorithm.
JPanel getButtonPanel()	Returns the button panel.
Action getRunAction()	Returns the run action for this AperturePhotometryPanel.
Action getResetAction()	Returns the reset action for this AperturePhotometryPanel.

Methods
Modifier addDefaultModifier(String name) Returns the default modifier for the parameter with the given name.
JLabel newLabel(String title, String tooltip) Returns a label with the given title and tooltip.
Border newBorder(String title) Returns a border with the given title.
setVariableSelection(VariableSelection selection) Setting the variable selection.
setSiteEventHandler(SiteEventHandler handler) Setting the site event handler.
setTask(TaskApi task) Associating the task.
Display getDisplay() Returns the display.
Image getImage() Returns the associated image.
TaskApi getTask() Returns the associated task.
Map getSelectionMap() Returns the associated selection map.
Map getModifierMap() Returns the associated modifier map.
SiteEventHandler getHandler() Returns the associated site event handler.
double getTargetRadius() Returns the target radius in pixels.
trigger() Triggers the execution of the associated task.
transferFromModifierToSelectionMap() Transfers the information from the modifier to the selection map.
drawFigures() Draws the figures bounding the apertures on the image.
moveConfirmedFigures() Allows to move/resize the figures bounding the apertures.
clearSkyAperture() Clears the sky aperture.

API details

Constructor

AperturePhotometryPanel()
The construction of a new AperturePhotometryPanel.

AperturePhotometryPanel ()

The construction of a new AperturePhotometryPanel.

Methods

JPanel getImagePanel ()

Returns a display for the image.

Returns a panel that displays the image for this PhotometryPanel.

Return

[JPanel](#)

Returns a panel that displays the image for this AperturePhotometryPanel.

JPanel getTargetCenterPanel ()

Returns the panel where to specify the target center.

Returns the panel where to specify the target center for this AperturePhotometryPanel.

Return

[JPanel](#)

Returns the panel where to specify the target center for this AperturePhotometryPanel.

adjustTargetCenter ()

Allows the user to drage the circle indicating the target center.

Allows the user to drag the circle indicating the target center for this AperturePhotometryPanel.

boolean isInImage (MouseEvent me)

Checks whether the given mouse event occurred inside the image.

Returns true if the given mouse event occurred inside the image for this AperturePhotometryPanel; false otherwise.

Argument

[MouseEvent](#) **me** [INPUT, MANDATORY]

Return

boolean

Returns true if the given mouse event occurred inside the image for this AperturePhotometryPanel; false otherwise.

boolean isInImage (double pixX, double pixY)

Checks whether the given pixel coordinates lie inside the image.

Returns true if the given pixel coordinates lie inside the image for this AperturePhotometryPanel; false otherwise.

Arguments

boolean `isInImage(double pixX, double pixY)`

double `pixX` [INPUT, MANDATORY]

double `pixY` [INPUT, MANDATORY]

Return

boolean

Returns true if the given pixel coordinates lie inside the image for this AperturePhotometryPanel; false otherwise.

[JPanel](#) `getAperturesPanel()`

Returns the panel where to specify the parameters for the apertures for this AperturePhotometryPanel.

Returns the panel where to specify the parameters for the apertures for this AperturePhotometryPanel.

Return

[JPanel](#)

Returns the panel where to specify the parameters for the apertures for this AperturePhotometryPanel.

[JPanel](#) `getTargetAperturePanel()`

Returns the panel where to specify the target radius.

Returns the panel where to specify the target radius for this AperturePhotometryPanel.

Return

[JPanel](#)

Returns the panel where to specify the target radius for this AperturePhotometryPanel.

[JComponent](#) `getSkyApertureComponent()`

Returns the panel where to specify the parameters for the annular/rectangular sky aperture.

Returns the panel where to specify the parameters for the annular/rectangular sky aperture for this AperturePhotometryPanel.

Return

[JComponent](#)

Returns the panel where to specify the parameter for the annular/rectangular sky aperture for this AperturePhotometryPanel.

[JPanel](#) `getAlgorithmPanel()`

Returns the panel where to specify the kind of pixels to use and the sky estimation algorithm.

Returns the panel where to specify the kind of pixels to use and the sky estimation algorithm for this AperturePhotometryPanel.

Return

[JPanel](#)

JPanel `getAlgorithmPanel()`

Returns the panel where to specify the kind of pixels to use and the sky estimation algorithm for this AperturePhotometryPanel.

JPanel `getButtonPanel()`

Returns the button panel.

Returns the button panel for this AperturePhotometryPanel

Return

[JPanel](#)

Returns the button panel for this AperturePhotometryPanel.

Action `getRunAction()`

Returns the run action for this AperturePhotometryPanel.

Returns the run action for this AperturePhotometryPanel.

Return

[Action](#)

Returns the run action for this AperturePhotometryPanel.

Action `getResetAction()`

Returns the reset action for this AperturePhotometryPanel.

Returns the reset action for this AperturePhotometryPanel.

Return

[Action](#)

Returns the reset action for this AperturePhotometryPanel.

Modifier `addDefaultModifier(String name)`

Returns the default modifier for the parameter with the given name.

Returns the default modifier for the parameter with the given name for this AperturePhotometryPanel.

Argument

[String](#) name [INPUT, MANDATORY]

Return

Modifier

Returns the default modifier for the parameter with the given name.

JLabel `newLabel(String title, String tooltip)`

Returns a label with the given title and tooltip.

Returns a label with the given title and tooltip.

[JLabel](#) newLabel([String](#) title, [String](#) tooltip)

Arguments

[String](#) title [INPUT, MANDATORY]

[String](#) tooltip [INPUT, MANDATORY]

Return

[JLabel](#)

Returns a label with the given title and tooltip.

[Border](#) newBorder([String](#) title)

Returns a border with the given title.

Returns a label with the given title and tooltip.

Argument

[String](#) title [INPUT, MANDATORY]

Return

[Border](#)

Returns a border with the given title.

setVariableSelection([VariableSelection](#) selection)

Setting the variable selection.

Sets the variable selection for this AperturePhotometryPanel to the given variable selection.

Argument

[VariableSelection](#) selection [INPUT, MANDATORY]

setSiteEventHandler([SiteEventHandler](#) handler)

Setting the site event handler.

Sets the site event handler for this AperturePhotometryPanel to the given site event handler.

Argument

[SiteEventHandler](#) handler [INPUT, MANDATORY]

setTask([TaskApi](#) task)

Associating the task.

Associates the given task with this AperturePhotometryPanel.

Argument

[TaskApi](#) task [INPUT, MANDATORY]

[Display](#) getDisplay()

Returns the display.

Returns the display for this AperturePhotometryPanel.

Return

[Display](#)

Display `getDisplay()`

Returns the display for this AperturePhotometryPanel.

Image `getImage()`

Returns the associated image.

Returns the image associated with this AperturePhotometryPanel.

Return**Image**

Returns the image associated with this AperturePhotometryPanel.

TaskApi `getTask()`

Returns the associated task.

Returns the task associated with this AperturePhotometryPanel.

Return**TaskApi**

Returns the task associated with this AperturePhotometryPanel.

Map `getSelectionMap()`

Returns the associated selection map.

Returns the selection map associated with this AperturePhotometryPanel.

Return**Map**

Returns the selection map associated with this AperturePhotometryPanel.

Map `getModifierMap()`

Returns the associated modifier map.

Returns the modifier map associated with this AperturePhotometryPanel.

Return**Map**

Returns the modifier map associated with this AperturePhotometryPanel.

SiteEventHandler `getHandler()`

Returns the associated site event handler.

Returns the site event handler associated with this AperturePhotometryPanel.

Return**SiteEventHandler**

Returns the site event handler associated with this AperturePhotometryPanel.

double getTargetRadius()
Returns the target radius in pixels.
Returns the target radius for this AperturePhotometryPanel in pixels.
Return
double
Returns the target radius for this AperturePhotometryPanel in pixels.

trigger()
Triggers the execution of the associated task.
Triggers the execution of the task associated with this AperturePhotometryPanel.


transferFromModifierToSelectionMap()
Transfers the information from the modifier to the selection map.
Transfers the information for this AperturePhotometryPanel from the modifier to the selection map.

drawFigures()
Draws the figures bounding the apertures on the image.
Draws the figures bounding the apertures for this AperturePhotometryPanel on the image.

moveConfirmedFigures()
Allows to move/resize the figures bounding the apertures.
Allows to move/resize the figures bounding the apertures for this AperturePhotometryPanel.

clearSkyAperture()
Clears the sky aperture.
Clears the sky aperture for this AperturePhotometryPanel.

2.19. AperturePhotometryProduct

Full Name:	herschel.ia.dataset.image.AperturePhotometryProduct
Type:	Java Class - 
Import:	from herschel.ia.dataset.image import AperturePhotometryProduct

Description

A class to deal with the results of aperture photometry.

API Summary

Constructor	
AperturePhotometryProduct()	The constructor of a new AperturePhotometryProduct.
Methods	
setTargetCenter(double centerX, double centerY)	Sets the target center for this AperturePhotometryProduct to
setTargetCenter(double centerX, double centerY, String centerRA, String centerDec)	Sets the target center for this AperturePhotometryproduct
setTargetRadius(double pixels)	Sets the target radius for this AperturePhotometryProduct
setTargetRadius(double pixels, double arcsec)	Sets the target radius for this AperturePhotometryProduct
setPixels(boolean fractional)	Sets the type of pixels (entire / fractional) used for this
setUnit(Unit unit)	Sets the unit for this AperturePhotometryProduct to the given unit.
setResultsTable(Double2d resultsTable)	Sets the results table for this AperturePhotometryProduct to the
setCurveOfGrowth(Double1d growthRadius, Double1d growthFlux)	Sets the curve of growth for this AperturePhotometryProduct for the given
Double1d getTargetCenterPixelCoordinates()	Returns the target center for this AperturePhotometryProduct in
String1d getTargetCenterSkyCoordinates()	Returns the target center for this AperturePhotometryProduct in
double getTargetRadiusPixels()	Returns the target radius for this AperturePhotometryProduct in pixels.
double getTargetRadiusArcsec()	Returns the target radius for this AperturePhotometryProduct in arcsec.
String getPixels()	Returns the String representation of the type of pixels (fractional / entire) used

Methods
<p>String getUnit() Returns the unit for this AperturePhotometryProduct.</p>
<p>TableDataset getTable() Returns the results table for this AperturePhotometryProduct.</p>
<p>Double2d getDouble2dTable() Returns the results table for this AperturePhotometryProduct as a Double2d.</p>
<p>Double1d getTotal() Returns the total flux for the target, the sky and the target from</p>
<p>Double1d getNbOfPixels() Returns the number of pixels for the target, the sky and the target from</p>
<p>Double1d getIntensityPerPixel() Returns the intensity per pixel for the target, the sky and the</p>
<p>Double1d getError() Returns the error for the target, the sky and the target from which the</p>
<p>double getTargetPlusSkyTotal() Returns the total flux for the target (sky included) for this</p>
<p>double getNbOfTargetPlusSkyPixels() Returns the number of pixels for the target (sky included) for this</p>
<p>double getIntensityPerTargetPlusSkyPixel() Returns the intensity per pixel for the target (sky included) for this</p>
<p>double getTargetPlusSkyError() Returns the error for the target (sky included) for this AperturePhotometryProduct.</p>
<p>double getIntensityPerSkyPixel() Returns the intensity per pixel for the sky for this AperturePhotometryProduct.</p>
<p>double getTargetTotal() Returns the total flux for the target (sky subtracted) for this</p>
<p>double getNbOfTargetPixels() Returns the number of pixels for the target (sky subtracted) for this</p>
<p>double getIntensityPerTargetPixel() Returns the intensity per pixel for the target (sky subtracted) for this</p>
<p>double getTargetError() Returns the error for the target (sky subtracted) for this</p>
<p>TableDataset getCurveOfGrowth() Returns the curve of growth for this AperturePhotometryProduct.</p>
<p>Double1d getGrowthRadius() Returns the growth radius for this AperturePhotometryProduct.</p>
<p>Double1d getGrowthFlux() Returns the growth flux for this AperturePhotometryProduct,</p>

Miscellaneous

Aperture specific information is stored in meta data, all other data (results of calculations) are stored in TableDatasets.

API details

Constructor

AperturePhotometryProduct()

The constructor of a new AperturePhotometryProduct.

Methods

setTargetCenter(double centerX, double centerY)

Sets the target center for this AperturePhotometryProduct to

the given pixel coordinates.

Arguments

double **centerX** [INPUT, MANDATORY]

double **centerY** [INPUT, MANDATORY]

setTargetCenter(double centerX, double centerY, [String](#) centerRA, [String](#) centerDec)

Sets the target center for this AperturePhotometryproduct

to the given pixel and sky coordinates.

Arguments

double **centerX** [INPUT, MANDATORY]

double **centerY** [INPUT, MANDATORY]

[String](#) **centerRA** [INPUT, MANDATORY]

[String](#) **centerDec** [INPUT, MANDATORY]

setTargetRadius(double pixels)

Sets the target radius for this AperturePhotometryProduct

to the given number of pixels.

Argument

double **pixels** [INPUT, MANDATORY]

setTargetRadius(double pixels, double arcsec)

Sets the target radius for this AperturePhotometryProduct

to the given number of pixels and arcsec.

Arguments

double **pixels** [INPUT, MANDATORY]

double **arcsec** [INPUT, MANDATORY]

setPixels(boolean fractional)

Sets the type of pixels (entire / fractional) used for this

AperturePhotometryProduct.

setPixels(boolean fractional)
Argument boolean fractional [INPUT, MANDATORY]
setUnit(Unit unit)
Sets the unit for this AperturePhotometryProduct to the given unit. Argument Unit unit [INPUT, MANDATORY]
setResultsTable(Double2d resultsTable)
Sets the results table for this AperturePhotometryProduct to the given table. Argument Double2d resultsTable [INPUT, MANDATORY]
setCurveOfGrowth(Double1d growthRadius, Double1d growthFlux)
Sets the curve of growth for this AperturePhotometryProduct for the given growth radius and growth flux. Arguments Double1d growthRadius [INPUT, MANDATORY] Double1d growthFlux [INPUT, MANDATORY]
Double1d getTargetCenterPixelCoordinates()
Returns the target center for this AperturePhotometryProduct in pixel coordinates. Return Double1d Returns the target center for this AperturePhotometryProduct in pixel coordinates.
String1d getTargetCenterSkyCoordinates()
Returns the target center for this AperturePhotometryProduct in sky coordinates. Return String1d Returns the target center for this AperturePhotometryProduct in sky coordinates.
double getTargetRadiusPixels()
Returns the target radius for this AperturePhotometryProduct in pixels. Return double

double `getTargetRadiusPixels()`

Returns the target radius for this AperturePhotometryProduct in pixels.

double `getTargetRadiusArcsec()`

Returns the target radius for this AperturePhotometryProduct in arcsec.

Return**double**

Returns the target radius for this AperturePhotometryProduct in arcsec.

String `getPixels()`

Returns the String representation of the type of pixels (fractional / entire) used for this AperturePhotometryProduct.

Return**String**

Returns the String representation of the type of pixels (fractional / entire) used for this AperturePhotometryProduct.

String `getUnit()`

Returns the unit for this AperturePhotometryProduct.

Return**String**

Returns the unit for this AperturePhotometryProduct.

TableDataset `getTable()`

Returns the results table for this AperturePhotometryProduct.

Return**TableDataset**

Returns the results table for this AperturePhotometryProduct.

Double2d `getDouble2dTable()`

Returns the results table for this AperturePhotometryProduct as a Double2d.

Return**Double2d**

Returns the results table for this AperturePhotometryProduct as a Double2d.

Double1d `getTotal()`

Returns the total flux for the target, the sky and the target from which the sky has been subtracted for this AperturePhotometryProduct.

Return

DoubleId getTotal()

DoubleId

Returns the total flux for the target, the sky and the target from which the sky has been subtracted for this AperturePhotometryProduct.

DoubleId getNbOfPixels()

Returns the number of pixels for the target, the sky and the target from which the sky has been subtracted.

Return

DoubleId

Returns the number of pixels for the target, the sky and the target from which the sky has been subtracted.

DoubleId getIntensityPerPixel()

Returns the intensity per pixel for the target, the sky and the target from which the sky has been subtracted for this AperturePhotometryProduct.

Return

DoubleId

Returns the intensity per pixel for the target, the sky and the target from which the sky has been subtracted for this AperturePhotometryProduct.

DoubleId getError()

Returns the error for the target, the sky and the target from which the sky has been subtracted for this AperturePhotometryProduct.

Return

DoubleId

Returns the error for the target, the sky and the target from which the sky has been subtracted for this AperturePhotometryProduct.

double getTargetPlusSkyTotal()

Returns the total flux for the target (sky included) for this AperturePhotometryProduct.

Return

double

Returns the total flux for the target (sky included) for this AperturePhotometryProduct.

double getNbOfTargetPlusSkyPixels()

Returns the number of pixels for the target (sky included) for this AperturePhotometryProduct.

```
double getNbOfTargetPlusSkyPixels()
```

Return**double**

Returns the number of pixels for the target (sky included) for this AperturePhotometryProduct.

```
double getIntensityPerTargetPlusSkyPixel()
```

Returns the intensity per pixel for the target (sky included) for this AperturePhotometryProduct.

Return**double**

Returns the intensity per pixel for the target (sky included) for this AperturePhotometryProduct.

```
double getTargetPlusSkyError()
```

Returns the error for the target (sky included) for this AperturePhotometryProduct.

Return**double**

Returns the error for the target (sky included) for this AperturePhotometryProduct.

```
double getIntensityPerSkyPixel()
```

Returns the intensity per pixel for the sky for this AperturePhotometryProduct.

Return**double**

Returns the intensity per pixel for the sky for this AperturePhotometryProduct.

```
double getTargetTotal()
```

Returns the total flux for the target (sky subtracted) for this AperturePhotometryProduct.

Return**double**

Returns the total flux for the target (sky subtracted) for this AperturePhotometryProduct.

```
double getNbOfTargetPixels()
```


Returns the number of pixels for the target (sky subtracted) for this AperturePhotometryProduct.

Return**double**

Returns the number of pixels for the target (sky subtracted) for this AperturePhotometryProduct.

double <code>getIntensityPerTargetPixel()</code>
Returns the intensity per pixel for the target (sky subtracted) for this AperturePhotometryProduct.
Return
double
Returns the intensity per pixel for the target (sky subtracted) for this AperturePhotometryProduct.
double <code>getTargetError()</code>
Returns the error for the target (sky subtracted) for this AperturePhotometryProduct.
Return
double
Returns the error for the target (sky subtracted) for this AperturePhotometryProduct.
TableDataset <code>getCurveOfGrowth()</code>
Returns the curve of growth for this AperturePhotometryProduct.
Return
TableDataset
Returns the curve of growth for this AperturePhotometryProduct.
Double1d <code>getGrowthRadius()</code>
Returns the growth radius for this AperturePhotometryProduct.
Return
Double1d
Returns the growth radius for this AperturePhotometryProduct.
Double1d <code>getGrowthFlux()</code>
Returns the growth flux for this AperturePhotometryProduct,
Return
Double1d
Returns the growth flux for this AperturePhotometryProduct.

2.20. AperturePhotometryTask

Full Name:	herschel.ia.toolbox.image.AperturePhotometryTask
Type:	Java Task - 
Import:	from herschel.ia.toolbox.image import AperturePhotometryTask

Description

An abstract Task for aperture photometry.

Three Tasks implement this interface. They all use a circular target aperture. Two use a sky aperture (annular/rectangular) to estimate the sky with one of the five sky estimation algorithms. The third one uses a fixed value for the sky.

API Summary

Properties
Image image [INPUT, MANDATORY, default=No default value]
Double centerX [INPUT, OPTIONAL, default=Default value : NaN]
Double centerY [INPUT, OPTIONAL, default=Default value : NaN]
String centerRA [INPUT, OPTIONAL, default=Default value : "center-RA"]
String centerDec [INPUT, OPTIONAL, default=Default value : "centerDec"]
Double radiusPixels [INPUT, OPTIONAL, default=Default value : NaN]
Integer radiusArcsec [INPUT, OPTIONAL, default=Default value : 4]
boolean fractional [INPUT, OPTIONAL, default=Default value : true]
boolean centroid [INPUT, OPTIONAL, default=Default value : false]
AperturePhotometryProduct result [OUTPUT, MANDATORY, default=No default value]

Miscellaneous

In the current version two subclasses extend this class : - an abstract class for aperture photometry for which the sky is estimated through a sky aperture (annular/rectangular) and a sky estimation algorithm - a class for aperture photometry for which to sky has a fixed value, given by the user In both cases, a circular target aperture is used.


API details

Properties

Image image [INPUT, MANDATORY, default=No default value]
The image.
Double centerX [INPUT, OPTIONAL, default=Default value : NaN]
The x-pixel-coordinate of the target center.

Double centerY [INPUT, OPTIONAL, default=Default value : NaN]
The y-pixel-coordinate of the target center.
String centerRA [INPUT, OPTIONAL, default=Default value : "centerRA"]
The right ascension of the target center.
String centerDec [INPUT, OPTIONAL, default=Default value : "centerDec"]
The declination of the target center.
Double radiusPixels [INPUT, OPTIONAL, default=Default value : NaN]
The radius of the circular target aperture in pixels.
Integer radiusArcsec [INPUT, OPTIONAL, default=Default value : 4]
The radius of the circular target aperture in arcsec.
boolean fractional [INPUT, OPTIONAL, default=Default value : true]
The type of pixels.
boolean centroid [INPUT, OPTIONAL, default=Default value : false]
Indicates whether to centroid on the target center.
AperturePhotometryProduct result [OUTPUT, MANDATORY, default=No default value]
The result.

2.21. ARCCOS

Full Name:	herschel.ia.numeric.toolbox.basic.ArcCos
Alias:	ARCCOS
Type:	Java Class - 
Import:	from herschel.ia.numeric.toolbox.basic import ArcCos

Description

Computes the inverse cosine of a number or array, and may be of any type.

Gives the inverse cosine of a number or array. The input must be in the range $[-1,1]$, and may be of any type. Returns a float array for float input array, complex array for input complex array, and double array for any other numeric array type. The result is in radians.

Example

Example 1: Apply ARCCOS on a Float1d

```
x=Float1d([0,0.5])
print ARCCOS(x) # [1.5707964,1.0471976]
```

API Summary

Jython Syntax

```
<y>=ARCCOS(<x>)
```

Properties

[any type x \[INPUT, MANDATORY, default=no default value\]](#)

[radians y \[OUTPUT, MANDATORY, default=no default value\]](#)

API details

Properties

any type x [INPUT, MANDATORY, default=no default value]

must be in the range $[-1,1]$, and may be of any type.


radians y [OUTPUT, MANDATORY, default=no default value]

Returns a float array for float input array, complex array for input complex array, and double array for any other numeric array type. The result is in radians.

See also

- [COS](#)
- [COSH](#)

2.22. ARCSIN

Full Name:	herschel.ia.numeric.toolbox.basic.ArcSin
Alias:	ARCSIN
Type:	Java Class - 
Import:	from herschel.ia.numeric.toolbox.basic import ArcSin

Description

Computes the inverse sine of a number or array, and may be of any type.

Gives the inverse sine of a number or array. The input must be in the range [-1,1], and may be of any type. Returns a float array for float input array, complex array for input complex array, and double array for any other numeric array type. The result is in radians.

Example

Example 1: Apply ARCSIN on a Float1d

```
x=Float1d([0,0.5])
print ARCSIN(x) # [0.0,0.5235988]
```

API Summary

Jython Syntax

```
<y>=ARCSIN(<x>)
```

Properties

[any type x \[INPUT, MANDATORY, default=no default value\]](#)

[radians y \[OUTPUT, MANDATORY, default=no default value\]](#)

API details

Properties

any type x [INPUT, MANDATORY, default=no default value]

must be in the range [-1,1], and may be of any type.


radians y [OUTPUT, MANDATORY, default=no default value]

Returns a float array for float input array, complex array for input complex array, and double array for any other numeric array type. The result is in radians.

See also

- [SIN](#)
- [SINH](#)

2.23. ARCTAN

Full Name:	herschel.ia.numeric.toolbox.basic.ArcTan
Alias:	ARCTAN
Type:	Java Class - 
Import:	from herschel.ia.numeric.toolbox.basic import ArcTan

Description

Computes the inverse tangent of a number or array, and may be of any type.

Gives the inverse tangent of a number or array. The input must be in the range $[-1,1]$, and may be of any type. Returns a float array for float input array, complex array for input complex array, and double array for any other numeric array type. The result is in radians.

Example

Example 1: Apply ARCTAN on a Float1d

```
x=Float1d([0,0.5])
print ARCTAN(x) # [0.0,0.4636476]
```

API Summary

Jython Syntax

```
<y>=ARCTAN(<x>)
```

Properties

[any type x \[INPUT, MANDATORY, default=no default value\]](#)

[radians y \[OUTPUT, MANDATORY, default=no default value\]](#)

API details

Properties

any type x [INPUT, MANDATORY, default=no default value]

must be in the range $[-1,1]$, and may be of any type.


radians y [OUTPUT, MANDATORY, default=no default value]

Returns a float array for float input array, complex array for input complex array, and double array for any other numeric array type. The result is in radians.

See also

- [TAN](#)
- [TANH](#)

2.24. ArctanModel

Full Name:	herschel.ia.numeric.toolbox.fit.ArctanModel
Type:	Java Class - 
Import:	from herschel.ia.numeric.toolbox.fit import ArctanModel

Description

ArcTangus Model.

$$f(x;p) = p_0 * \arctan(p_1 * (x - p_2))$$

where p_0 = amplitude, p_1 = slope and p_2 = offset. As always x = input.

The parameters are initialized at $\{2/\pi, 1.0, 0.0\}$. It is a non-linear Model.


See [example](#)

Example

Example 1: ArctanModel

```
arct = ArctanModel( -)          # arctangus
print arct.getNumberOfParameters() # 3
```

2.25. ArithmeticSpectrumTask

Full Name:	herschel.ia.toolbox.spectrum.ArithmeticSpectrumTask
Type:	Java Task - 
Import:	from herschel.ia.toolbox.spectrum import ArithmeticSpectrumTask

Description

Base task for applying a scalar operation to the flux data in a spectrum container or processing two spectrum containers on a spectrum-by-spectrum basis.

For the scalar mode, 'ds' and 'param' need to be specified - for the pair-wise mode, 'ds1' and 'ds2' need to be set. In the pair-wise mode, the task iterates through the two containers and computes the pair-operation. In case the size of the two containers is different, the iteration stops at the minimum size. Hence, the remaining spectra in the larger container are ignored. In case one of the datasets only contains a single spectrum, the task always refers to this single spectrum while iterating over all the spectra in the other dataset.

Different selection schemes are available for selecting the point spectra or the segments to be processed. The most simple scheme is to specify lists of point spectrum indices (`selection=[1, 3, 4, 2]`). In this case, the operation (scalar- or pair-operation) is just taken over the point spectra with corresponding indices. Instead of specifying a list of indices (which requires the knowledge of these) other recipes to specify a selection are available. Typical recipes test attribute values to match certain values or to be included in a given interval. These recipes we refer to as selection models. The different options to specify selections can be combined. Here, an AND logic is adopted.

Similarly, you can specify what segments to consider. In the most simple case, you can just specify the indices of segments to be considered (for the scalar operation `segments=[1, 3, 4]` and for the pair operation `segments1=[1, 3, 4]`, `segments2=[3, 6, 5]`). In more advanced situations you can use a `SegmentSelection` object. Note that the pairs of segments to be processed need to be consistent - otherwise the processing will fail.

You can specify whether the original datasets should be overwritten with the 'overwrite' flag. By default no data is overwritten. In the pair-wise mode, it is not always possible to overwrite the data - in particular if non-trivial segment selections are specified.

For the pair-wise mode, there are further options that can be selected by the 'variant' (e.g. whether to use flags and weights).

API Summary

Properties
SpectrumContainer ds [INPUT, OPTIONAL, default=no default value.]
Double param [INPUT, OPTIONAL, default=no default value.]
Object selection [INPUT, OPTIONAL, default=None.]
PyDictionary Map> selection_lookup [INPUT, OPTIONAL, default=no default value.]
PyList selection_index [INPUT, OPTIONAL, default=No default value.]
Boolean overwrite [INPUT, OPTIONAL, default=False.]
SpectrumContainer ds1 [INPUT, OPTIONAL, default=no default value.]
SpectrumContainer ds2 [INPUT, OPTIONAL, default=no default value.]

Properties
Object <code>segments</code> [INPUT, OPTIONAL, default=no default value.]
Object <code>segments1</code> [INPUT, OPTIONAL, default=no default value.]
Object <code>segments2</code> [INPUT, OPTIONAL, default=no default value.]
String <code>variant</code> [INPUT, OPTIONAL, default=no default value.]
SpectrumContainer <code>result</code> [OUTPUT, OPTIONAL, default=no default value]

API details

Properties

SpectrumContainer <code>ds</code> [INPUT, OPTIONAL, default=no default value.]
Input container to be processed by the task in case the task is used as a scalar operation.
Double <code>param</code> [INPUT, OPTIONAL, default=no default value.]
The scalar parameter to be considered when the task is used as scalar operation.
Object <code>selection</code> [INPUT, OPTIONAL, default=None.]
Specify what point spectra the operation should be applied to. Different ways to specify these selections are possible: <ul style="list-style-type: none"> Specify a list of indices (in jython) of the point spectra for which the add should be applied. Use a dictionary (in jython) to specify a selection model (lookup for attributes, filter attributes to be included in ranges or intervals). Pass a string representation of a jython list or a jython dictionary. The string will be parsed and the processing delegated to the two cases above. Pass any java instance that implements the SelectionModel interface (for the advanced user).
PyDictionary Map<T> <code>selection_lookup</code> [INPUT, OPTIONAL, default=no default value.]
Specify a PyDictionary with keys given by the attribute name to look up and a list of admissible values. Behind the scenes, a DiscreteValueSelectionModel is instantiated.
PyList <code>selection_index</code> [INPUT, OPTIONAL, default=No default value.]
Specify a PyList with the indices of the point spectra to be considered.
Boolean <code>overwrite</code> [INPUT, OPTIONAL, default=False.]
Specify whether the input data container can be reused - the values found therein is overwritten.
SpectrumContainer <code>ds1</code> [INPUT, OPTIONAL, default=no default value.]
First input container for pair-wise operations.

SpectrumContainer ds2 [INPUT, OPTIONAL, default=no default value.]

Second input container for pair-wise operations.
--

Object segments [INPUT, OPTIONAL, default=no default value.]

Specify what segments the operation should be applied to. There are two options available:
--

- Either pass an instance of a SegmentSelection which gives the information on what segments for each point spectrum included in the container.
- Specify a PyList of segment indices.

Object segments1 [INPUT, OPTIONAL, default=no default value.]
--

Specify the segment selection to be associated with 'ds1'.
--

Object segments2 [INPUT, OPTIONAL, default=no default value.]
--

Specify the segment selection to be associated with 'ds2'.
--

String variant [INPUT, OPTIONAL, default=no default value.]
--

Specify the variant of processing mode you would like to run (including flags / weights / ...). HCSS defaults include: "flux" / "flux-wave" / "flux-wave-weight" / "flux-flag-wave" / "flux-weight-flag-wave"
--


SpectrumContainer result [OUTPUT, OPTIONAL, default=no default value]
--

Result object containing the results of the operation applied.
--

History

- 2009-05-20 - meli: Initial.

2.26. ArrayAssistant

Full Name:	herschel.ia.numeric.toolbox.fit.ArrayAssistant
Type:	Java Class - 
Import:	from herschel.ia.numeric.toolbox.fit import ArrayAssistant

Description

ArrayAssistant contains 2 methods to assist with more dimensional fitting.

1. `getIndices`:

Generates indices for data arrays of any dimension.

To be used as input in the Fitter classes.

2. `resizeData`:

Resizes the data arrays into a 1-dimensional array.


To be used as data in the Fitter.

Example

Example 1: ArrayAssistant

```
# Suppose y is a 2-dimensional map of something
aass = ArrayAssistant()
input = aass.getIndices( y -)
fitter = Fitter( input, some2dModel -)
pars = fitter.fit( aass.resizeData( y -) -)
yfit = some2dModel( input -)          # Double1d
yfit2d = aass.resizeData( yfit, y -)  # Double2d, same size as y
```

2.27. AsciiTableReader

Full Name:	herschel.ia.toolbox.util.AsciiTableReaderTask
Alias:	asciiTableReader
Type:	Java Task - 
Import:	from herschel.ia.toolbox.util import AsciiTableReaderTask
Category:	utility task

Description

AsciiTableReaderTask

Task for reading ascii files.

Examples

Example 1: Read a raw space-separated ascii table

```
# test_space.dat contents
## start
#1 2 3
#
#2 3 4
#5 6 7
#
#
## end
from herschel.ia.io.ascii import AsciiParser
atrt = AsciiTableReaderTask()
tds = atrt(file="test_space.dat", \
           parserDelim=" ", \
           #template=TableTemplate(3,names=["a","b","c"]), \
           parserIgnore= -'^\s*$|^\s*#', \
           parserIgnoreWarn= 1, \
           parserGuess=AsciiParser.GUESS_TRY)
# tds will hold a TableDataset with the data, blank lines will be ignored,
and warnings will be issued for ignored lines
```

Example 2: Read a file

```
atrt = AsciiTableReaderTask()
tableDataset = atrt(file="file.ascii")
print tableDataset
```

Example 3: Read a file without HCSS header (first row does not contain names), delimiter is tab

```
atrt = AsciiTableReaderTask()
tableDataset = atrt(file="file.ascii"; parserDelim='\t',
                    parserGuess=AsciiParser.GUESS_TRY)
```

API Summary

Jython Syntax

```
tableDataset=asciiTableReader(<file>, [, <configFile>, <config-
FileOutput>, <parser>, \
<parserIgnorer>, <parserIgnoreWarn>, <parserSkip>, <parserTrim>,
<parserGuess>, \
```


Jython Syntax
<parserDelim>, <parseNames>, <template>])
see examples

Properties
String file [INPUT, MANDATORY, default=null]
String table [OUTPUT, MANDATORY, default=null]
String configFile [INPUT, OPTIONAL, default=null]
String configFileOutput [INPUT, OPTIONAL, default=null]
AsciiParser parser [INPUT, OPTIONAL, default=default AsciiTableTool parser]
String parserIgnore [INPUT, OPTIONAL, default=default AsciiParser ignore value.]
Boolean parserIgnoreWarn [INPUT, OPTIONAL, default=default AsciiTableTool warnWhenIgnore value (false).]
Integer parserSkip [INPUT, OPTIONAL, default=default AsciiParser skipping rows value.]
Boolean parserTrim [INPUT, OPTIONAL, default=default AsciiParser trim rows value.]
Integer parserGuess [INPUT, OPTIONAL, default=GUESS_NONE Guess behavior.]
String parserDelim [INPUT, OPTIONAL, default=comma separator.]
Boolean parseNames [INPUT, OPTIONAL, default=default AsciiParser parseNames value.]
TableTemplate template [INPUT, OPTIONAL, default=extracted from the first file rows.]

API details

Properties

String file [INPUT, MANDATORY, default=null]
Input file.
String table [OUTPUT, MANDATORY, default=null]
TableDataset object loaded.
String configFile [INPUT, OPTIONAL, default=null]
Configuration file where the formatter (AsciiFormatter), parser (AsciiParser) and table template (TableTemplate) must be specified. When configFile parameter is specified, any parameter related to parser or to table template are not allowed.
String configFileOutput [INPUT, OPTIONAL, default=null]
If a file is specified, an output configuration file will be created.
AsciiParser parser [INPUT, OPTIONAL, default=default AsciiTableTool parser]
AsciiParser object

String parserIgnore [INPUT, OPTIONAL, default=default AsciiParser ignore value.]

String expression to ignore when parsing a file. AsciiParser.setIgnore(expression) is called.

Boolean parserIgnoreWarn [INPUT, OPTIONAL, default=default AsciiTableTool warnWhenIgnore value (false).]

Specifies if the parser must issue a warning if a line is ignored (emit only if true). AsciiTableTool.setWarnWhenIgnore(expression) is called.

Integer parserSkip [INPUT, OPTIONAL, default=default AsciiParser skipping rows value.]

Number of rows to skip when reading a file. AsciiParser.setSkip(number of lines) is called.

Boolean parserTrim [INPUT, OPTIONAL, default=default AsciiParser trim rows value.]

Specifies if the parser must trim each row when reading a file. AsciiParser.setTrim(strip lines) is called.

Integer parserGuess [INPUT, OPTIONAL, default=GUESS_NONE Guess behavior.]

Specifies if the parser should guess column types. Files should not contain HCSS header (use skip=AsciiReader.HCSS_HEADER for skipping HCSS header or comment these lines) AsciiParser.setGuess(guessType) is called. Valid options: - AsciiParser.GUESS_NONE: (default) file must contains template or template must be provided (no guess) -AsciiParser.GUESS_TRY: guess types based on the first 100 records - AsciiParser.GUESS_ALL: guess types based on all records -AsciiParser.ALL_STRING: each record is a string (no guess required) -AsciiParser.ALL_BOOLEAN: each record is a boolean (no guess required) -AsciiParser.ALL_BYTE: each record is a byte (no guess required) -AsciiParser.ALL_INTEGER: each record is an integer (no guess required) - AsciiParser.ALL_LONG: each record is a long (no guess required) -AsciiParser.ALL_FLOAT: each record is a float (no guess required) -AsciiParser.ALL_DOUBLE: each record is a double (no guess required) -AsciiParser.ALL_COMPLEX: each record is a complex (no guess required)

String parserDelim [INPUT, OPTIONAL, default=comma separator.]

Specifies the field separator. If it is one character, a csvParser is selected. If it is an expression, a RegExpParser is selected.

Boolean parseNames [INPUT, OPTIONAL, default=default AsciiParser parseNames value.]

Specifies if the parser must read column names when reading a file. AsciiParser.firstRowAreColumnNames(expression) is called.

TableTemplate template [INPUT, OPTIONAL, default=extracted from the first file rows.]


Specifies the data structure.

History

- 2007-11-22 - JCS: initial version
- 2008-08-22 - JDS: added optional parameter parserIgnoreWarn (SCR HCSS-3674), serialVersionUID, parameter descriptions, better jython doc

- 2009-09-30 - JSS: modifiers are created through `getCustomModifiers` (SCR HCSS-8253)

2.28. asciiTableWriter

Full Name:	herschel.ia.toolbox.util.AsciiTableWriterTask
Alias:	asciiTableWriter
Type:	Java Task - 
Import:	from herschel.ia.toolbox.util import AsciiTableWriterTask
Category:	utility task

Description

Task for writing ASCII files.

Example

Example 1: write a table in a text file

```
#...
atwt = AsciiTableWriterTask()
atwt(file="file.ascii", table=TableDataset())
#...
```

API Summary

Jython Syntax

```
asciiTableWriter(<file>, <table> [, <configFile>, <configFileOutput>, <formatter>, \
<formatterHeader>, <formatterCommented>, <formatterCommentPrefix>,
<template>, <warn>])
```

Properties

String file [INPUT, MANDATORY, default=null]
TableDataset table [INPUT, MANDATORY, default=null]
String configFile [INPUT, OPTIONAL, default=null]
String configFileOutput [INPUT, OPTIONAL, default=null]
AsciiFormatter formatter [INPUT, OPTIONAL, default=default AsciiTableTool formatter.]
Boolean formatterHeader [INPUT, OPTIONAL, default=default AsciiFormatter header allowance value.]
Boolean formatterCommented [INPUT, OPTIONAL, default=default AsciiFormatter comments allowance value.]
String formatterCommentPrefix [INPUT, OPTIONAL, default=default AsciiFormatter comments prefix value.]
TableTemplate template [INPUT, OPTIONAL, default=extracted from the first file rows.]
Boolean warn [INPUT, OPTIONAL, default=true]

API details


Properties

<code>String</code> file [INPUT, MANDATORY, default=null]
Output file.
<code>TableDataset</code> table [INPUT, MANDATORY, default=null]
TableDataset object to be saved.
<code>String</code> configFile [INPUT, OPTIONAL, default=null]
Configuration file where the formatter (AsciiFormatter), parser (AsciiParser) and table template (TableTemplate) must be specified. When configFile parameter is specified, any parameter related to parser or to table template are not allowed.
<code>String</code> configFileOutput [INPUT, OPTIONAL, default=null]
If a file is specified, an output configuration file will be created.
<code>AsciiFormatter</code> formatter [INPUT, OPTIONAL, default=default AsciiTableTool formatter.]
AsciiFormatter object
<code>Boolean</code> formatterHeader [INPUT, OPTIONAL, default=default AsciiFormatter header allowance value.]
Specifies allowance of header information AsciiFormatter.setHeader(true/false) is called.
<code>Boolean</code> formatterCommented [INPUT, OPTIONAL, default=default AsciiFormatter comments allowance value.]
Specifies allowance of comments when writing a file. AsciiFormatter.setCommented(true/false) is called.
<code>String</code> formatterCommentPrefix [INPUT, OPTIONAL, default=default AsciiFormatter comments prefix value.]
Specifies the prefix of all comments. AsciiFormatter.setCommentPrefix(expression) is called.
<code>TableTemplate</code> template [INPUT, OPTIONAL, default=extracted from the first file rows.]
Specifies the data structure.
<code>Boolean</code> warn [INPUT, OPTIONAL, default=true]
If true, asks confirmation before overwriting.

History

- 2007-11-22 - JCS: initial version
- 2009-09-30 - JSS: modifiers are created through getCustomModifiers (SCR HCSS-8253)
- 2009-12-09 - JDS: warn parameter

2.29. AttribQuery

Full Name:	herschel.ia.pal.query.AttribQuery
Type:	Java Class - 
Import:	from herschel.ia.pal.query import AttribQuery

Description

Attribute Query formulates a query on the attributes of a Product.

In principle this can be the fastest query on the Product Access Layer. Known attributes are:

- type : String
- creator : String
- creationDate: FineTime
- startDate : FineTime
- endDate : FineTime
- modelName : String
- instrument : String
- description : String

Example


Example 1: Example of a query on attributes

```
date = SimpleDateFormat().parse("2008-10-31T12:00:00  
TAI").microsecondsSince1958()  
q = AttribQuery(Product, -"p", -"p.creationDate < -" + str(date) + -"L and  
p.creator = -'Me'")
```

See also

- [???](#)

2.30. AutoCorrelationTask

Full Name:	herschel.ia.toolbox.image.AutoCorrelationTask
Type:	Java Task - 
Import:	from herschel.ia.toolbox.image import AutoCorrelationTask

Description

A Task to calculate the linear Pearson correlation coefficient of an image.

A Task to calculate the linear Pearson correlation coefficient of an image. We start from a $M \times N$ image (M = number of rows, N = number of columns) and calculate the $N \times N$ linear Pearson coefficient matrix. The i 'th row and j 'th column element corresponds to the correlation of the i 'th and j 'th columns of the $M \times N$ matrix.

API Summary


Properties
Image image [INPUT, MANDATORY, default=No default value]
Double2d correlation [OUTPUT, MANDATORY, default=No default value]

API details

Properties

Image image [INPUT, MANDATORY, default=No default value]
The image.
Double2d correlation [OUTPUT, MANDATORY, default=No default value]
The correlation.

2.31. AutomaticContourTask

Full Name:	herschel.ia.toolbox.image.AutomaticContourTask
Type:	Java Task - 
Import:	from herschel.ia.toolbox.image import AutomaticContourTask

Description

A Task for making contours.

A Task for making contours for a given number of contour levels with a given distribution between the given extreme contour values.

API Summary

Properties
Image image [INPUT, MANDATORY, default=No default value]
Integer levels [INPUT, MANDATORY, default=No default value]
Double min [INPUT, MANDATORY, default=No default value]
Double max [INPUT, MANDATORY, default=No default value]
String distribution [INPUT, MANDATORY, default=No default value]
Color[] colors [INPUT, OPTIONAL, default=No default value]
ImageContour contours [OUTPUT, MANDATORY, default=No default value]

API details


Properties

Image image [INPUT, MANDATORY, default=No default value]
The image.
Integer levels [INPUT, MANDATORY, default=No default value]
The number of contour levels.
Double min [INPUT, MANDATORY, default=No default value]
The minimum contour value.
Double max [INPUT, MANDATORY, default=No default value]
The maximum contour value.
String distribution [INPUT, MANDATORY, default=No default value]
The distribution of the contour values.
Color[] colors [INPUT, OPTIONAL, default=No default value]
The colors to visualise the contours.

<code>ImageContour contours [OUTPUT, MANDATORY, default=No default value]</code>
--

The ImageContour.

2.32. AutomaticContourTaskSignatureComponent

Full Name:	herschel.ia.toolbox.image.gui.AutomaticContourTaskSignatureComponent
Type:	Java Class - 
Import:	from herschel.ia.toolbox.image.gui import AutomaticContourTaskSignatureComponent

Description

The task dialog for the AutomaticContourTask.

A task dialog to serve as GUI for the AutomaticContourTask.

API Summary

Constructor	
AutomaticContourTaskSignatureComponent()	The construction of a new AutomaticContourTaskSignatureComponent.
Methods	
check()	Validates the current modifications.
clear()	Clears the current modifications.
Map getParameters()	Returns a map of parameters and modifiers.
setSignature(SignatureApi signature)	Setting the signature.
setVariableSelection(VariableSelection selection)	Setting the variable selection.
JComponent getComponent()	Returns the component.

API details

Constructor


<code>AutomaticContourTaskSignatureComponent()</code>
The construction of a new AutomaticContourTaskSignatureComponent.
The construction of a new AutomaticContourTaskSignatureComponent.

Methods

<code>check()</code>
Validates the current modifications.

check()
Validates the current modifications for this AutomaticContourTaskSignatureComponent.
clear()
Clears the current modifications.
Clears the current modifications for this AutomaticContourTaskSignatureComponent.
Map getParameters()
Returns a map of parameters and modifiers.
Returns a map of parameters and modifiers storing the value selected by the user for this AutomaticContourTaskSignatureComponent.
Return
Map
Returns a map of parameters and modifiers storing the value selected by the user for this AutomaticContourTaskSignatureComponent.
setSignature(SignatureApi signature)
Setting the signature.
Sets the given signature as the signature for this AutomaticContourTaskSignatureComponent.
Argument
SignatureApi signature [INPUT, MANDATORY]
setVariableSelection(VariableSelection selection)
Setting the variable selection.
Sets the given selection as the variable selection for this AutomaticContourTaskSignatureComponent.
Argument
VariableSelection selection [INPUT, MANDATORY]
JComponent getComponent()
Returns the component.
Returns the component for this AutomaticContourTaskSignatureComponent.
Return
JComponent
Returns the component for this AutomaticContourTaskSignatureComponent.

2.33. AverageSpectrum

Full Name:	herschel.ia.toolbox.spectrum.AverageSpectrum
Type:	Java Task - 
Import:	from herschel.ia.toolbox.spectrum import AverageSpectrum

Description

Task for averaging the individual spectra included in one or several `SpectrumContainer` (see in `herschel.ia.dataset.spectrum`).

The averaging operation is applied to the spectral segments included in the different spectra - i.e. the actual spectrum information containing the 'flux' or 'intensity'. These values are averaged on a 'per pixel' basis without checking whether the frequencies assigned to these pixels are well aligned across the different spectra. If this is not the case, the spectra should first be resampled to a common frequency grid. For other attributes characterizing the spectrum information and included in the same spectrum container, suitable defaults can be defined (depending on the runtime types or the 'instrument' metadata element; typically, these defaults are defined by the ICC's). A further option is that the user can register such operations from the command line:

```
avg.register("integrations", "ADD")
```

The input data with the spectra to be averaged can be provided in different forms: Either any data object that implements `SpectrumContainer` or any array of such. However, for a successful processing of the task, the data included in the containers should all be consistent. This means that the number of segments found in all the spectra in the containers should be the same and the lengths of these segments should be consistent. Otherwise an exception should be thrown.

Different selection schemes are available for selecting the point spectra or the segments to be averaged. The most simple scheme is to specify lists of point spectrum indices (`selection=[1, 3, 4, 2]`). In this case, the average is just taken over the point spectra with corresponding indices. Instead of specifying a list of indices (which requires the knowledge of these) other recipes to specify a selection are available. Typical recipes test attribute values to match certain values or to be included in a given interval. These recipes we refer to as selection models. The different options to specify selections can be combined. Here, an AND logic is adopted.

Similarly, you can specify what segments to consider in the averaging. In the most simple case, you can just specify the indices of segments to be considered `segments=[1, 3, 4]`. In more advanced situations you can use a `SegmentSelection` object.

Sometimes, the selection model also provides a natural segmentation of the data into groups. If you want to calculate the average on a per group basis then you can set the `per_group` flag to `true`.

Using the keyword `grouping` the user may specify a grouping model which allows to partition the spectra included in the container into suitable sub-groups. Here, when a grouping model is specified, the average is calculated for each sub-group separately. Grouping and selections can be combined such that the selection model(s) impose(s) constraints on the spectra to be included in the groups.

Example

Example 1: from Jide:

```
from herschel.ia.toolbox.spectrum import AverageSpectrum
avg = AverageSpectrum()
avg.register("integrations", "ADD")
averagedSpectra = avg(ds=spectra)
spectraOut = avg(ds=spectra, segments=[1,3])
```

Example 1: from Jide:

```
spectraOut = avg(ds=spectra, selection={"bbtype":[6031, 6032]})
spectraOut = avg(ds=spectra, selection=[0,1,2,3])
spectraOut = avg(ds=spectra, selection={"LoFrequency":(4000.0,5000.0)})
spectraOut = avg(ds=spectra, selection={"Chopper":([-4.4,5.9],0.1)})
spectraOut = avg(ds=spectra, selection={"Chopper":([-4.4,5.9],0.1),
segments=[1,2]})
from herschel.ia.toolbox.spectrum.selections.models import RangesSelectionModel
spectraOut = avg(ds=spectra, selection=RangesSelectionModel("Chopper",
[-4.4,5.9], 0.1)) # the same as above
# or all combined:
spectraOut = avg(ds=spectra, selection={"bbtype":[6031, 6032], -"Chopper":
([-4.4,5.9],0.1), -"LoFrequency":(4000.0,5000.0)})
from herschel.ia.toolbox.spectrum.selections.models import RangesGroupingModel
averagedSpectraPerGroup = avg(ds=spectra,
grouping=RangesGroupingModel("Chopper", 0.1))
```

API Summary

Properties
Object ds [INPUT, OPTIONAL, default=no default value.]
Object selection [INPUT, OPTIONAL, default=None.]
PyDictionary Map> selection_lookup [INPUT, OPTIONAL, default=no default value.]
PyList selection_index [INPUT, OPTIONAL, default=No default value.]
Object segments [INPUT, OPTIONAL, default=no default value.]
String variant [INPUT, OPTIONAL, default="flux-weight-flag".]
Boolean per_group [INPUT, OPTIONAL, default=no default value.]
GroupingModel grouping [INPUT, OPTIONAL, default=no default value.]
SpectrumContainer result [OUTPUT, OPTIONAL, default=no default value.]

API details

Properties


<p>Object ds [INPUT, OPTIONAL, default=no default value.]</p> <p>Input data to be processed by the task. Several types are possible:</p> <ul style="list-style-type: none"> • SpectrumContainer • Array of SpectrumContainer, i.e. SpectrumContainer[] • List of SpectrumContainer, i.e. List • Any product with implementations of SpectrumContainer inside.
<p>Object selection [INPUT, OPTIONAL, default=None.]</p> <p>Specification of what point spectra the average should be restricted to. Different ways to specify these selections are possible:</p>

Object selection [INPUT, OPTIONAL, default=None.]
<ul style="list-style-type: none"> • Specify a list of indices (in jython) of the point spectra for which the average should be taken. • Use a dictionary (in jython) to specify a selection model (lookup for attributes, filter attributes to be included in ranges or intervals). • Pass a string representation of a jython list or a jython dictionary. The string will be parsed and the processing delegated to the two cases above. • Pass any java instance that implements the SelectionModel interface (for the advanced user).
PyDictionary Map> selection_lookup [INPUT, OPTIONAL, default=no default value.]
Specify a PyDictionary with keys given by the attribute name to look up and a list of admissible values. Behind the scenes, a DiscreteValueSelectionModel is instantiated.
PyList selection_index [INPUT, OPTIONAL, default=No default value.]
Specify a PyList with the indices of the point spectra to be considered.
Object segments [INPUT, OPTIONAL, default=no default value.]
Specify what segments the operation should be applied to. There are two options available: <ul style="list-style-type: none"> • Either pass an instance of a SegmentSelection which gives the information on what segments for each point spectrum included in the container. • Specify a PyList of segment indices.
String variant [INPUT, OPTIONAL, default="flux-weight-flag".]
Specify the variant of processing mode you would like to run (including flags / weights / ...). HCSS defaults include: "flux" / "flux-wave" / "flux-wave-weight" / "flux-weight-flag" / "flux-flag-wave" / "flux-weight-flag-wave"
Boolean per_group [INPUT, OPTIONAL, default=no default value.]
Specify that the average should be calculated on a per group basis - once a selection model is specified that provides a natural grouping of the data.
GroupingModel grouping [INPUT, OPTIONAL, default=no default value.]
Grouping model that can be used to partition the point spectra into groups and calculate the average on a per group basis. When a grouping model is specified, the 'per_group'-flag is obsolete.
SpectrumContainer result [OUTPUT, OPTIONAL, default=no default value]
Result object containing the results of the operation applied.

See also

- [ManyToOneSpectrumTask](#)

2.34. bg

Full Name:	herschel.ia.toolbox.util.BackgroundTask
Alias:	bg
Type:	Java Task - 
Import:	from herschel.ia.toolbox.util import BackgroundTask
Category:	task

Description

Execute jython commands in the background

The bg task is used to execute jython commands in a background Thread. The Thread is returned to allow Thread operations.

Example

Example 1: save specified variable names and values

```
bg( "myfunc( )" )
```

API Summary

Jython Syntax

```
bg( <command> )
```

Properties

[String command](#) [INPUT, YES, default=No default value]

[String command](#) [OUPUT, YES, default=The created Tread]

Limitations

Only applicable from a jide session

API details

Properties

[String command](#) [INPUT, YES, default=No default value]

The name of command to be executed

[String command](#) [OUPUT, YES, default=The created Tread]

Return the Thread created internally to allow Thread operations


See also

- [???](#)

History

- 2004-07-13 - NdC: first release.
- 2010-03-17 - JDS: added BusyJob
- 2010-04-12 - JDS: fixed string escaping

2.35. Bilinear

Full Name:	herschel.ia.numeric.toolbox.interp.Bilinear
Type:	Java Class - 
Import:	from herschel.ia.numeric.toolbox.interp import Bilinear
Category:	numerics/interp

Description

Bilinear interpolation takes 2-D data (i.e. an image), and returns values at arbitrary positions by linearly combining the pixels around the position of interest. Positions are relative to pixel indices, where the center of the top-left pixel is (0,0). No extrapolation is allowed;

If a requested pixel position extends past the size of the array, the value at the edge is used.

Bilinear Algorithm

The Bilinear class uses bilinear interpolation algorithm to compute the value at the point (x,y) on a regular grid where the images[i, j] i=0, m-1 and j=0, n-1 are known. The nearest neighbor extrapolation is used when the point (x,y) falls outside the boundary of the regular grid if the extrapolation is requested.

The Bilinear has the following arguments:

images: Input, a double2d data array

x: Input, a location array

y: Input, a location array

grid: The grid keyword controls how the location arrays specify where interpolates are desired.

If grid is not set, the location arrays, x, y must have the same number of elements. The result has the same structure and number of elements as x. Let n be the number of elements in x and y and the data array is images[n, n], and the result is also a one dimension array of size n.

If grid is set, let m be the number of elements in x, let n be the number of elements in y. The result has dimensions [m, n]. The element [i,j] of the result contains the value of images interpolated at position (xi, yj).

missing: The value to return for elements outside the bounds of images.

If this value is not specified, interpolated positions that fall outside the bounds of the array images - that is, elements of the x or y arguments that are either less than zero or greater than the largest subscript in the corresponding dimension of images - are set equal to the value of the nearest element of the images.

Examples

Example 1: All data points are inside the boundary

```
from herschel.ia.numeric.toolbox.interp import Bilinear
from herschel.ia.numeric import *
from herschel.ia.numeric.toolbox.basic import *
images=RESHAPE(DoubleIcd.range(16), [4,4])
x=[0.0, 1.5]
y=[0.5, 2.5]
bi_imag =Bilinear(Bilinear.GRID, x,y)(images)
print bi_imag
```

Example 1: All data points are inside the boundary

```
[
  [0.5,2.5],
  [6.5,8.5]
-]
```

Example 2: Some data points are on the boundary without GRID is set (the default)

```
from herschel.ia.numeric.toolbox.interp import Bilinear
from herschel.ia.numeric import *
from herschel.ia.numeric.toolbox.basic import *
images=RESHAPE(DoubleIcd.range(16), [4,4])
x=[0.0, 1.5, 3]
y=[0.5, 1.0, 2.5]
bi_imag =Bilinear(x,y)(images)
print bi_imag
[0.5,7.0,14.0]
```

Example 3: Some data points are on the boundary with GRID is set

```
from herschel.ia.numeric.toolbox.interp import Bilinear
from herschel.ia.numeric import *
from herschel.ia.numeric.toolbox.basic import *
images=RESHAPE(DoubleIcd.range(16), [4,4])
x=[0.0, 1.5, 3]
y=[0.5, 1.0, 2.5]
bi_imag =Bilinear(Bilinear.GRID,x,y)(images)
print bi_imag
[
  [0.5,1.0,2.5],
  [6.5,7.0,8.5],
  [12.5,13.0,14.5]
-]
```

Example 4: Input x and y coordinate have different array size when GRID is set.

```
images=RESHAPE(DoubleIcd.range(16), [4,4])
x=[0.0, 1.0]
y=[0.5, 1.0, 2.5, 3.5]
bi_imag=Bilinear(Bilinear.GRID,x,y)(images)
print bi_imag
[
  [0.5,1.0,2.5,3.0],
  [4.5,5.0,6.5,7.0]
]
```

Example 5: Some data points are outside boundary with missing=-1 and GRID is set.

```
from herschel.ia.numeric.toolbox.interp import Bilinear
from herschel.ia.numeric import *
from herschel.ia.numeric.toolbox.basic import *
images=RESHAPE(DoubleIcd.range(16), [4,4])
x=[-0.5,2]
y=[0.5,1.5]
bi_imag1=Bilinear(Bilinear.GRID, x,y, --1.0)(images)
print bi_imag1
[
  [-1.0,-1.0],
  [8.5,9.5]
]
```

Example 6: Some data points are outside boundary with missing=-2 and without GRID is set

```
from herschel.ia.numeric.toolbox.interp import Bilinear
```

Example 6: Some data points are outside boundary with missing=-2 and without GRID is set

```

from herschel.ia.numeric import *
from herschel.ia.numeric.toolbox.basic import *
images=RESHAPE(Double1d.range(16), [4,4])
x=Double1d.range(3).add(1.5)
y=Double1d.range(3).add(1.5)
img1=Bilinear(Bilinear.GRID, x,y, --2.0)(images)
print img1
[7.5,12.5,-2.0]

```

Example 7: Java example

```

import herschel.ia.numeric.Double1d;
import herschel.ia.numeric.Double2d;
import herschel.ia.numeric.toolbox.basic.Reshape;
import herschel.ia.numeric.toolbox.interp.Bilinear;
public class Test {
    public static void main(String[] args) {
        Double2d images=(Double2d) Double1d.range(16).apply(new Reshape(4,4));
        Double1d x=Double1d.range(3).add(0.5);
        Double1d y=Double1d.range(3).add(0.5);
        Double2d newImages1=(Double2d) images.apply(new
Bilinear(Bilinear.GRID,x, y));
        Double1d newImages2=(Double1d) images.apply(new Bilinear(x, y));
        Double2d newImages3=(Double2d) new Bilinear(Bilinear.GRID,x,
y -).of(images);
        Double1d newImages4=(Double1d) new Bilinear(x, y).of(images);
        System.out.println("newImages1="+newImages1);
        System.out.println("newImages2="+newImages2);
        System.out.println("newImages3="+newImages3);
        System.out.println("newImages4="+newImages4);
    }
}
newImages1=[
[2.5,3.5,4.5],
[6.5,7.5,8.5],
[10.5,11.5,12.5]
]
newImages2=[2.5,7.5,12.5]
newImages3=[
[2.5,3.5,4.5],
[6.5,7.5,8.5],
[10.5,11.5,12.5]
]
newImages4=[2.5,7.5,12.5]

```

API Summary

Constructors
Bilinear(Double1d x, Double1d y)
Bilinear(double[] x, double[] y)
Bilinear(Boolean grid, Double1d x, Double1d y)
Bilinear(Boolean grid, double[] x, double[] y)
Bilinear(Double1d x, Double1d y, double missing)
Bilinear(double[] x, double[] y, double missing)
Bilinear(Boolean grid, Double1d x, Double1d y, double missing)
Bilinear(Boolean grid, double[] x, double[] y, double missing)
Method
ArrayData of(Double2d images)

API details

Constructors

Bilinear([DoubleId](#) **x**, [DoubleId](#) **y**)

Arguments

[DoubleId](#) **x** [INPUT, MANDATORY, default=no default value]

The array containing the x "virtual subscripts" of the images for which to interpolate values.

[DoubleId](#) **y** [INPUT, MANDATORY, default=no default value]

The array containing the y "virtual subscripts" of the images for which to interpolate values

Errors

Array

size error The length of array x and y has to be equal. The exception will be thrown if the lengths of the array x and y are not equal.

Bilinear([double\[\]](#) **x**, [double\[\]](#) **y**)

Arguments

[double\[\]](#) **x** [INPUT, MANDATORY, default=no default value]

The array containing the x "virtual subscripts" of the images for which to interpolate values.

[double\[\]](#) **y** [INPUT, MANDATORY, default=no default value]

The array containing the y "virtual subscripts" of the images for which to interpolate values

Errors

Array

size error The length of array x and y has to be equal. The exception will be thrown if the lengths of the array x and y are not equal.

Bilinear([Boolean](#) **grid**, [DoubleId](#) **x**, [DoubleId](#) **y**)

Arguments

[Boolean](#) **grid** [INPUT, MANDATORY, default=no default value]

If grid is true and if the x contains m elements and y contains n elements, the result has dimensions [m, n], where the element [i,j] of the result contains the value of Images interpolated at position (xi, yj). If the grid is false, the result is the same as calling Bilinear without the grid argument.

[DoubleId](#) **x** [INPUT, MANDATORY, default=no default value]

The array containing the x "virtual subscripts" of the images for which to interpolate values.

[DoubleId](#) **y** [INPUT, MANDATORY, default=no default value]

The array containing the y "virtual subscripts" of the images for which to interpolate values

Bilinear([Boolean](#) **grid**, [double\[\]](#) **x**, [double\[\]](#) **y**)

Arguments

[Boolean](#) **grid** [INPUT, MANDATORY, default=no default value]

If grid is true and if the x contains m elements and y contains n elements, the result has dimensions [m, n], where the element [i,j] of the result contains the value of Images interpolated at position (xi, yj). If the grid is false, the result is the same as calling Bilinear without the grid argument.

[double\[\]](#) **x** [INPUT, MANDATORY, default=no default value]

Bilinear([Boolean](#) **grid, double[] x, double[] y)**

The array containing the x "virtual subscripts" of the images for which to interpolate values.
 double[] **y** [INPUT, MANDATORY, default=no default value]
 The array containing the y "virtual subscripts" of the images for which to interpolate values

Bilinear([DoubleId](#) **x, DoubleId y, double missing)**

Arguments

[DoubleId](#) **x** [INPUT, MANDATORY, default=no default value]
 The array containing the x "virtual subscripts" of the images for which to interpolate values.
[DoubleId](#) **y** [INPUT, MANDATORY, default=no default value]
 The array containing the y "virtual subscripts" of the images for which to interpolate values
 double **missing** [INPUT, MANDATORY, default=no default value]
 When missing is given, the value outside the boundary is set to missing.

Bilinear(double[] x, double[] y, double missing)

Arguments

double[] **x** [INPUT, MANDATORY, default=no default value]
 The array containing the x "virtual subscripts" of the images for which to interpolate values.
 double[] **y** [INPUT, MANDATORY, default=no default value]
 The array containing the y "virtual subscripts" of the images for which to interpolate values
 double **missing** [INPUT, MANDATORY, default=no default value]
 When missing is given, the value outside the boundary is set to missing.

Bilinear([Boolean](#) **grid, DoubleId x, DoubleId y, double missing)**

Arguments

[Boolean](#) **grid** [INPUT, MANDATORY, default=no default value]
 If grid is true and if the x contains m elements and y contains n elements, the result has dimensions [m, n], where the element [i,j] of the result contains the value of Images interpolated at position (xi, yj). If the grid is false, the result is the same as calling Bilinear without the grid argument.
[DoubleId](#) **x** [INPUT, MANDATORY, default=no default value]
 The array containing the x "virtual subscripts" of the images for which to interpolate values.
[DoubleId](#) **y** [INPUT, MANDATORY, default=no default value]
 The array containing the y "virtual subscripts" of the images for which to interpolate values
 double **missing** [INPUT, MANDATORY, default=no default value]
 When missing is given, the value outside the boundary is set to missing.

Bilinear([Boolean](#) **grid, double[] x, double[] y, double missing)**

Arguments

[Boolean](#) **grid** [INPUT, MANDATORY, default=no default value]
 If grid is true and if the x contains m elements and y contains n elements, the result has dimensions [m, n], where the element [i,j] of the result contains the value of Images interpolated at position (xi, yj). If the grid is false, the result is the same as calling Bilinear without the grid argument.
 double[] **x** [INPUT, MANDATORY, default=no default value]

Bilinear (Boolean grid, double[] x, double[] y, double missing)

<p>The array containing the x "virtual subscripts" of the images for which to interpolate values. double[] y [INPUT, MANDATORY, default=no default value]</p> <p>The array containing the y "virtual subscripts" of the images for which to interpolate values double missing [INPUT, MANDATORY, default=no default value]</p> <p>When missing is given, the value outside the boundary is set to missing.</p>
--

Method

ArrayData of(Double2d images)

Argument

[Double2d](#) **images** [INPUT, MANDATORY, default=no default value]

A two dimension data array.

Return

ArrayData

the interpolated image values at the (xi, yj).

See also


- [CubicSplineInterpolator](#)
- [NearestNeighborInterpolator](#)
- <http://en.wikipedia.org/wiki/Bilinear>

History

- 2007-03-05 - first: version
- 2007-03-24 - revised: version
- 2008-03-27 - implemented: SPR4608
- 2009-02-25 - add: URM SPR-6124
- March 5, 2007 - first version
- March 24, 2008 - revised version
- Include missing argument to set the value where the point is outside the boundary
- images: A two dimensional array, Double2d data
- Missing:
 - The value to return for elements outside the bounds of iamges. If this keyword is not
 - specified, interpolated positions that fall outside the bounds of the array images -
 - that is, elements of the x or y arguments that are either less than zero or
 - greater than the largest subscript in the corresponding dimension of images - are set
 - equal to the value of the nearest element of images.

- Feb. 24, 2009
- Updated documentation

2.36. BinCentres

Full Name:	herschel.ia.numeric.toolbox.basic.BinCentres
Alias:	BinCentres
Type:	Java Class - 
Import:	from herschel.ia.numeric.toolbox.basic import BinCentres

Description

Return the bin centers for a Numeric data histogram base on a use-supplied bin size.

Examples

Example 1: Plot Histogram over BinCentres for a Double1d

```

from herschel.ia.numeric import *
from herschel.ia.numeric.toolbox.random import RandomGauss
from herschel.ia.numeric.toolbox.basic import Histogram
from herschel.ia.numeric.toolbox.basic import BinCentres
d = Double1d(200000,10.0)
d[90000:115000] = 20.0
d[99000:110000] = 22.0
d[99900:100100] = 23.0
d[99990:100010] = 24.0
d[100000] = 24,5
rnd = RandomGauss()
for ix in range(200000):
    d[ix] += rnd.calc(0.1)
p = PlotXY (d)
binsize = STDDEV (d) * 2.35
print binsize
hist=Histogram(binsize)
bins=BinCentres(binsize)
p2=PlotXY(bins(d),hist(d))
style=p2.getStyle()
style.setChartType(Style.HISTOGRAM)
p.close ()
p2.close ()

```

Example 2: Plot Histogram over BinCentres for a Byte1d

```

vals = [0, 27, 25, 60, 62, 70, 71, 73, 74, 100, 120, 92, 85, 116]
d = Byte1d (vals)
p = PlotXY (d)
binsize = 5
print binsize
hist=Histogram(binsize)
bins=BinCentres(binsize)
p2=PlotXY(bins(d),hist(d))
style=p2.getStyle()
style.setChartType(Style.HISTOGRAM)
p.close ()
p2.close ()

```

Example 3: Plot Histogram over BinCentres for a Double5d

```

d = Double5d (5,4,3,2,1, 2.0)
d.set (4,3,2,1,0, 10.0)
d.set (3,3,2,1,0, 9.0)
d.set (2,3,2,1,0, 8.0)

```

Example 3: Plot Histogram over BinCentres for a Double5d

```
d.set (1,3,2,1,0, 7.0)
binsize = 1.0
hist = Histogram (binsize)
bins = BinCentres (binsize)
p=PlotXY(bins(d),hist(d))
style=p.getStyle()
style.setChartType(Style.HISTOGRAM)
p.close ()
```

API Summary

Jython Syntax

```
bin=BinCentres(1)
<c>=bin(<x>)
```

Property

[MANDATORY. x INPUT \[any scalar array, MANDATORY, default=no default value\]](#)

API details


Property

MANDATORY. x INPUT [any scalar array, MANDATORY, default=no default value]

See also

- [Histogram](#)

2.37. BinomialModel

Full Name:	herschel.ia.numeric.toolbox.fit.BinomialModel
Type:	Java Class - 
Import:	from herschel.ia.numeric.toolbox.fit import BinomialModel

Description

General binomial model of arbitrary degree.

$$f(x,y;p) = \sum_k p_k * x^k * y^{\text{degree} - k}$$


where the sum is over k running from 0 to degree (inclusive).

It is a 2-dimensional linear model.

Example

Example 1: BinomialModel	
<pre>poly = BinomialModel(3 -) # 3rd degree polynomial print poly.getNumberOfParameters() # 4</pre>	

2.38. Bool1d


Full Name:	herschel.ia.numeric.Bool1d
Type:	Java Class - 
Import:	from herschel.ia.numeric import Bool1d

Description

A rectangular numeric boolean array of rank 1.

Numeric arrays are part of the Numeric library. An explanation on how to use them, you can find in chapter **Numeric** -> **Arrays**. (Yes, it should be a link)

2.39. Bool2d


Full Name:	herschel.ia.numeric.Bool2d
Type:	Java Class - 
Import:	from herschel.ia.numeric import Bool2d

Description

A rectangular numeric boolean array of rank 2.

Numeric arrays are part of the Numeric library. An explanation on how to use them, you can find in chapter **Numeric** -> **Arrays**. (Yes, it should be a link)

2.40. Bool3d


Full Name:	herschel.ia.numeric.Bool3d
Type:	Java Class - 
Import:	from herschel.ia.numeric import Bool3d

Description

A rectangular numeric boolean array of rank 3.

Numeric arrays are part of the Numeric library. An explanation on how to use them, you can find in chapter **Numeric** -> **Arrays**. (Yes, it should be a link)

2.41. Bool4d


Full Name:	herschel.ia.numeric.Bool4d
Type:	Java Class - 
Import:	from herschel.ia.numeric import Bool4d

Description

A rectangular numeric boolean array of rank 4.

Numeric arrays are part of the Numeric library. An explanation on how to use them, you can find in chapter **Numeric** -> **Arrays**. (Yes, it should be a link)

2.42. Bool5d


Full Name:	herschel.ia.numeric.Bool5d
Type:	Java Class - 
Import:	from herschel.ia.numeric import Bool5d

Description

A rectangular numeric boolean array of rank 5.

Numeric arrays are part of the Numeric library. An explanation on how to use them, you can find in chapter **Numeric** -> **Arrays**. (Yes, it should be a link)

2.43. BoxCarFilter

Full Name:	herschel.ia.numeric.toolbox.filter.BoxCarFilter
Alias:	BoxCarFilter
Type:	Java Class - 
Import:	from herschel.ia.numeric.toolbox.filter import BoxCarFilter

Description

Creates a box car filter, that can be applied to numeric arrays of rank 1.

Example

Example 1: Apply BoxCarFilter on a Int1d
<pre>x=Int1d([1,3,2,4,6,5]) f=BoxCarFilter(2) print f(x) # [0.5,2.0,2.5,3.0,5.0,5.5]</pre>

API Summary

Jython Syntax
<pre><f>=BoxCarFilter(<width> [, <center>=true false] [, <edge>=Convolution.ZEROES CIRCULAR REPEAT]) <x>=<f>(<x>)</pre>

Properties
integer width [INPUT, MANDATORY, default=no default value]
boolean center [INPUT, NOT_MANDATORY, default=false]
Convolution.ZEROES CIRCULAR REPEAT edge [INPUT, NOT_MANDATORY, default=Convolution.ZEROES]

API details

Properties

integer width [INPUT, MANDATORY, default=no default value]
It must be an integer value

boolean center [INPUT, NOT_MANDATORY, default=false]
Set center to true or false

Convolution.ZEROES CIRCULAR REPEAT edge [INPUT, NOT_MANDATORY, default=Convolution.ZEROES]
Set edge to the following: <ul style="list-style-type: none"> • edge=Convolution.ZEROES: Set result to zero at edges. • edge=Convolution.CIRCULAR: Wrap around at edges (circular convolution).


```
Convolution.ZEROES|CIRCULAR|REPEAT edge [INPUT, NOT_MANDATORY,  
default=Convolution.ZEROES]
```

- edge=Convolution.REPEAT: Repeat the edge values of input array.

See also

- [Convolution](#)
- [GaussianFilter](#)

2.44. BoxCarSmoothingTask

Full Name:	herschel.ia.toolbox.image.BoxCarSmoothingTask
Type:	Java Task - 
Import:	from herschel.ia.toolbox.image import BoxCarSmoothingTask

Description

A Task to smooth an image using a convolution with a box car.

A Task to smooth an image by convolving it with a box car function.

API Summary


Properties
Image image [INPUT, MANDATORY, default=No default value]
Double width [INPUT, MANDATORY, default=Default value : Integer(3)]
Image smoothed [OUTPUT, MANDATORY, default=No default value]

API details

Properties

Image image [INPUT, MANDATORY, default=No default value]
The image before smoothing.
Double width [INPUT, MANDATORY, default=Default value : Integer(3)]
The width of the box car function.
Image smoothed [OUTPUT, MANDATORY, default=No default value]
The image after smoothing.

2.45. Byte1d


Full Name:	herschel.ia.numeric.Byte1d
Type:	Java Class - 
Import:	from herschel.ia.numeric import Byte1d

Description

A rectangular numeric byte array of rank 1.

Numeric arrays are part of the Numeric library. An explanation on how to use them, you can find in chapter **Numeric** -> **Arrays**. (Yes, it should be a link)

2.46. Byte2d


Full Name:	herschel.ia.numeric.Byte2d
Type:	Java Class - 
Import:	from herschel.ia.numeric import Byte2d

Description

A rectangular numeric byte array of rank 2.

Numeric arrays are part of the Numeric library. An explanation on how to use them, you can find in chapter **Numeric** -> **Arrays**. (Yes, it should be a link)

2.47. Byte3d


Full Name:	herschel.ia.numeric.Byte3d
Type:	Java Class - 
Import:	from herschel.ia.numeric import Byte3d

Description

A rectangular numeric byte array of rank 3.

Numeric arrays are part of the Numeric library. An explanation on how to use them, you can find in chapter **Numeric** -> **Arrays**. (Yes, it should be a link)

2.48. Byte4d


Full Name:	herschel.ia.numeric.Byte4d
Type:	Java Class - 
Import:	from herschel.ia.numeric import Byte4d

Description

A rectangular numeric byte array of rank 4.

Numeric arrays are part of the Numeric library. An explanation on how to use them, you can find in chapter **Numeric** -> **Arrays**. (Yes, it should be a link)

2.49. Byte5d

Full Name:	herschel.ia.numeric.Byte5d
Type:	Java Class - 
Import:	from herschel.ia.numeric import Byte5d

Description

A rectangular numeric byte array of rank 5.

Numeric arrays are part of the Numeric library. An explanation on how to use them, you can find in chapter **Numeric** -> **Arrays**. (Yes, it should be a link)

2.50. CachedPool

Full Name:	herschel.ia.pal.pool.cache.CachedPool
Type:	Java Class - 
Import:	from herschel.ia.pal.pool.cache import CachedPool

Description

Implementation of a PAL ProductPool that adds caching functionality to a remote ProductPool.

Example

Example 1: # combining a cached remote pool with local pool
<pre> hsa=CachedPool(HsaReadPool() -) local=LocalStoreFactory.getStore("local") store=ProductStorage([local,hsa]) # load an observation by tag name tag="OBS:S:00500019C0" obs=store.load(tag).product # modify and save new version into local pool obs.creator="My funny Valentine" ref=store.saveAs(obs,tag) </pre>

API Summary

Constructors
CachedPool(String remote) Creates an instance of a CachedPool that is wrapped around another
CachedPool(String remote, ProductPool delegated) Creates an instance of a CachedPool that is wrapped around another

API details

Constructors

CachedPool(String remote)
Creates an instance of a CachedPool that is wrapped around another supplied ProductPool.
Argument String remote [INPUT, MANDATORY, default=The name of the ProductPool] this CachedPool should cache.
Return CachedPool An instance of the CachedPool.

CachedPool([String](#) remote, ProductPool delegated)

Creates an instance of a CachedPool that is wrapped around another supplied ProductPool.

Arguments

[String](#) **remote** [INPUT, MANDATORY, default=The name of the ProductPool]

this CachedPool should cache.


ProductPool **delegated** [INPUT, MANDATORY]

Return

[CachedPool](#)

An instance of the CachedPool.

2.51. CEIL

Full Name:	herschel.ia.numeric.toolbox.basic.Ceiling
Alias:	CEIL
Type:	Java Class - 
Import:	from herschel.ia.numeric.toolbox.basic import Ceiling

Description

Gives the smallest integer greater than or equal to x .

Returns a float array for float input array and double array for any other numeric array type.

Example

Example 1: Apply CEIL on a Float1d

```
x=Float1d([0.1,0.5,0.9])
print CEIL(x) # [1.0,1.0,1.0] *
```

API Summary

Jython Syntax

```
<y>=CEIL(<x>)
```

Properties

[any array of rank 1 \$x\$ \[INPUT, MANDATORY, default=no default value\]](#)
[float or double array \$y\$ \[OUTPUT, MANDATORY, default=no default value\]](#)

API details

Properties

any array of rank 1 x [INPUT, MANDATORY, default=no default value]

The input array can be an array of rank 1


float or double array y [OUTPUT, MANDATORY, default=no default value]

Returns a float array for float input array and double array for any other numeric array type.

See also

- [FLOOR](#)
- [ROUND](#)

2.52. ChannelMapPlotting

Full Name:	herschel.ia.gui.cube.ChannelMapPlotting
Type:	Java Class - 
Import:	from herschel.ia.gui.cube import ChannelMapPlotting

Description

ChannelMapPlotting

A class to deal with ChannelMapPlotting.

API Summary

Constructor	
ChannelMapPlotting(boolean useAsComponent, CubeSpectrumAnalysis-Toolbox toolbox)	Constructor for ChannelMapPlotting. When the new ChannelMapPlotting is
Methods	
int getUsedContourValues()	Returns the used type of contour values for this ChannelMapPlotting
int getUsedContourLevelRange()	Returns the used type of contour level range for this
String getErrorMessage()	Returns a convenient error message for the given input. If no
ArrayList getContourValues()	Returns a list with all contour values for this ChannelMapPlotting.
ArrayList getContourColors()	Returns the colors that should be used for the contours of the
int getNumberOfContourLevels()	Returns the number of contour levels for this ChannelMapPlotting if
double[] getContourLevels()	Returns the used lower and upper contour value for this
double getMinimumContourLength()	Returns the minimum length for contours to be drawn on the image.
String getUsedDistribution()	Returns the type of distribution of the contour levels, that is
void setUsedContourValues(int usedvalue)	Returns the used type of contour level range for this
boolean checkInput()	Checks whether the input for this ChannelMapPlotting is correct. The
ArrayList calculateContourValues()	Calculates the contour values for the given number of contour
ArrayList getImageFigures()	

Methods

Returns all ImageFigures used for this ChannelMapPlotting.

API details

Constructor

ChannelMapPlotting(boolean useAsComponent, [CubeSpectrumAnalysis-Toolbox](#) toolbox)

Constructor for ChannelMapPlotting. When the new ChannelMapPlotting is component-based, a window for contour plotting is opened; otherwise nothing will be shown.

Arguments

boolean **useAsComponent** [INPUT, MANDATORY]

[CubeSpectrumAnalysisToolbox](#) **toolbox** [INPUT, MANDATORY]

Methods

int [getUsedContourValues\(\)](#)

Returns the used type of contour values for this ChannelMapPlotting (manual or automatic).

Return

int

Returns the used type of contour values for this ChannelMapPlotting (manual or automatic).

int [getUsedContourLevelRange\(\)](#)

Returns the used type of contour level range for this ChannelMapPlotting (manual or image cut levels).

Return

int

Returns the used type of contour level range for this ChannelMapPlotting (manual or image cut levels).

String [getErrorMessage\(\)](#)

Returns a convenient error message for the given input. If no errors occur in the input, null is returned.

Return

[String](#)

Returns a convenient error message for the given input. If no errors occur in the input, null is returned.

ArrayList [getContourValues\(\)](#)

Returns a list with all contour values for this ChannelMapPlotting.

[ArrayList](#) getContourValues()

Return

[ArrayList](#)

Returns a list with all contour values for this ChannelMapPlotting.

[ArrayList](#) getContourColors()

Returns the colors that should be used for the contours of the different contour values.

Return

[ArrayList](#)

Returns the colors that should be used for the contours of the different contour values.

int getNumberOfContourLevels()

Returns the number of contour levels for this ChannelMapPlotting if the contour values are to be calculated automatically; otherwise -1 is returned.

Return

int

Returns the number of contour levels for this ChannelMapPlotting if the contour values are to be calculated automatically; otherwise -1 is returned.

double[] getContourLevels()

Returns the used lower and upper contour value for this ChannelMapPlotting (manual or image cut levels).

Return

double[]

The used lower and upper contour value for this ChannelMapPlotting (manual or image cut levels).

double getMinimumContourLength()

Returns the minimum length for contours to be drawn on the image.

Return

double

Returns the minimum length for contours to be drawn on the image.

[String](#) getUsedDistribution()

Returns the type of distribution of the contour levels, that is used for this ChannelMapPlotting (linear, log or ln).

Return

[String](#)

[String](#) getUsedDistribution()

Returns the type of distribution of the contour levels, that is used for this ChannelMapPlotting (linear, log or ln).

void setUsedContourValues(int usedvalue)

Returns the used type of contour level range for this ChannelMapPlotting (manual or image cut levels).

Argument

int **usedvalue** [INPUT, MANDATORY]

Return

void

Returns the used type of contour level range for this ChannelMapPlotting (manual or image cut levels).

boolean checkInput()

Checks whether the input for this ChannelMapPlotting is correct. The error message is adapted, when errors occur.

Return

boolean

Returns true if the given input for this ChannelMapPlotting is correct; false otherwise.

[ArrayList](#) calculateContourValues()

Calculates the contour values for the given number of contour

levels and their distribution, and extreme values if you indicated to use the automatically calculated contour values; otherwise null is returned.

Return

[ArrayList](#)

Returns the contour values for the given number of contour levels and their distribution, and extreme contour values if you indicated to use the automatically calculated contour values; otherwise null is returned.

[ArrayList](#) getImageFigures()


Returns all ImageFigures used for this ChannelMapPlotting.

Return

[ArrayList](#)

Returns all ImageFigures used for this ChannelMapPlotting.

2.53. CholeskyDecomposition

Full Name:	herschel.ia.numeric.toolbox.matrix.CholeskyDecomposition
Alias:	CholeskyDecomposition
Type:	Java Class - 
Import:	from herschel.ia.numeric.toolbox.matrix import CholeskyDecomposition

Example

Example 1:

```
##### CholeskyDecomposition #####
#####
#
### HIPE ###
A=Double2d (\
    [\
        [3944.2141901709658, --39.83013886638742,
24.856149542676924, --49.90362121047403],\
        [-39.83013886638742, 3958.7850982903415, 39.244285251565444,
37.29778768650653],\
        [24.856149542676924, 39.244285251565444,
3952.2910884701473, --1.216735893731549],\
        [-49.90362121047403, 37.29778768650653, --1.216735893731549,
3950.7089115298527],\
    -])
B = Double2d (\
    [\
        [1184.8157857233916],\
        [-360.85066234104187],\
        [-426.7929837904702],\
        [-179.95015720518913],\
    ])
CholeskyD = CholeskyDecomposition(A)
solution = B.apply(CholeskyD)
print solution
[
[0.29968672246274],
[-0.08666981085100102],
[-0.10902299389485229],
[-0.04097866183941969]
]
isSpd = CholeskyD.isSpd
print isSpd
1
triangularFactor = CholeskyD.l
print triangularFactor
[
[62.8029791504429,0.0,0.0,0.0],
[-0.6342077940438997,62.91568070651636,0.0,0.0],
[0.39577978431778954,0.6277495758130477,62.86286962351098,0.0],
[-0.7946059547737568,0.5848119575853922,-0.020192561781548726,62.84691798442629]
]
```

API Summary

Jython Syntax

```
A=Double2d()
B=Double2d()
res=B.apply(CholeskyDecomposition(A))
```


Jython Syntax

```
CholeskyD = CholeskyDecomposition(A)
solution = B.apply(CholeskyD)
isSpd = CholeskyD.isSpd
triangularFactor = CholeskyD.l
```

Property

[Double2d](#) **A** [INPUT, MANDATORY, default=no default value]


API details

Property

[Double2d](#) **A** [INPUT, MANDATORY, default=no default value]

Input must be a Double2d or Float2d array.

2.54. CircleHistogramExplorer

Full Name:	herschel.ia.gui.image.CircleHistogramExplorer
Type:	Java Class - 
Import:	from herschel.ia.gui.image import CircleHistogramExplorer

Description

An explorer for CircleHistogramProducts.

An explorer for CircleHistogramProducts.

API Summary

Constructors
CircleHistogramExplorer() The construction of a new CircleHistogramExplorer.
CircleHistogramExplorer(Object object) The construction of a new CircleHistogramExplorer associated with the given object.
Methods
String getName() Returns the name.
String getDescription() Returns the description.
boolean canHandle(Class className) Checks whether this CircleHistogramExplorer can handle objects of the given class.
setObject(Object object) Sets the object.
CircleHistogramProduct getObject() Returns the object.
Class getVariableType() Returns the expected variable type.
JScrollPane getParameterTable() Returns the parameter table.

API details

Constructors

CircleHistogramExplorer() The construction of a new CircleHistogramExplorer. The construction of a new CircleHistogramExplorer.
CircleHistogramExplorer(Object object) The construction of a new CircleHistogramExplorer associated with the given object.

CircleHistogramExplorer(Object object)

The construction of a new CircleHistogramExplorer associated with the given object.

Argument

Object object [INPUT, MANDATORY]

Methods

String getName()

Returns the name.

Returns the name for this CircleHistogramExplorer.

Return

String

Returns the name for this CircleHistogramExplorer.

String getDescription()

Returns the description.

Returns the description for this CircleHistogramExplorer.

Return

String

Returns the description for this CircleHistogramExplorer.

boolean canHandle(Class className)

Checks whether this CircleHistogramExplorer can handle objects of the given class.

Returns true if this CircleHistogramExplorer can handle objects of the given class; false otherwise.

Argument

Class className [INPUT, MANDATORY]

Return

boolean

Returns true if this CircleHistogramExplorer can handle objects of the given class; false otherwise.

setObject(Object object)

Sets the object.

Sets the object for this CircleHistogramExplorer to the given object.

Argument

Object object [INPUT, MANDATORY]

CircleHistogramProduct getObject()

Returns the object.

Returns the object for this CircleHistogramExplorer.

[CircleHistogramProduct](#) getObject()

Return

[CircleHistogramProduct](#)

Returns the object for this CircleHistogramExplorer.

[Class](#) getVariableType()

Returns the expected variable type.

Returns the expected variable type for this CircleHistogramExplorer.

Return

[Class](#)

Returns the expected variable type for this CircleHistogramExplorer.

[JScrollPane](#) getParameterTable()

Returns the parameter table.


Returns the parameter table for this CircleHistogramExplorer.

Return

[JScrollPane](#)

Returns the parameter table for this CircleHistogramExplorer

2.55. CircleHistogramPanel

Full Name:	herschel.ia.toolbox.image.gui.CircleHistogramPanel
Type:	Java Class - 
Import:	from herschel.ia.toolbox.image.gui import CircleHistogramPanel

Description

A panel for the CircleHistogramPanel.

A panel to serve as GUI for the CircleHistogramTask.

API Summary

Constructor
CircleHistogramPanel() The construction of a new CircleHistogramPanel.
Methods
drawFigure() Draws the circle on the image.
updateFigure() Updates the circle.
trigger() Triggers the execution of the associated task.
updateHistogram() Updates the histogram.

API details

Constructor

<code>CircleHistogramPanel()</code>
The construction of a new CircleHistogramPanel.
The construction of a new CircleHistogramPanel.


Methods

<code>drawFigure()</code>
Draws the circle on the image.
Draws the circle on the image associated with this CircleHistogramPanel.
<code>updateFigure()</code>
Updates the circle.
Updates the circle associated with this CircleHistogramPanel.

trigger()
Triggers the execution of the associated task.
Triggers the execution of the task associated with this CircleHistogramPanel.

updateHistogram()
Updates the histogram.
Updates the histogram associated with this CircleHistogramPanel.

2.56. CircleHistogramProduct

Full Name:	herschel.ia.dataset.image.CircleHistogramProduct
Type:	Java Class - 
Import:	from herschel.ia.dataset.image import CircleHistogramProduct

Description

A class to deal with the results of a circle histogram.

API Summary

Constructor
CircleHistogramProduct() The constructor of a new CircleHistogramProduct.
Methods
setCenter(double centerX, double centerY) Sets the center for this CircleHistogramProduct
setCenter(double centerX, double centerY, String centerRA, String centerDec) Sets the center for this CircleHistogramProduct
setRadius(double pixels) Sets the radius for this CircleHistogramProduct
setRadius(double pixels, double arcsec) Sets the radius for this ImageHistogramProduct
DoubleId getCenterPixelCoordinates() Returns the center for this CircleHistogramProduct in
StringId getCenterSkyCoordinates() Returns the center for this CircleHistogramProduct in
double getRadiusPixels() Returns the radius for this CircleHistogramProduct in pixels.
double getRadiusArcsec() Returns the radius for this CircleHistogramProduct in arcsec.

API details

Constructor

CircleHistogramProduct()
The constructor of a new CircleHistogramProduct.

Methods

setCenter(double centerX, double centerY)
Sets the center for this CircleHistogramProduct

setCenter(double centerX, double centerY)

to the given pixel coordinates.

Arguments

double **centerX** [INPUT, MANDATORY]
 double **centerY** [INPUT, MANDATORY]

setCenter(double centerX, double centerY, [String](#) centerRA, [String](#) centerDec)

Sets the center for this CircleHistogramProduct

to the given pixel and sky coordinates.

Arguments

double **centerX** [INPUT, MANDATORY]
 double **centerY** [INPUT, MANDATORY]
[String](#) **centerRA** [INPUT, MANDATORY]
[String](#) **centerDec** [INPUT, MANDATORY]

setRadius(double pixels)

Sets the radius for this CircleHistogramProduct

to the given number of pixels.

Argument

double **pixels** [INPUT, MANDATORY]

setRadius(double pixels, double arcsec)

Sets the radius for this ImageHistogramProduct

to the given number of pixels and arcsec.

Arguments

double **pixels** [INPUT, MANDATORY]
 double **arcsec** [INPUT, MANDATORY]

[DoubleId](#) **getCenterPixelCoordinates()**

Returns the center for this CircleHistogramProduct in

pixel coordinates.

Return

[DoubleId](#)

Returns the center for this CircleHistogramProduct in pixel coordinates.

[StringId](#) **getCenterSkyCoordinates()**

Returns the center for this CircleHistogramProduct in

sky coordinates.

Return

[StringId](#) getCenterSkyCoordinates()

[StringId](#)

Returns the center for this CircleHistogramProduct in sky coordinates.

double getRadiusPixels()

Returns the radius for this CircleHistogramProduct in pixels.

Return

double

Returns the radius for this CircleHistogramProduct in pixels.

double getRadiusArcsec()


Returns the radius for this CircleHistogramProduct in arcsec.

Return

double

Returns the radius for this CircleHistogramProduct in arcsec.

2.57. CircleHistogramTask

Full Name:	herschel.ia.toolbox.image.CircleHistogramTask
Type:	Java Task - 
Import:	from herschel.ia.toolbox.image import CircleHistogramTask

Description

A Task to make a histogram within a circle.

A Task to make a histogram of a region of interest, which is bounded by a circle.

API Summary

Properties
Image image [INPUT, MANDATORY, default=No default value]
Double lowCut [INPUT, OPTIONAL, default=Default value : NaN]
Double highCut [INPUT, OPTIONAL, default=Default value : NaN]
Integer bins [INPUT, MANDATORY, default=Default value : new Integer(2000)]
Double centerX [INPUT, OPTIONAL, default=Default value : NaN]
Double centerY [INPUT, OPTIONAL, default=Default value : NaN]
String centerRA [INPUT, OPTIONAL, default=Default value : "center-RA"]
String centerDec [INPUT, OPTIONAL, default=Default value : "centerDec"]
Double radiusPixels [INPUT, OPTIONAL, default=Default value : NaN]
Double radiusArcsec [INPUT, OPTIONAL, default=Default value : NaN]

API details

Properties

Image image [INPUT, MANDATORY, default=No default value]
The image.
Double lowCut [INPUT, OPTIONAL, default=Default value : NaN]
The lower cut level.
Double highCut [INPUT, OPTIONAL, default=Default value : NaN]
The upper cut level.
Integer bins [INPUT, MANDATORY, default=Default value : new Integer(2000)]
The number of bins.
Double centerX [INPUT, OPTIONAL, default=Default value : NaN]
The x-pixel-coordinate of the center of the circle.

Double centerY [INPUT, OPTIONAL, default=Default value : NaN]

The y-pixel-coordinate of the center of the circle.

String centerRA [INPUT, OPTIONAL, default=Default value : "centerRA"]

The right ascension of the center of the circle.

String centerDec [INPUT, OPTIONAL, default=Default value : "centerDec"]

The declination of the center of the circle.


Double radiusPixels [INPUT, OPTIONAL, default=Default value : NaN]

The radius of the circle in pixels.

Double radiusArcsec [INPUT, OPTIONAL, default=Default value : NaN]

The radius of the circle in arcsec.

2.58. CircularIntegrationTask


Full Name:	herschel.ia.toolbox.image.CircularIntegrationTask
Type:	Java Task - 
Import:	from herschel.ia.toolbox.image import CircularIntegrationTask

Description

An abstract Task that deals with the integration of a circular region.

To do the integration of a circular region, this region is divided into : * the central pixel (in which the center of the circular region lies) * the rows and columns between two quadrants

2.59. clamp

Full Name:	herschel.ia.toolbox.image.ClampTask
Alias:	clamp
Type:	Java Task - 
Import:	from herschel.ia.toolbox.image import ClampTask
Category:	task/image

Description

Clamping images and cubes.

The Clamp task for Images and Cubes. ClampTask is a task which restricts the range of pixel values for a source image by constraining the range of pixels to defined "low" and "high" values. The clamp algorithm sets all the pixels whose value is below "low" to "low" and sets all the pixels whose value is above "high" to "high". The "low" value must be less than or equal to the "high" value.

Example

Example 1: Clamping an image to lower value 11 and higher value 240

```
clamp(image = image, low = 11, high = 240)
```

API Summary

Jython Syntax

```
clamp(image, 11, 240)
```

Properties

Image **image** [INPUT, MANDATORY, default=No default value]

double **low** [INPUT, OPTIONAL, default=No default value]

double **high** [INPUT, OPTIONAL, default=No default value]

Image **clampedImage** [OUTPUT, MANDATORY, default=No default value]

API details

Properties

Image **image** [INPUT, MANDATORY, default=No default value]

The Image to be clamped.

double **low** [INPUT, OPTIONAL, default=No default value]

The low clamp value.


double **high** [INPUT, OPTIONAL, default=No default value]

The high clamp value.

Image **clampedImage** [OUTPUT, MANDATORY, default=No default value]

The clamped image.

2.60. ClampTaskSignatureComponent

Full Name:	herschel.ia.toolbox.image.gui.ClampTaskSignatureComponent
Type:	Java Class - 
Import:	from herschel.ia.toolbox.image.gui import ClampTaskSignatureComponent

Description

The task dialog for the ClampTask.

A task dialog to serve as GUI for the ClampTask.

API Summary

Constructor
ClampTaskSignatureComponent() The construction of a new ClampTaskSignatureComponent.
Methods
check() Validates the current modifications.
clear() Clears the current modifications.
getParameters() Returns a map of parameters and modifiers storing the value selected by the user.
setSignature(SignatureApi signature) Setting the signature.
setVariableSelection(VariableSelection selection) Setting the variable selection.
JLabel newLabel(String title, String tooltip) Returns a lable with the given title and tooltip.
JComponent getComponent() Returns the component.

API details

Constructor


<code>ClampTaskSignatureComponent()</code>
The construction of a new ClampTaskSignatureComponent.
The construction of a new ClampTaskSignatureComponent.

Methods

<code>check()</code>
Validates the current modifications.

check()
Validates the current modifications for this ClampTaskSignatureComponent.
clear()
Clears the current modifications. Clears the current modifications for this ClampTaskSignatureComponent.
getParameters()
Returns a map of parameters and modifiers storing the value selected by the user. Returns a map of parameters and modifiers storing the value selected by the user for this ClampTaskSignatureComponent.
setSignature(SignatureApi signature)
Setting the signature. Sets the given signature as the signature for this ClampTaskSignatureComponent. Argument SignatureApi signature [INPUT, MANDATORY]
setVariableSelection(VariableSelection selection)
Setting the variable selection. Sets the given selection as the variable selection for this ClampTaskSignatureComponent. Argument VariableSelection selection [INPUT, MANDATORY]
JLabel newLabel(String title, String tooltip)
Returns a lable with the given title and tooltip. Returns a lable with the given title and tooltip. Arguments String title [INPUT, MANDATORY] String tooltip [INPUT, MANDATORY] Return JLabel Returns a lable with the given title and tooltip.
JComponent getComponent()
Returns the component. Returns the component for this ClampTaskSignatureComponent. Return JComponent Returns the component for this ClampTaskSignatureComponent.

2.61. clear

Full Name:	herschel.ia.toolbox.util.ClearTask
Alias:	clear
Type:	Java Task - 
Import:	from herschel.ia.toolbox.util import ClearTask
Category:	task

Description

Clear variables from the jython session.

The clear task is used to clear a variable from the jython session and reclaim the allocated memory. The task allows also for clearing every variable with the exception of reserved names. When used as in the following "Clear(all=True)" deletes all parameters made by the user - including those from user set up files

Examples

Example 1: clear a single variable

```
clear("variableName")
```

Example 2: clear a list of variables

```
clear("variableA, variableB, -...")
```

Example 3: clear all variables created by the user - including those from user set up files

```
clear(all=True)
```

API Summary

Jython Syntax

```
clear("variableName")
clear("variableA,variableB")
clear(all=True)
```

Properties

```
String variable [INPUT, No, default=NO default value]
Boolean all [INPUT, NO, default=False]
```

Limitations

The current implementation allows deleting of packages and class, no pre-filter is performed

Miscellaneous

No miscellaneous

API details

Properties

<code>String</code> <code>variable</code> [<code>INPUT</code> , <code>NO</code> , <code>default=NO</code> <code>default value</code>]

The name of the variable to erase or a list (comma separated) of variables


<code>Boolean</code> <code>all</code> [<code>INPUT</code> , <code>NO</code> , <code>default=False</code>]

The all option for erasing every variable

History

- 2004-07-13 - NdC: first release

2.62. Complex1d


Full Name:	herschel.ia.numeric.Complex1d
Type:	Java Class - 
Import:	from herschel.ia.numeric import Complex1d

Description

A rectangular numeric Complex array of rank 1.

Numeric arrays are part of the Numeric library. An explanation on how to use them, you can find in chapter **Numeric** -> **Arrays**. (Yes, it should be a link)

2.63. Complex2d


Full Name:	herschel.ia.numeric.Complex2d
Type:	Java Class - 
Import:	from herschel.ia.numeric import Complex2d

Description

A rectangular numeric Complex array of rank 2.

Numeric arrays are part of the Numeric library. An explanation on how to use them, you can find in chapter **Numeric** -> **Arrays**. (Yes, it should be a link)

2.64. Complex3d


Full Name:	herschel.ia.numeric.Complex3d
Type:	Java Class - 
Import:	from herschel.ia.numeric import Complex3d

Description

A rectangular numeric Complex array of rank 3.

Numeric arrays are part of the Numeric library. An explanation on how to use them, you can find in chapter **Numeric** -> **Arrays**. (Yes, it should be a link)

2.65. Complex4d


Full Name:	herschel.ia.numeric.Complex4d
Type:	Java Class - 
Import:	from herschel.ia.numeric import Complex4d

Description

A rectangular numeric Complex array of rank 4.

Numeric arrays are part of the Numeric library. An explanation on how to use them, you can find in chapter **Numeric** -> **Arrays**. (Yes, it should be a link)

2.66. Complex5d


Full Name:	herschel.ia.numeric.Complex5d
Type:	Java Class - 
Import:	from herschel.ia.numeric import Complex5d

Description

A rectangular numeric Complex array of rank 5.

Numeric arrays are part of the Numeric library. An explanation on how to use them, you can find in chapter **Numeric** -> **Arrays**. (Yes, it should be a link)

2.67. CONCATENATE

Full Name:	herschel.ia.numeric.toolbox.basic.Concatenate
Alias:	CONCATENATE
Type:	Java Class - 
Import:	from herschel.ia.numeric.toolbox.basic import Concatenate

Description

Concatenates an array of rank > 1 into an array rank 1 of the same type.

The elements are concatenated starting from the highest dimension.

Example

Example 1: Apply ABS on a Int1d

```
x=Double2d([ [1,2,3,4],[5,6,7,8] -])
print CONCATENATE(x) # [1.0,2.0,3.0,4.0,5.0,6.0,7.0,8.0]
x=Int3d([ [ [1,2],[3,4] -], [ [5,6],[7,8] -], [ [9,10],[11,12] -] -])
print CONCATENATE(x) # [1,2,3,4,5,6,7,8,9,10,11,12]
```

API Summary

Jython Syntax

```
<y>=CONCATENATE(<x>)
```

Properties

[any array type **x** \[INPUT, MANDATORY, default=no default value\]](#)

[an array of any type **y** \[OUTPUT, MANDATORY, default=no default value\]](#)

API details

Properties

any array type **x [INPUT, MANDATORY, default=no default value]**

The input must be an array of rank > 1


an array of any type **y [OUTPUT, MANDATORY, default=no default value]**

The result is an array of rank 1 .

See also

- [RESHAPE](#)

2.68. Condense

Full Name:	herschel.ia.numeric.toolbox.basic.Collapse
Type:	Java Class - 
Import:	from herschel.ia.numeric.toolbox.basic import Collapse

Description

Condense array x along dimension d applying function f using a truncation size.

This function obtains an array where each item is the result of applying a function (ie: SUM, COS, etc.) to the original array to the specified dimension. The 'chunkSize' argument specifies how many items are used for each item of the returned array.

Condense modes (see 'mode' parameter):

-Default mode : Condense array x along dimension d applying function f using a truncation size

-Boxcar mode : Run a boxcar window of a certain size over the data array

Condense works like any ArrayToNumber or ArrayReducer over a dimension (ie: SUM(array,along_dim_d)) but you can split the original array into chunks. You will obtain an array with the number of chunks that you have specified (the number of items of the returned array is truncated if it is required, see example). Nevertheless, if you use 'boxcar' mode, chunks are created based on the current position for each item of the returned array (see 'mode' parameter and the example).

Example

Example 1: Untitled

```
IA>>x=RESHAPE(Int1d.range(4*2),[4,2])
IA>>print x
[
  [0,1],
  [2,3],
  [4,5],
  [6,7]
-]
IA>>print Condense(0,1,SUM)(x) #mode = Default
java.lang.IllegalArgumentException: Chunk need to be bigger then 1 -!
IA>>print Condense(0,2,SUM)(x) #mode = Default
[
  [2,4],
  [10,12]
-]
#Get chunks of two items along dimension 0:
#First chunk: SUM(Int2d([[0,1],[2,3]]),0) = [2,4]
#Second chunk: SUM(Int2d([[4,5],[6,7]]),0) = [10,12]
IA>>print Condense(0,3,SUM)(x) #mode = Default
[
  [6,9]
-]
#Get chunks of three items along dimension 0:
#First chunk: SUM(Int2d([[0,1],[2,3],[4,5]]),0) = [6,9]
#Second chunk: [6,7] skipped
IA>>print Condense(0,4,SUM)(x) #mode = Default
[
  [12,16]
-]
#Get chunks of four items along dimension 0:
#First chunk: SUM(Int2d([[0,1],[2,3],[4,5],[6,7]]),0) = [12,16]
IA>>print Condense(0,2,SUM,"boxcar")(x)
[
```

Example 1: Untitled

```

    [2,4],
    [6,8],
    [10,12],
    [6,7]
  -]
  #Get chunks of two items along dimension 0 => one item after current
  position:
  #First  chunk: SUM(Int2d([[0,1],[2,3]]),0) = [2,4]
  #Second chunk: SUM(Int2d([[2,3],[4,5]]),0) = [6,8]
  #Third  chunk: SUM(Int2d([[4,5],[6,7]]),0) = [10,12]
  #Fourth chunk: SUM(Int2d([[6,7]]),0) = [10,12]
  IA>>print Condense(0,3,SUM,'boxcar')(Int2d([[0,1],[2,3],[4,5],[6,7],[8,9],
  [10,11],[12,13]]))
  [
    [2,4],
    [6,9],
    [12,15],
    [18,21],
    [24,27],
    [30,33],
    [22,24]
  -]
  #Get chunks of three items along dimension 0 => one item before current
  position, one item
  #after current position:
  #First  chunk: SUM (Int2d([[0,1],[2,3]]),0) = [2,4]
  #Second chunk: SUM (Int2d([[0,1],[2,3],[4,5]]),0) = [6,9]
  #Third  chunk: SUM (Int2d([[2,3],[4,5],[6,7]]),0) = [12,15]
  #Fourth chunk: SUM (Int2d([[4,5],[6,7],[8,9]]),0) = [18,21]
  #Fifth  chunk: SUM (Int2d([[6,7],[8,9],[10,11]]),0) = [24,27]
  #Sixth  chunk: SUM (Int2d([[8,9],[10,11],[12,13]]),0) = [30,33]
  #Seventh chunk: SUM (Int2d([[10,11],[12,13]]),0) = [22,24]
  IA>>print Condense(0,4,SUM,'boxcar')(Int2d([[0,1],[2,3],[4,5],[6,7],[8,9],
  [10,11],[12,13]]))
  [
    [6,9],
    [12,16],
    [20,24],
    [28,32],
    [36,40],
    [18,20],
    [22,24]
  -]
  #Get chunks of four items along dimension 0 => one item before current
  position, two items
  #after current position:
  #First  chunk: SUM (Int2d([[0,1],[2,3],[4,5]]),0) = [6,9]
  #Second chunk: SUM (Int2d([[0,1],[2,3],[4,5],[6,7]]),0) = [12,16]
  #Third  chunk: SUM (Int2d([[2,3],[4,5],[6,7],[8,9]]),0) = [20,24]
  #Fourth chunk: SUM (Int2d([[4,5],[6,7],[8,9],[10,11]]),0) = [28,32]
  #Fifth  chunk: SUM (Int2d([[6,7],[8,9],[10,11],[12,13]]),0) = [36,40]
  #Sixth  chunk: SUM (Int2d([[8,9],[10,11]]),0) = [18,20]
  #Seventh chunk: SUM (Int2d([[10,11],[12,13]]),0) = [22,24]
  </pre>

```

API Summary

Jython Syntax

```
outArray = Condense ( alongDimension, chunkSize, function [,mode])
(inArray)
```

Properties

float or double array **inArray** [INPUT, MANDATORY, default=p]

integer **alongDimension** [INPUT, MANDATORY, default=p]

Properties
<code>int chunkSize [INPUT, MANDATORY, default=p]</code>
<code>ArrayReductor ArrayToNumber function [INPUT, MANDATORY, default=p]</code>
<code>OPTIONAL mode [INPUT, default = false, default=no default value]</code>
<code>float or double array outArray [OUTPUT, MANDATORY, default=no default value]</code>

API details

Properties

<code>float or double array inArray [INPUT, MANDATORY, default=p]</code>
Input array of n dimensions
<code>integer alongDimension [INPUT, MANDATORY, default=p]</code>
Dimension in which the function will be applied (starting on zero)
<code>int chunkSize [INPUT, MANDATORY, default=p]</code>
Size of chunks to process. See boxcar parameter.
<code>ArrayReductor ArrayToNumber function [INPUT, MANDATORY, default=p]</code>
Function to apply
<code>OPTIONAL mode [INPUT, default = false, default=no default value]</code>
<p>Selection of special mode. Currently "default" or "boxcar"</p> <p>If this argument is specified, the mode is set to "boxcar" (it does not matter the string you write). If this argument is not specified, the mode is set to "default".</p> <ul style="list-style-type: none"> • "default": the array is splitted into arrays of the size specified by 'chunkSize' (along the specified dimension). The remaining items (that cannot fit in a chunk) are ignored. 'chunkSize' 1 is not allowed. • "boxcar" : chunks are created using a "box" centered in the current item. The process goes item by item along the specified dimension: <ul style="list-style-type: none"> • 'chunkSize' = 1: not allowed. • 'chunkSize' < 1: <ul style="list-style-type: none"> • Odd 'chunkSize': chunk is compound of the item in the current position, (chunkSize/2) item(s) before the current position and (chunkSize/2) item(s) after the current position. (Division is integer division) • Even 'chunkSize': chunk is compound of the item in the current position, ((chunkSize/2)-1) item(s) before the current position and (chunkSize/2) item(s) after the current position. (Division is integer division) <p>Example: 'chunkSize' = 3: (along the specified dimension) one item before current position, the current item and another item before current position. 'chunkSize' = 4: one item before current position, the current item and two items after current position.</p>

OPTIONAL mode [INPUT, default = false, default=no default value]


(See examples.)

float or double array outArray [OUTPUT, MANDATORY, default=no default value]

Return an array where the function 'function' was applied on data chunks 'chunkSize' in dimension 'alongDimension'

The center of the box is always the actual index. in case of an "even" box the value before the theoretical center (see boxcar parameter)

2.69. ConjugateGradientFitter

Full Name:	herschel.ia.numeric.toolbox.fit.ConjugateGradientFitter
Type:	Java Class - 
Import:	from herschel.ia.numeric.toolbox.fit import ConjugateGradientFitter

Description

Non-linear fitter using the conjugate gradient method.

This class is a wrapper around the ConjugateGradientMethod which integrates it into the Fitter concept.

The ConjugateGradientFitter (CGF) does not use matrix inversions to iterate to the solution (as LevenbergMarquardt). It calculates the gradient to the (local) ChiSq function and proceeds along that direction until it encounters a minimum. From that new position it iterates further.

This behaviour makes it fit for solving problems with many (>10 or so) parameters.

For documentation on the various options see Numerical Recipes Cpt 10.6.

Example


Example 1: ConjugateGradientFitter (for non-linear models)

```
# assume x and y are Double1d data arrays.
gauss = GaussModel( -)          # Gaussian
gauss += PolynomialModel( 1 -)  # add linear background
print gauss.getNumberOfParameters() # 5 (= 3 for Gauss + 2 for line)
cgfit = ConjugateGradientFitter( x, gauss -)
param = cgfit.fit( y -)
print param.length()           # 5
stdev = cgfit.getStandardDeviation() # stdevs on the parameters
chisq = cgfit.getChiSquared()
scale = cgfit.getScale()        # noise scale
yfit = cgfit.getResult()        # fitted values
yband = cgfit.monteCarloError() # 1 sigma confidence region
```

Limitations

1. CGF is **not** guaranteed to find the global minimum.
2. CGF does **not** work with fixed parameters or limits

2.70. ConnectorBox

Full Name:	herschel.ia.dataflow.ConnectorBox
Type:	Java Class - 
Import:	from herschel.ia.dataflow import ConnectorBox

Description

Non-process components as processes.

This class allows non-process components to be managed by dataflow as they were processes. The non-process component will have a ConnectorBox as member variable and the developer can call create* methods. The non-process component must implement Connectable interface and in its getProcess() method will return the connector box.

Example

Example 1: how to create a ConnectorBox

```
<pre>
public class MyComponent extends JFrame implement Connectable,
ProcessInputListener {
    private ConnectorBox _conn_box;
    // implementation of Connectable getProcess method
    public IaProcess getProcess() { return _conn_box; -}
    public MyComponent( String name_as_component, String name_as_process -) {
        super(name_as_component);
        _conn_box = new ConnectorBox( name_as_process -);
        ProcessInput input = _conn_box.createInput( -"input", Product.class -);
        input.addProcessInputListener( this -);
    -}
    // implementation of ProcessInputListener handle method
    public void handle( ProcessInputEvent ev -) {
        -// do stuff with the product that comes inside ev.
    -}
}
</pre>
```

Limitations

no limitation

See also

- [reference](#)

History

- 26-5-2005 jcg: change javadoc format for help support.

2.71. Contour

Full Name:	herschel.ia.dataset.image.Contour
Type:	Java Class - 
Import:	from herschel.ia.dataset.image import Contour

Description

A class to deal with Contours.

A class to deal with Contours.

API Summary

Constructor
<p>Contour(Double2d contourPoints)</p> <p>This constructor creates a new Contour with the given contour points.</p>
Methods
<p>int getNbOfContourPoints()</p> <p>Returns the number of contour points for this Contour.</p>
<p>convert(Wcs wcsOld, Wcs wcsNew)</p> <p>Conversion from one Wcs to another.</p>

API details

Constructor

<code>Contour(Double2d contourPoints)</code>
<p>This constructor creates a new Contour with the given contour points.</p> <p>This constructor creates a new Contour and sets the data (contour points) for the new Contour to the given data (contour points).</p>
<p>Argument</p> <p><code>Double2d contourPoints</code> [INPUT, MANDATORY]</p>

Methods

<code>int getNbOfContourPoints()</code>
<p>Returns the number of contour points for this Contour.</p> <p>Returns the number of contour points for this Contour.</p>
<p>Return</p> <p><code>int</code></p> <p>Returns the number of contour points for this Contour.</p>


convert([Wcs](#) wcsOld, [Wcs](#) wcsNew)

Conversion from one Wcs to another.

Converts the pixel coordinates of the contour points of this Contour from one Wcs to pixel coordinates to pixel coordinates in another Wcs.

Arguments[Wcs](#) **wcsOld** [INPUT, MANDATORY][Wcs](#) **wcsNew** [INPUT, MANDATORY]

2.72. ContourLevel

Full Name:	herschel.ia.dataset.image.ContourLevel
Type:	Java Class - 
Import:	from herschel.ia.dataset.image import ContourLevel

Description

A class to deal with ContourLevels.

A class to deal with ContourLevels.

API Summary

Constructors
ContourLevel(double contourValue) Creation of a new ContourLevel.
ContourLevel(double contourValue, Color color) Creation of a new ContourLevel.
Methods
setColor(Color color) Sets the color for this ContourLevel.
Color getColor() Returns the color for this ContourLevel.
addContour(Contour contour) Adds the given Contour to this ContourLevel.

API details

Constructors

ContourLevel(double contourValue) Creation of a new ContourLevel. This constructor creates a new ContourLevel for the given contour value. Argument double contourValue [INPUT, MANDATORY]
ContourLevel(double contourValue, Color color) Creation of a new ContourLevel. This constructor creates a new ContourLevel for the given contour value and with the given color. Arguments double contourValue [INPUT, MANDATORY] Color color [INPUT, MANDATORY]

Methods

setColor([Color](#) color)

Sets the color for this ContourLevel.

Sets the given color as the color for this ContourLevel.

Argument

[Color](#) color [INPUT, MANDATORY]

[Color](#) getColor()

Returns the color for this ContourLevel.

Returns the color for this ContourLevel.

Return

[Color](#)

Returns the color for this ContourLevel.

addContour([Contour](#) contour)


Adds the given Contour to this ContourLevel.

Adds the given Contour to this ContourLevel.

Argument

[Contour](#) contour [INPUT, MANDATORY]

2.73. ContourPlotting

Full Name:	herschel.ia.toolbox.image.gui.ContourPlotting
Type:	Java Class - 
Import:	from herschel.ia.toolbox.image.gui import ContourPlotting

Description

An implementation of contour plotting.

API Summary

Constructors
ContourPlotting(ImageAnalysisIToolbox toolbox) The standard constructor for ContourPlotting. This constructor creates a window
ContourPlotting(boolean useAsComponent, ImageAnalysisIToolbox toolbox) Constructor for ContourPlotting. When the new ContourPlotting is

Methods
int getUsedContourValues() Returns the used type of contour values for this ContourPlotting
int getUsedContourLevelRange() Returns the used type of contour level range for this
String getErrorMessage() Returns an appropriate error message for the given input. If no
ArrayList getContourValues() Returns a list with all contour values for this ContourPlotting.
ArrayList getContourColors() Returns the colors that should be used for the contours of the
int getNumberOfContourLevels() Returns the number of contour levels for this ContourPlotting if
double[] getContourLevels() Returns the used lower and upper contour value for this
double getMinimumContourLength() Returns the minimum length for contours to be drawn on the image.
String getUsedDistribution() Returns the type of distribution of the contour levels, that is
ArrayList getImageFigures() Returns all ImageFigures used for this ContourPlotting.

API details

Constructors

ContourPlotting([ImageAnalysisToolbox](#) toolbox)

The standard constructor for ContourPlotting. This constructor creates a window

in which you must give the input parameters, needed to perform contour plotting on the image, analyzed with the given ImageAnalysisToolbox.

Argument

[ImageAnalysisToolbox](#) **toolbox** [INPUT, MANDATORY]

ContourPlotting(boolean useAsComponent, [ImageAnalysisToolbox](#) toolbox)

Constructor for ContourPlotting. When the new ContourPlotting is

component-based, a window for contour plotting is opened; otherwise nothing will be shown.

Arguments

boolean **useAsComponent** [INPUT, MANDATORY]

[ImageAnalysisToolbox](#) **toolbox** [INPUT, MANDATORY]

Methods

int [getUsedContourValues\(\)](#)

Returns the used type of contour values for this ContourPlotting

(manual or automatic).

Return

int

Returns the used type of contour values for this ContourPlotting (manual or automatic).

int [getUsedContourLevelRange\(\)](#)

Returns the used type of contour level range for this

ContourPlotting (manual or image cut levels).

Return

int

Returns the used type of contour level range for this ContourPlotting (manual or image cut levels).

String [getErrorMessage\(\)](#)

Returns an appropriate error message for the given input. If no

errors occur in the input, null is returned.

Return

[String](#)

String `getErrorMessage()`

Returns an appropriate error message for the given input. If no errors occur in the input, null is returned.

ArrayList `getContourValues()`

Returns a list with all contour values for this ContourPlotting.

Return

[ArrayList](#)

Returns a list with all contour values for this ContourPlotting.

ArrayList `getContourColors()`

Returns the colors that should be used for the contours of the different contour values.

Return

[ArrayList](#)

Returns the colors that should be used for the contours of the different contour values.

int `getNumberOfContourLevels()`

Returns the number of contour levels for this ContourPlotting if the contour values are to be calculated automatically; otherwise -1 is returned.

Return

int

Returns the number of contour levels for this ContourPlotting if the contour values are to be calculated automatically; otherwise -1 is returned.

double[] `getContourLevels()`

Returns the used lower and upper contour value for this ContourPlotting (manual or image cut levels).

Return

double[]

The used lower and upper contour value for this ContourPlotting (manual or image cut levels).

double `getMinimumContourLength()`

Returns the minimum length for contours to be drawn on the image.

Return

double

Returns the minimum length for contours to be drawn on the image.

String `getUsedDistribution()`

Returns the type of distribution of the contour levels, that is

[String](#) `getUsedDistribution()`

used for this ContourPlotting (linear, log or ln).

Return**[String](#)**

Returns the type of distribution of the contour levels, that is used for this ContourPlotting (linear, log or ln).


[ArrayList](#) `getImageFigures()`

Returns all ImageFigures used for this ContourPlotting.

Return**[ArrayList](#)**

Returns all ImageFigures used for this ContourPlotting.

2.74. ContourTask

Full Name:	herschel.ia.toolbox.image.ContourTask
Type:	Java Task - 
Import:	from herschel.ia.toolbox.image import ContourTask

Description

A Task for making contours for a given contour value.

A Task for making contours for a given contour value and that is called withing AutomaticContourTask and ManualContourTask.

API Summary


Properties
Image image [INPUT, MANDATORY, default=No default value]
Double value [INPUT, MANDATORY, default=No default value]
Color color [INPUT, OPTIONAL, default=No default value]
ImageContour contours [OUTPUT, MANDATORY, default=No default value]

API details

Properties

Image image [INPUT, MANDATORY, default=No default value]
The image.
Double value [INPUT, MANDATORY, default=No default value]
The contour value.
Color color [INPUT, OPTIONAL, default=No default value]
The color to visualise the contours.
ImageContour contours [OUTPUT, MANDATORY, default=No default value]
The image contour.

2.75. ConvertImageUnitTask

Full Name:	herschel.ia.toolbox.image.ConvertImageUnitTask
Type:	Java Task - 
Import:	from herschel.ia.toolbox.image import ConvertImageUnitTask

Description

A Task for converting the units of an image from surface brightness to another surface brightness.

A Task for converting the units of an image from surface brightness to another surface brightness.

API Summary


Properties
Image image [INPUT, MANDATORY, default=No default value]
Unit newUnit [INPUT, MANDATORY, default=No default value]
float beamArea [INPUT, OPTIONAL, default=0.0]
Image convertedImage [OUTPUT, MANDATORY, default=No default value]

API details

Properties

Image image [INPUT, MANDATORY, default=No default value]
The image.
Unit newUnit [INPUT, MANDATORY, default=No default value]
The new unit for the image.
float beamArea [INPUT, OPTIONAL, default=0.0]
The beam area in square arc seconds.
Image convertedImage [OUTPUT, MANDATORY, default=No default value]
The converted image.

2.76. ConvertImageUnitTaskSignatureComponent

Full Name:	herschel.ia.toolbox.image.gui.ConvertImageUnitTaskSignatureComponent
Type:	Java Class - 
Import:	from herschel.ia.toolbox.image.gui import ConvertImageUnitTaskSignatureComponent

Description

The task dialog for the ConvertImageUnitTask.

A task dialog to serve as GUI for the ConvertImageUnitTask.

API Summary

Constructor
ConvertImageUnitTaskSignatureComponent() The construction of a new ConvertImageUnitTaskSignatureComponent.
Methods
check() Validates the current modifications.
clear() Clears the current modifications.
getParameters() Returns a map of parameters and modifiers storing the value selected by the user.
setSignature(SignatureApi signature) Setting the signature.
setVariableSelection(VariableSelection selection) Setting the variable selection.
JLabel newLabel(String title, String tooltip) Returns a label with the given title and tooltip.
JComponent getComponent() Returns the component.

API details

Constructor

ConvertImageUnitTaskSignatureComponent()
The construction of a new ConvertImageUnitTaskSignatureComponent.
The construction of a new ConvertImageUnitTaskSignatureComponent.

Methods

check()
Validates the current modifications. Validates the current modifications for this ConvertImageUnitTaskSignatureComponent.
clear()
Clears the current modifications. Clears the current modifications for this ConvertImageUnitTaskSignatureComponent.
getParameters()
Returns a map of parameters and modifiers storing the value selected by the user. Returns a map of parameters and modifiers storing the value selected by the user for this ConvertImageUnitTaskSignatureComponent.
setSignature(SignatureApi signature)
Setting the signature. Sets the given signature as the signature for this ConvertImageUnitTaskSignatureComponent. Argument SignatureApi signature [INPUT, MANDATORY]
setVariableSelection(VariableSelection selection)
Setting the variable selection. Sets the given selection as the variable selection for this ConvertImageUnitTaskSignatureComponent. Argument VariableSelection selection [INPUT, MANDATORY]
JLabel newLabel(String title, String tooltip)
Returns a label with the given title and tooltip. Returns a label with the given title and tooltip. Arguments String title [INPUT, MANDATORY] String tooltip [INPUT, MANDATORY] Return JLabel Returns a label with the given title and tooltip.
JComponent getComponent()
Returns the component.

[JComponent](#) `getComponent()`


Returns the component for this ConvertImageUnitTaskSignatureComponent.

Return

[JComponent](#)

Returns the component for this ConvertImageUnitTaskSignatureComponent.

2.77. Convolution

Full Name:	herschel.ia.numeric.toolbox.filter.Convolution
Alias:	Convolution
Type:	Java Class - 
Import:	from herschel.ia.numeric.toolbox.filter import Convolution

Description

Creates a Convolution filter, that can be applied to numeric arrays of rank 1.

Example

Example 1: Apply Convolution on a Int1d

```
x=Int1d([1,3,2,4,6,5])
f=Convolution( Double1d([1,3,2]) -)
print f(x) # [1.0,6.0,13.0,16.0,22.0,31.0]
```

API Summary

Jython Syntax

```
<f>=Convolution(<kernel> [, <center>=true|false]
[, <edge>=Convolution.ZEROES|CIRCULAR|REPEAT])
<x>=<f>(<x>)
```

Properties

[Double1d kernel](#) [INPUT, MANDATORY, default=no default value]

[boolean center](#) [INPUT, NOT_MANDATORY, default=false]

[Convolution.ZEROES|CIRCULAR|REPEAT center](#) [INPUT, NOT_MANDATORY, default=Convolution.ZEROES]

API details

Properties

[Double1d kernel](#) [INPUT, MANDATORY, default=no default value]

It must be a Double1d array, when =true, the kernel should have an odd number of elements.

[boolean center](#) [INPUT, NOT_MANDATORY, default=false]

Set center to true or false.

[Convolution.ZEROES|CIRCULAR|REPEAT center](#) [INPUT, NOT_MANDATORY, default=Convolution.ZEROES]

Set edge to true or false

- `edge=Convolution.ZEROES`: Set result to zero at edges.
- `edge=Convolution.CIRCULAR`: Wrap around at edges (circular convolution).

```
Convolution.ZEROES | CIRCULAR | REPEAT center [INPUT, NOT_MANDATORY,  
default=Convolution.ZEROES]
```

- edge=Convolution.REPEAT: Repeat the edge values of input array.


See also

- [BoxCarFilter](#)
- [GaussianFilter](#)

History

- 23 Apr 2008 AS Modify default for center parameter; default is center=True.

2.78. Correlate

Full Name:	herschel.ia.numeric.toolbox.basic.Correlate
Alias:	Correlate
Type:	Java Class - 
Import:	from herschel.ia.numeric.toolbox.basic import Correlate

Description

Yields the linear Pearson correlation coefficient of the input arrays.

If one vector is longer than the other, only the values up to the length of the shortest vector will be taken into account.

Example

Example 1: Correlation of two vectors of long

```
x = Long1d([65, 63, 67, 64, 68, 62, 70, 66, 68, 67, 69, 71])
y = Long1d([68, 66, 68, 65, 69, 66, 68, 65, 71, 67, 68, 70])
print Correlate(x)(y) # 0.7026516450800809
```

API Summary

Jython Syntax

```
<r>=Correlate(<x>)(<y>)
```

Properties

[any ordered 1d array **x** \[INPUT, MANDATORY, default=no default value\]](#)

[any ordered 1d array **y** \[INPUT, MANDATORY, default=no default value\]](#)

[double **r** \[OUTPUT, MANDATORY, default=no default value\]](#)

API details

Properties

any ordered 1d array **x [INPUT, MANDATORY, default=no default value]**

May be an integral or floating-point array; the former implicitly transformed to a double array.

any ordered 1d array **y [INPUT, MANDATORY, default=no default value]**

May be an integral or floating-point array; the former implicitly transformed to a double array.


double **r [OUTPUT, MANDATORY, default=no default value]**

Returns the linear Pearson correlation coefficient of the input arrays.

See also

- [CorrelateMatrix](#)

2.79. CorrelateMatrix

Full Name:	herschel.ia.numeric.toolbox.basic.CorrelateMatrix
Alias:	CorrelateMatrix
Type:	Java Class - 
Import:	from herschel.ia.numeric.toolbox.basic import CorrelateMatrix

Description

Yields the linear Pearson correlation coefficients of the input M x N matrix.

The result is a N x N matrix with each i, j value equal to the correlation coefficient of the i and j columns of the original matrix.

Example

Example 1: Apply Correlation of a matrix of integers

```
m = Int2d([ [65, 67], [64, 68], [67, 71] -])
print CorrelateMatrix()(m)
# [ [1.0, 0.8386278693775344], [0.8386278693775344, 1.0] -]
```

API Summary

Jython Syntax

```
<r>=CorrelateMatrix()( <m> )
```

Properties

[any ordered 2d array \(M rows x N columns\)](#) **m** [INPUT, MANDATORY, default=no default value]

[Double2d](#) **r** [OUTPUT, MANDATORY, default=no default value]

API details

Properties

any ordered 2d array (M rows x N columns) m [INPUT, MANDATORY, default=no default value]

May be an integral or floating-point array; the former implicitly transformed to a double array.


Double2d r [OUTPUT, MANDATORY, default=no default value]

Returns the N x N matrix with the linear Pearson correlation coefficients of the input matrix.

See also

- [Correlate](#)

2.80. COS

Full Name:	herschel.ia.numeric.toolbox.basic.Cos
Alias:	COS
Type:	Java Class - 
Import:	from herschel.ia.numeric.toolbox.basic import Cos

Description

Computes the trigonometric cosine of an number or array

Gives the cosine of a number or array. The input is in radians, and may be of any type. Returns a float array for float input array, complex array for input complex array, and double array for any other numeric array type.

Example

Example 1: Apply COS on a Float1d
<pre>x=Float1d([0,0.5]) print COS(x) # [1.0,0.87758255]</pre>

API Summary

Jython Syntax
<y>=COS(<x>)
Properties
any type x [INPUT, MANDATORY, default=no default value]
any type y [OUTPUT, NOT_MANDATORY, default=no default value]

API details


Properties

any type x [INPUT, MANDATORY, default=no default value]
The input is in radians, and may be of any type.
any type y [OUTPUT, NOT_MANDATORY, default=no default value]
Returns a float array for float input array, complex array for input complex array, and double array for any other numeric array type.

See also

- [ARCCOS](#)
- [COSH](#)
- [SIN](#)
- [TAN](#)

2.81. COSH

Full Name:	herschel.ia.numeric.toolbox.basic.CosH
Alias:	COSH
Type:	Java Class - 
Import:	from herschel.ia.numeric.toolbox.basic import CosH

Description

Computes the trigonometric hyperbolic cosine of an number or array

Gives the hyperbolic cosine of a number or array. The input is in radians, and may be of any type. Returns a float array for float input array, complex array for input complex array, and double array for any other numeric array type.

API Summary

Jython Syntax
<code><y>=COSH (<x>)</code>

Properties
<code>any type x [INPUT, MANDATORY, default=no default value]</code>
<code>any type y [OUTPUT, NOT_MANDATORY, default=no default value]</code>

Miscellaneous

Does not work for complex values.

API details

Properties


<code>any type x [INPUT, MANDATORY, default=no default value]</code>
The input is in radians, and may be of any type.

<code>any type y [OUTPUT, NOT_MANDATORY, default=no default value]</code>
Returns a float array for float input array, complex array for input complex array, and double array for any other numeric array type.

See also

- [COS](#)
- [ARCCOS](#)

2.82. CreateRgbImageTask

Full Name:	herschel.ia.toolbox.image.CreateRgbImageTask
Type:	Java Task - 
Import:	from herschel.ia.toolbox.image import CreateRgbImageTask

Description

A Task to combine three images to an RGB image.

A Task to combine three images of the same region on the sky, but in different filters, to an RGB image.

API Summary

Properties
Image red [INPUT, MANDATORY, default=No default value]
Image green [INPUT, MANDATORY, default=No default value]
Image blue [INPUT, MANDATORY, default=No default value]
Double percent [INPUT, MANDATORY, default=Default value : 98.0]
Double lowRed [INPUT, OPTIONAL, default=No default value]
Double highRed [INPUT, OPTIONAL, default=No default value]
Double lowGreen [INPUT, OPTIONAL, default=No default value]
Double highGreen [INPUT, OPTIONAL, default=No default value]
Double lowBlue [INPUT, OPTIONAL, default=No default value]
Double highBlue [INPUT, OPTIONAL, default=No default value]
RgbSimpleImage rgb [OUTPUT, MANDATORY, default=No default value]

API details

Properties

Image red [INPUT, MANDATORY, default=No default value]
The red input image.
Image green [INPUT, MANDATORY, default=No default value]
The green input image.
Image blue [INPUT, MANDATORY, default=No default value]
The blue input image.
Double percent [INPUT, MANDATORY, default=Default value : 98.0]
The percentage of values to include.
Double lowRed [INPUT, OPTIONAL, default=No default value]
The low cut level for the red image.

Double highRed [INPUT, OPTIONAL, default=No default value]

The high cut level for the red image.

Double lowGreen [INPUT, OPTIONAL, default=No default value]

The low cut level for the green image.

Double highGreen [INPUT, OPTIONAL, default=No default value]

The high cut level for the green image.

Double lowBlue [INPUT, OPTIONAL, default=No default value]

The low cut level for the blue image.


Double highBlue [INPUT, OPTIONAL, default=No default value]

The high cut level for the blue image.

RgbSimpleImage rgb [OUTPUT, MANDATORY, default=No default value]

The output RGB image.

2.83. CreateRgbImageTaskSignatureComponent

Full Name:	herschel.ia.toolbox.image.gui.CreateRgbImageTaskSignatureComponent
Type:	Java Class - 
Import:	from herschel.ia.toolbox.image.gui import CreateRgbImageTaskSignatureComponent

Description

The task dialog for the CreateRgbImageTask.

A task dialog to serve as GUI for the CreateRgbImageTask.

API Summary

Constructor	
CreateRgbImageTaskSignatureComponent()	The construction of a new CreateRgbImageTaskSignatureComponent.
Methods	
check()	Validates the current modifications.
clear()	Clears the current modifications.
Map getParameters()	Returns a map of parameters and modifiers storing the value selected by the user.
setSignature(SignatureApi signature)	Setting the signature.
setVariableSelection(VariableSelection selection)	Setting the variable selection.
getComponent()	Returns the component.

API details

Constructor


<code>CreateRgbImageTaskSignatureComponent()</code>
The construction of a new CreateRgbImageTaskSignatureComponent.
The construction of a new CreateRgbImageTaskSignatureComponent.

Methods

<code>check()</code>
Validates the current modifications.

check()
Validates the current modifications for this CreateRgbImageTaskSignatureComponent.
clear()
Clears the current modifications.
Clears the curent modifications for this CreateRgbImageTaskSignatureComponent.
Map getParameters()
Returns a map of parameters and modifiers storing the value selected by the user.
Returns a map of parameters and modifiers storing the value selected by the user for this CreateRgbImageTaskSignatureComponent.
Return
Map
Returns a map of parameters and modifiers storing the value selected by the user for this CreateRgbImageTaskSignatureComponent.
setSignature(SignatureApi signature)
Setting the signature.
Sets the given signature as the signature for this CreateRgbImageTaskSignatureComponent.
Argument
SignatureApi signature [INPUT, MANDATORY]
setVariableSelection(VariableSelection selection)
Setting the variable selection.
Sets the given selection as the variable selection for this CreateRgbImageTaskSignatureComponent.
Argument
VariableSelection selection [INPUT, MANDATORY]
getComponent()
Returns the component.
Returns the component for this CreateRgbImageTaskSignatureComponent.

2.84. crop

Full Name:	herschel.ia.toolbox.image.CropTask
Alias:	crop
Type:	Java Task - 
Import:	from herschel.ia.toolbox.image import CropTask
Category:	task/image

Description

A Task to crop images and cubes.

A Task to crop images and cubes to a specified rectangular area. The upper left pixel and the lower right pixel should be given.

Example

Example 1: Cropping an image between x-coordinate 11 and 240, and between y-coordinates 240 and 300

```
crop(image = image, x1 = 11, x2 = 55, y1 = 240, y2 = 300)
```

API Summary

Jython Syntax

```
crop(image, 11, 240, 55, 300)
```

Properties

`Image image` [INPUT, MANDATORY, default=No default value]

`int row1` [INPUT, MANDATORY, default=No default value]

`int column1` [INPUT, MANDATORY, default=No default value]

`int row2` [INPUT, OPTIONAL, default=No default value]

`int column2` [INPUT, MANDATORY, default=No default value]

API details

Properties

`Image image` [INPUT, MANDATORY, default=No default value]

The Image to be cropped

`int row1` [INPUT, MANDATORY, default=No default value]

The x coordinate of the left upper pixel of the rectangle to crop

`int column1` [INPUT, MANDATORY, default=No default value]

The y coordinate of the left upper pixel of the rectangle to crop


`int row2` [INPUT, OPTIONAL, default=No default value]

The x coordinate of the lower right pixel of the rectangle to crop

<code>int column2 [INPUT, MANDATORY, default=No default value]</code>

The y coordinate of the lower right pixel of the rectangle to crop
--

2.85. CrossCorrelationTask

Full Name:	herschel.ia.toolbox.image.CrossCorrelationTask
Type:	Java Task - 
Import:	from herschel.ia.toolbox.image import CrossCorrelationTask

Description

A Task to calculate the linear Pearson correlation coefficient of two images.

A Task to calculate the linear Pearson correlation coefficient of two images. We start from a $M \times N$ image (M = number of rows, N = number of columns) and calculate the $N \times N$ linear Pearson coefficient matrix. The i 'th row and j 'th column element corresponds to the correlation of the i 'th and j 'th columns of the $M \times N$ matrix.

API Summary


Properties
Image <code>image1</code> [INPUT, MANDATORY, default=No default value]
Image <code>image2</code> [INPUT, MANDATORY, default=No default value]
paramType <code>correlation</code> [OUTPUT, MANDATORY, default=No default value]

API details

Properties

Image <code>image1</code> [INPUT, MANDATORY, default=No default value]
The first image.
Image <code>image2</code> [INPUT, MANDATORY, default=No default value]
The second image.
paramType <code>correlation</code> [OUTPUT, MANDATORY, default=No default value]
The correlation.

2.86. Cube3Dviewer

Full Name:	herschel.ia.gui.cube.Cube3Dviewer
Type:	Java Class - 
Import:	from herschel.ia.gui.cube import Cube3Dviewer

Description

A Test Gui for a first real 3D viewer for spectral cubes. demonstrator jsut to see if the concept is interesting

API Summary

Constructor
Cube3Dviewer() The standard constructor for CubeSpectrumAnalysisToolbox. This
Methods
SaveCurrentImage() Makes a menubar (Image, Help) for the Jpanel version this CubeSpectrumAnalysisToolbox.
setCube(SpectralSimpleCube spectralsimplecube)
exit() Exits this CubeSpectrumAnalysisToolbox (closes (the window of) this
closeActiveTab() Closes the active tab, if there is one.
closeAllTabs() Closes all tabs, if there are any.
setEnabled(boolean enabled) Disables/enables all tabs on the JTabbedPane ans the menus for
boolean isEnabled() Returns whether the JTabbedPane and menus for closing tabs and
DoubleId getWaves() Method that returns the FloatId array of the wavelength of this CubeSpectrumAnalysisToolbox.
boolean isInHipe() Method that returns the way the CubeSpectrumAnalysisToolbox was opened.
SimpleCube getSimpleCube() Method that returns the FloatId array of the wavelength of this CubeSpectrumAnalysisToolbox.
SpectralSimpleCube getSpectralSimpleCube() Method that returns the FloatId array of the wavelength of this CubeSpectrumAnalysisToolbox.
JFrame getFrame() Method that returns the JFrame of this CubeSpectrumAnalysisToolbox.

Methods
JPanel <code>getPanel()</code> Method that returns the JPanel of this CubeSpectrumAnalysisToolbox.
SimpleImage <code>getSimpleImage()</code> Returns the displayed image (SimpleImage).
Display <code>getDisplay()</code> Returns the Display of the Cube you are analysing with
Double <code>getMaxValue()</code> Returns the maximal value of the "image" of the cube
Double <code>getMinValue()</code> Returns the maximal value of the "image" of the cube
String <code>getCubeName()</code> returns the name of the cube
setCubeName(String name) set the name of the cube
boolean <code>isInImage(MouseEvent e)</code> Checks whether the given MouseEvent has occurred within the image
boolean <code>isInImage(double pixX, double pixY)</code> Checks whether the point with given PixelCoordinates (pixX, pixY)
boolean <code>hasWCS()</code> Returns true if SkyCoordinates are available for the image,
ArrayList <code>getAllFigures()</code> Returns all ImageFigures used for this CubeSpectrumAnalysisToolbox.
updateImage() Updates the image, when another tab is made active.

API details

Constructor

<code>Cube3Dviewer()</code>
The standard constructor for CubeSpectrumAnalysisToolbox. This constructor shows an empty CubeSpectrumAnalysisToolbox window in 2 parts, with only a title, buttons for maximizing, minimizing and closing the window, a menubar (File, Image, Help) and a colorbar and a statusbar at the bottom.

Methods

<code>SaveCurrentImage()</code>
Makes a menubar (Image, Help) for the JPanel version this CubeSpectrumAnalysisToolbox.
<code>setCube(SpectralSimpleCube spectralsimplecube)</code>
Argument SpectralSimpleCube <code>spectralsimplecube</code> [INPUT, MANDATORY]

exit()
Exits this CubeSpectrumAnalysisToolbox (closes (the window of) this CubeSpectrumAnalysisToolbox).
closeActiveTab()
Closes the active tab, if there is one.
closeAllTabs()
Closes all tabs, if there are any.
setEnabled(boolean enabled)
Disables/enables all tabs on the JTabbedPane and the menus for closing tabs and performing image analysis, of this CubeSpectrumAnalysisToolbox.
Argument
boolean enabled [INPUT, MANDATORY]
boolean isEnabled()
Returns whether the JTabbedPane and menus for closing tabs and performing image analysis, of this CubeSpectrumAnalysisToolbox are enabled/disabled.
Return
boolean
Returns true is the JTabbedPane and menus for closing tabs and performing image analysis, of this CubeSpectrumAnalysisToolbox are enabled; false otherwise.
Double1d getWaves()
Method that returns the Float1d array of the wavelength of this CubeSpectrumAnalysisToolbox.
Return
Double1d
The Float1d array of the wavelength of this CubeSpectrumAnalysisToolbox.
boolean isInHipe()
Method that returns the way the CubeSpectrumAnalysisToolbox was opened.
true if it's open with the right menu in the variable list of hipe false when opened with the classical command line call
Return
boolean
the way the CubeSpectrumAnalysisToolbox was opened. true if it's open with the right menu in the variable list of hipe false when opened with the classical command line call
SimpleCube getSimpleCube()
Method that returns the Float1d array of the wavelength of this CubeSpectrumAnalysisToolbox.

[SimpleCube](#) `getSimpleCube()`

Return

[SimpleCube](#)

The Float1d array of the wavelength of this CubeSpectrumAnalysisToolbox.

[SpectralSimpleCube](#) `getSpectralSimpleCube()`

Method that returns the Float1d array of the wavelength of this CubeSpectrumAnalysisToolbox.

Return

[SpectralSimpleCube](#)

The Float1d array of the wavelength of this CubeSpectrumAnalysisToolbox.

[JFrame](#) `getFrame()`

Method that returns the JFrame of this CubeSpectrumAnalysisToolbox.

Return

[JFrame](#)

The JFrame of this CubeSpectrumAnalysisToolbox.

[JPanel](#) `getPanel()`

Method that returns the JPanel of this CubeSpectrumAnalysisToolbox.

Return

[JPanel](#)

The JPanel of this CubeSpectrumAnalysisToolbox.

[SimpleImage](#) `getSimpleImage()`

Returns the displayed image (SimpleImage).

Return

[SimpleImage](#)

Returns the displayed image (SimpleImage).

[Display](#) `getDisplay()`

Returns the Display of the Cube you are analysing with this CubeSpectrumAnalysisToolbox.

Return

[Display](#)

Returns the display of the Cube you are analysing with this CubeSpectrumAnalysisToolbox.

[Double](#) `getMaxValue()`

Returns the maximal value of the "image" of the cube

Return

[Double](#)

Double getMaxValue()

Returns the maximal value of the "image" of the cube

Double getMinValue()

Returns the maximal value of the "image" of the cube

Return

Double

Returns the maximal value of the "image" of the cube

String getCubeName()

returns the name of the cube

Return

String

returns the name of the cube

setCubeName(String name)

set the name of the cube

Argument

String name [INPUT, MANDATORY]

boolean isInImage(MouseEvent e)

Checks whether the given MouseEvent has occurred within the image

we are analysing with this CubeSpectrumAnalysisToolbox.

Argument

MouseEvent e [INPUT, MANDATORY]

Return

boolean

Returns true if the given MouseEvent has occurred within the image that is being analysed with this CubeSpectrumAnalysisToolbox; false otherwise.

boolean isInImage(double pixX, double pixY)

Checks whether the point with given PixelCoordinates (pixX, pixY)

lies in the image we are analysing with this CubeSpectrumAnalysisToolbox.

Arguments

double **pixX** [INPUT, MANDATORY]

double **pixY** [INPUT, MANDATORY]

Return

boolean

Returns true if the point with given PixelCoordinates (pixX, pixY) lies in the image we are analysing with this CubeSpectrumAnalysisToolbox; false otherwise.

boolean hasWCS ()

Returns true if SkyCoordinates are available for the image, analysed with this CubeSpectrumAnalysisToolbox; false otherwise.

Return

boolean

Returns true if SkyCoordinates are available for the image, analysed with this CubeSpectrum-AnalysisToolbox; false otherwise.

[ArrayList](#) getAllFigures ()

Returns all ImageFigures used for this CubeSpectrumAnalysisToolbox.

Return

[ArrayList](#)

Returns all ImageFigures used for this CubeSpectrumAnalysisToolbox.


updateImage ()

Updates the image, when another tab is made active.

See also

- [Display](#)

2.87. CubeCompareDisplay

Full Name:	herschel.ia.gui.cube.CubeCompareDisplay
Type:	Java Class - 
Import:	from herschel.ia.gui.cube import CubeCompareDisplay

Description

A class to compare visually 2 spectralcubes when they share a same sky location

API Summary

Methods
setBegin(Double begin) Sets the begin of the straight line, associated with this
setEnd(Double end) Sets the end of the straight line, associated with this
ImageFigure getLine() Returns the straight line (ImageFigure), of which we wish to plot
Double getBegin() Returns the begin of the drawn line in PixelCoordinates.
Double getEnd() Returns the end of the drawn line in PixelCoordinates.
String getSkyCoordinates() Returns the String version of the sky coordinates
boolean isInImage(MouseEvent e) Checks whether the given MouseEvent has occurred within the image
boolean isInImage(double pixX, double pixY) Checks whether the point with given PixelCoordinates (pixX, pixY)

API details

Methods

setBegin(Double begin) Sets the begin of the straight line, associated with this Velocity position map to the given point (in UserCoordinates). Argument Double begin [INPUT , MANDATORY]
setEnd(Double end) Sets the end of the straight line, associated with this Velocity position map to the given point (in UserCoordinates). Argument

setEnd(Double end)

[Double](#) end [INPUT, MANDATORY]

ImageFigure getLine()

Returns the straight line (ImageFigure), of which we wish to plot the intensity in this CubeCompareDisplay.

Return

[ImageFigure](#)

The straight line (ImageFigure), of which we wish to plot the intensity in this CubeCompareDisplay.

Double getBegin()

Returns the begin of the drawn line in PixelCoordinates.

Return

[Double](#)

Returns the begin of the drawn line in PixelCoordinates.

Double getEnd()

Returns the end of the drawn line in PixelCoordinates.

Return

[Double](#)

Returns the end of the drawn line in PixelCoordinates.

String getSkyCoordinates()

Returns the String version of the sky coordinates (WorldCoordinates) of the begin and end of the drawn straight line.

Return

[String](#)

Returns the String version of the sky coordinates (WorldCoordinates) of the begin and end of the drawn straight line.

boolean isInImage(MouseEvent e)

Checks whether the given MouseEvent has occurred within the image we are analysing with this CubeSpectrumAnalysisToolbox.

Argument

[MouseEvent](#) e [INPUT, MANDATORY]

Return

boolean

Returns true if the given MouseEvent has occurred within the image that is being analysed with this CubeSpectrumAnalysisToolbox; false otherwise.

boolean isInImage(double pixX, double pixY)


Checks whether the point with given PixelCoordinates (pixX, pixY)

lies in the image we are analysing with this CubeSpectrumAnalysisToolbox.

Argumentsdouble **pixX** [INPUT, MANDATORY]double **pixY** [INPUT, MANDATORY]**Return****boolean**

Returns true if the point with given PixelCoordinates (pixX, pixY) lies in the image we are analysing with this CubeSpectrumAnalysisToolbox; false otherwise.

2.88. CubeSpectrumAnalysis

Full Name:	herschel.ia.gui.cube.CubeSpectrumAnalysis
Type:	Java Class - 
Import:	from herschel.ia.gui.cube import CubeSpectrumAnalysis

Description

CubeSpectrumAnalysis

A class for dealing with CubeSpectrumAnalysis (in the current version : SpectrumPlotting, Averaged Spectrum Plotting, Velocity map Channel Map).

API Summary

Constructor
<p>CubeSpectrumAnalysis(CubeSpectrumAnalysisToolbox toolbox, String title)</p> <p>Standard constructor for CubeSpectrumAnalysis. A new window with the</p>
Methods
<p>JFrame getFrame()</p> <p>Method that returns the JFrame of this CubeSpectrumAnalysis.</p>
<p>CubeSpectrumAnalysisToolbox getToolbox()</p> <p>Returns the CubeSpectrumAnalysisToolbox, associated with this</p>
<p>Container getComponent()</p> <p>Returns the content pane (Container) of the JFrame of this</p>
<p>ArrayList getImageFigures()</p> <p>Returns all ImageFigures used for this CubeSpectrumAnalysis.</p>

API details

Constructor

<p>CubeSpectrumAnalysis(CubeSpectrumAnalysisToolbox toolbox, String title)</p>
<p>Standard constructor for CubeSpectrumAnalysis. A new window with the given title and associated with the given CubeSpectrumAnalysisToolbox is created.</p>
<p>Arguments</p> <p>CubeSpectrumAnalysisToolbox toolbox [INPUT, MANDATORY]</p> <p>String title [INPUT, MANDATORY]</p>

Methods

<p>JFrame getFrame()</p>
<p>Method that returns the JFrame of this CubeSpectrumAnalysis.</p>

[JFrame](#) `getFrame()`

Return

[JFrame](#)

The JFrame of this CubeSpectrumAnalysis.

[CubeSpectrumAnalysisToolbox](#) `getToolbox()`

Returns the CubeSpectrumAnalysisToolbox, associated with this CubeSpectrumAnalysis.

Return

[CubeSpectrumAnalysisToolbox](#)

Returns the CubeSpectrumAnalysisToolbox, associated with this CubeSpectrumAnalysis.

[Container](#) `getComponent()`

Returns the content pane (Container) of the JFrame of this CubeSpectrumAnalysis.

Return

[Container](#)

Returns the content pane (Container) of the JFrame of this CubeSpectrumAnalysis.

[ArrayList](#) `getImageFigures()`


Returns all ImageFigures used for this CubeSpectrumAnalysis.

Return

[ArrayList](#)

Returns all ImageFigures used for this CubeSpectrumAnalysis.

2.89. CubeSpectrumAnalysisToolbox

Full Name:	herschel.ia.gui.cube.CubeSpectrumAnalysisToolbox
Type:	Java Class - 
Import:	from herschel.ia.gui.cube import CubeSpectrumAnalysisToolbox

Description

An implementation of a toolbox for spectrum analysis.

In the current version do the following :

1) You can extract a raw single pixel spectrum from a position selected when clicking on the mouse in the image. <br 2) You can extract an averaged spectrum on a region defined the image by drawing a shape with the mouse. - The circle selection use a linear approximation to compute the weight of the border pixels - The circle selection is "free floating"

5) You can construct a velocity position map, by drawing a line on the image. the velocities are computed from the different layers along the Z axis

6) From 15-09-08 this cube Spectrum Analysis toolbox is registered as a Viewer for the SimpleCube. this mean that in HIPE a simpleCube Variable can be automatically visualized with it . This toolbox is registered in HIPE on the SimpleCube and therefore can be launch via the right button menu In all cases the spectrum analysis is performed on the cube with the shown layer considered as the reference frame. The result is shown in a multi-tab section on the righthand side. All the features are using Task which can be launched from e JIDE environment.

A basic example on how to open a toolbox for spectrum analysis on an SimpleCube Cds or a couple of arrays (Double3d cube,Double1d wavelength)

or

A basic example on how to open a toolbox for spectrum analysis on an SimpleCube Cds or a couple of arrays (Double3d cube,Double1d wavelength)

```
cat = CubeSpectrumAnalysisToolbox(Cds)
```

or

```
cat = CubeSpectrumAnalysisToolbox(Cube,Wavearray)
```

```
cat = CubeSpectrumAnalysisToolbox()
```

```
cat.setCube(simpleCube)
```

or

```
cat = CubeSpectrumAnalysisToolbox()
```

```
cat.setCube(SpectralSimpleCube)
```

or

```
cat = CubeSpectrumAnalysisToolbox()
```

```
cat.setCube(Cube,VaweArray)
```

or

```
cat = CubeSpectrumAnalysisToolbox()
```

```
cat.setCube(SimpleCube)
```

to create a simplecube and launch the CubeAnalysistoolbox do this:

```

from herschel.ia.dataset import *
from herschel.ia.numeric import *
from herschel.ia.dataset.image import *
from herschel.ia.dataset.image.wcs import *
import herschel.ia.gui.plot.PlotXY
from herschel.share.unit.Length import *
Waves =Double1d.range(900)/900.*4.E-6
print Waves
a= Double1d(900)
b= RandomUniform()
cube = Double3d(900,5,6)
for raw in range(5):
for column in range(6):
shift1=int( 100.*b.calc(1.))
for wave in range(900):
if raw==2:
cube[wave,raw,column]=3 + EXP((-1.)*(wave - (350.+shift1))*(wave - (350.+shift1))/
((50.*50.))*100.+b.calc(1.)
elif raw==3:
cube[wave,raw,column]=3 + EXP((-1.)*(wave - (350.+shift1))*(wave - (350.+shift1))/
((50.*50.))*100.+b.calc(1.)
else :
cube[wave,raw,column]=0
myWcs = Wcs()
simplecube2 = SimpleCube()
simplecube2.setCube(cube)
myWcs.setNAxis(3)
myWcs.setCrval1(0.0)
myWcs.setCrval2(0.0)
myWcs.setCrpix1(0)
myWcs.setCrpix2(0)
myWcs.setCdelt1(0.1)
myWcs.setCdelt2(0.1)
myWcs.setCtype1("RA---TAN")
myWcs.setCtype2("DEC--TAN")

```



```

myWcs.setCtype3("Wavelength")
myWcs.setCunit1("[4.84813681109536E-6 rad]")
myWcs.setCunit2("[4.84813681109536E-6 rad]")
myWcs.setCunit3("[MICROMETERS]")
myWcs.setEpoch(2000)
myWcs.setImageIndex(Waves,MICROMETERS)
simplecube2.setWcs(myWcs)

import herschel.ia.gui.cube.CubeSpectrumAnalysisToolbox

# initialization of the CubeSpectrumAnalysisToolbox

cat = CubeSpectrumAnalysisToolbox()

cat.setCube(simplecube2)

```

Example

Example 1: A basic example on how to open a toolbox for spectrum analysis on an

```

SimpleCube Cds or a couple of arrays (Double3d cube,Double1d wavelength)
cat = CubeSpectrumAnalysisToolbox(Double3d)<br>
or<br>
cat = CubeSpectrumAnalysisToolbox(SimpleCube)<br>
or<br>
cat = CubeSpectrumAnalysisToolbox(Cube, Wavearray)<br>
cat = CubeSpectrumAnalysisToolbox()<br>
cat.setCube(Cds)<br>
or <br>
cat = CubeSpectrumAnalysisToolbox() <br>
cat.setCube(Cube, VaweArray)<br>
or <br>
<br>
cat = CubeSpectrumAnalysisToolbox() <br>
cat.setCube(SimpleCube)<br>
to create a simplecube and launch the CubeAnalysistoolbox do this:
from herschel.ia.dataset import *
from herschel.ia.numeric import *
from herschel.ia.dataset.image import *
from herschel.ia.dataset.image.wcs import *
import herschel.ia.gui.plot.PlotXY
from herschel.share.unit.Length import *
Waves =Double1d.range(900)/900.*4.E-6
print Waves
a= Double1d(900)
b= RandomUniform()
cube = Double3d(900,5,6)
for raw in range(5):
for column in range(6):
shift1=int( 100.*b.calc(1.))
for wave in range(900):
if raw==2:
cube[wave,raw,column]=3 + EXP((-1.)*(wave --(350.+shift1))*(wave -- (350.
+shift1))/((50.*50.))*100.+b.calc(1.))
elif raw==3:
cube[wave,raw,column]=3 + EXP((-1.)*(wave -- (350.+shift1))*(wave -- (350.
+shift1))/((50.*50.))*100.+b.calc(1.))
else -:
cube[wave,raw,column]=0
myWcs = Wcs()
simplecube2 = SimpleCube()
simplecube2.setCube(cube)
myWcs.setNAxis(3)
myWcs.setCrval1(0.0)

```

Example 1: A basic example on how to open a toolbox for spectrum analysis on an

```

myWcs.setCrval2(0.0)
myWcs.setCrpix1(0)
myWcs.setCrpix2(0)
myWcs.setCdelt1(0.1)
myWcs.setCdelt2(0.1)
myWcs.setType1("RA--TAN")
myWcs.setType2("DEC--TAN")
myWcs.setType3("Wavelength")
myWcs.setCunit1("[4.84813681109536E-6 rad]")
myWcs.setCunit2("[4.84813681109536E-6 rad]")
myWcs.setCunit3("[MICROMETERS]")
myWcs.setEpoch(2000)
myWcs.setImageIndex(Waves, MICROMETERS)
simplecube2.setWcs(myWcs)
import herschel.ia.gui.cube.CubeSpectrumAnalysisToolbox
# initialization of the CubeSpectrumAnalysisToolbox
cat = CubeSpectrumAnalysisToolbox()
cat.setCube(simplecube2)

```

API Summary

Constructors	
CubeSpectrumAnalysisToolbox()	The standard constructor for CubeSpectrumAnalysisToolbox. This
CubeSpectrumAnalysisToolbox(Array3dData array3d)	Constructor for CubeSpectrumAnalysisToolbox. This constructor shows an
CubeSpectrumAnalysisToolbox(Array3dData array3d, Double1d wave1d)	Constructor for CubeSpectrumAnalysisToolbox. This constructor shows an
CubeSpectrumAnalysisToolbox(Array3dData array3d, Double1d wave1d, String specDim, String specUnit)	Constructor for CubeSpectrumAnalysisToolbox. This constructor shows an
CubeSpectrumAnalysisToolbox(type simpleCube)	Constructor for CubeSpectrumAnalysisToolbox. This constructor shows a
Methods	
SaveCurrentImage()	Makes a menubar (Image, Help) for the Jpanel version this CubeSpectrumAnalysisToolbox.
setCube(Array3dData array3d)	Sets the image you wish to analyse with this CubeSpectrumAnalysisToolbox,
setCube(type simplecube)	Sets the Cube you wish to analyse with this CubeSpectrumAnalysisToolbox,
setCube(SpectralSimpleCube spectralsimplecube)	Sets the Cube you wish to analyse with this CubeSpectrumAnalysisToolbox,
setCube(Array3dData array3d, Double1d wavearray)	Sets the image you wish to analyse with this CubeSpectrumAnalysisToolbox,
addCube(Array3dData array3d)	Adds the given image (Array3dData) to the images, analysed with
exit()	Exits this CubeSpectrumAnalysisToolbox (closes (the window of) this
closeActiveTab()	

Methods	
	Closes the active tab, if there is one.
closeAllTabs()	Closes all tabs, if there are any.
setEnabled(boolean enabled)	Disables/enables all tabs on the JTabbedPane and the menus for
boolean isEnabled()	Returns whether the JTabbedPane and menus for closing tabs and
Double[] getWaves()	Method that returns the Float[] array of the wavelength of this CubeSpectrumAnalysisToolbox.
boolean isInHipe()	Method that returns the way the CubeSpectrumAnalysisToolbox was opened.
SimpleCube getSimpleCube()	Method that returns the Float[] array of the wavelength of this CubeSpectrumAnalysisToolbox.
SpectralSimpleCube getSpectralSimpleCube()	Method that returns the Float[] array of the wavelength of this CubeSpectrumAnalysisToolbox.
String getSpecDim()	Method that return The string indicating the physical dimension of the spectrum
String getSpecUnit()	Method that return The string indicating the unit of the spectrum
JFrame getFrame()	Method that returns the JFrame of this CubeSpectrumAnalysisToolbox.
JPanel getPanel()	Method that returns the JPanel of this CubeSpectrumAnalysisToolbox.
JTabbedPane getTabbedPane()	Method that returns the JTabbedPane of this CubeSpectrumAnalysisToolbox.
SimpleImage getSimpleImage()	Returns the displayed image (SimpleImage).
PlotXY getRtSpectrum()	Returns the displayed image (ImageDataset).
Display getDisplay()	Returns the Display of the Cube you are analysing with
int getSelectedLayer()	Returns the index of the shown layer, or the index of the selected
Double getMaxValue()	Returns the maximal value of the "image" of the cube
Double getMinValue()	Returns the maximal value of the "image" of the cube
int getSelectedTab()	Returns the index of the tab, selected for the shown layer.

Methods	
JTabbedPane getTabOnPane()	Returns the tab (JTabbedPane) on the JTabbedPane, that is
VelocityPosMapPlotting getVelocityPosMap()	Returns the VelocityPosMapPlotting object
SpectrumPlotting getSpecPlot()	Returns the SpectrumPlotting to acces to all the values of
SpectrumAvgPlotting getAvgSpecPlot()	Returns the SpectrumAvgPlotting jframe to acces to all the values of
ChannelMapPlotting getChannelMapPlot()	Returns the ChannelMapPlotting jframe to acces to all the values of
IntegratedMapDisplay getIntegratedMapPlot()	Returns the IntegratedMapDisplay jframe to acces to all the values of
IntensityMapGui getLineIntensityMap()	Returns the IntegratedMapDisplay jframe to acces to all the values of
SimpleCube getExtractedCube()	Returns the range extraction gui jframe to acces to all the values of
CubeCompareDisplay getCubeCompare()	Returns cube comparison GUI jframe to acces to all the values of
Spectrum1d getSinglePixelSpectrum()	Returns the single pixel spectrum as Spectrum1d
Spectrum1d getAvgspectrum()	that returns the Averaged spectrum from the region spectrum extarction task
SpectralSimpleCube getRangeExtractedCube()	returns the sub-range cube from the range extraction range feature
SimpleImage getVelocityAxisImage()	returns the velocity position map for the "map mode" of the velocityposition map feature
SimpleCube getVelocityMapCube()	returns the velocity position map for the "map mode" of the velocityposition map feature
ArrayList getIntegratedMapImages()	Returns the integrated images as a list of Simpleimages
setWaves (Array3dData cube3d)	Method that initialize the Double1d array of the wavelength of
setWaves (Double1d waves1d)	Method that initialize the Double1d array of the wavelength of this CubeSpectrumAnalysis-Toolbox.
String getCubeName()	returns the name of the cube
setCubeName (String name)	set the name of the cube
boolean isInImage (MouseEvent e)	Checks whether the given MouseEvent has occured within the image
boolean isInImage (double pixX, double pixY)	

Methods
Checks whether the point with given PixelCoordinates (pixX, pixY)
boolean hasWCS()
Returns true if SkyCoordinates are available for the image,
showlineIntensityMap()
Method which initialise and display the line intensity map GUI.
showSpectrum()
click on a spaxel in the image to stop the real time spectrum display
showSpectrumAvg()
Drawn a region on the image, depending of the option selected
showVelocityPosMap()
Draws a straight line on the image, starting at the point in the
showChannelMap()
Draws a straight line on the image, starting at the point in the
showIntegratedMap()
Draws a straight line on the image, starting at the point in the
guiRangeExtract()
Draws a straight line on the image, starting at the point in the
ArrayList getAllFigures()
Returns all ImageFigures used for this CubeSpectrumAnalysisToolbox.
addFigures(ArrayList figures)
Adds the ImageFigures belonging to the selected tab up front to
removeLastFigure(type figures The ImageFigures to add to the list of ImageFigures)
Adds the ImageFigures belonging to the selected tab up front to
updateImage()
Updates the image, when another tab is made active.
removeAllAnnotations()
Makes all annotations invisible.
createSpaceOnTabbedPane()
Creates a free space at the end of the Arraylist of ArrayLists of
createSpaceInTab()
Creates a free space at index 0 in the ArrayList of CanvasFigures,
RangeExtractionGui getRangeextraction()

API details

Constructors

CubeSpectrumAnalysisToolbox()
The standard constructor for CubeSpectrumAnalysisToolbox. This constructor shows an empty CubeSpectrumAnalysisToolbox window in 2 parts, with only a title, buttons for maximizing, minimizing and closing the window, a menubar (File, Image, Help) and a colorbar and a statusbar at the bottom.

CubeSpectrumAnalysisToolbox(Array3dData array3d)
<p>Constructor for CubeSpectrumAnalysisToolbox. This constructor shows an empty CubeSpectrumAnalysisToolbox window in 2 parts, with a title, buttons for maximising, minimising and closing the window, and a menubar (file, Spectrum, Help). Also the image you wish to analyse, is displayed.</p> <p>Argument</p> <p>Array3dData array3d [INPUT, MANDATORY]</p>
CubeSpectrumAnalysisToolbox(Array3dData array3d, Double1d wave1d)
<p>Constructor for CubeSpectrumAnalysisToolbox. This constructor shows an empty CubeSpectrumAnalysisToolbox window in 2 parts, with a title, buttons for maximizing, minimizing and closing the window, and a menubar (file, Spectrum, Help). Also the image you wish to analyse, is displayed. THE DEFAULT UNIT is MICROMETER</p> <p>Arguments</p> <p>Array3dData array3d [INPUT, MANDATORY]</p> <p>Double1d wave1d [INPUT, MANDATORY]</p>
CubeSpectrumAnalysisToolbox(Array3dData array3d, Double1d wave1d, String specDim, String specUnit)
<p>Constructor for CubeSpectrumAnalysisToolbox. This constructor shows an empty CubeSpectrumAnalysisToolbox window in 2 parts, with a title, buttons for maximising, minimising and closing the window, and a menubar (file, Spectrum, Help). Also the image you wish to analyse, is displayed.</p> <p>Arguments</p> <p>Array3dData array3d [INPUT, MANDATORY]</p> <p>Double1d wave1d [INPUT, MANDATORY]</p> <p>String specDim [INPUT, MANDATORY]</p> <p>String specUnit [INPUT, MANDATORY]</p>
CubeSpectrumAnalysisToolbox(type simpleCube)
<p>Constructor for CubeSpectrumAnalysisToolbox. This constructor shows a CubeSpectrumAnalysisToolbox window in 2 parts, with a title, buttons for maximising, minimising and closing the window, and a menubar (file, Spectrum, Help). Also the "cube" of the Simple-cube you wish to analyse, is displayed.</p> <p>Argument</p> <p>type simpleCube [INPUT, MANDATORY, default=no default value]</p> <p>The SimpleCube containing all the information of an observation you wish to analyse and display with this CubeSpectrumAnalysisToolbox.</p>

Methods

SaveCurrentImage()
Makes a menubar (Image, Help) for the Jpanel version this CubeSpectrumAnalysisToolbox.
setCube(Array3dData array3d)
Sets the image you wish to analyse with this CubeSpectrumAnalysisToolbox,

setCube(Array3dData array3d)
to the given ImageDataset.
Argument
Array3dData array3d [INPUT, MANDATORY]

setCube(type simplecube)
Sets the Cube you wish to analyse with this CubeSpectrumAnalysisToolbox,
to the given SimpleCube.
Argument
type simplecube [INPUT, MANDATORY, default=no default value]
The SimpleCube you wish to analyse.

setCube(SpectralSimpleCube spectralsimplecube)
Sets the Cube you wish to analyse with this CubeSpectrumAnalysisToolbox,
to the given spectralSimpleCube.
Argument
SpectralSimpleCube spectralsimplecube [INPUT, MANDATORY]

setCube(Array3dData array3d, Double1d wavearray)
Sets the image you wish to analyse with this CubeSpectrumAnalysisToolbox,
to the given ImageDataset.
Arguments
Array3dData array3d [INPUT, MANDATORY]
Double1d wavearray [INPUT, MANDATORY]

addCube(Array3dData array3d)
Adds the given image (Array3dData) to the images, analysed with
this CubeSpectrumAnalysisToolbox.
Argument
Array3dData array3d [INPUT, MANDATORY]

exit()
Exits this CubeSpectrumAnalysisToolbox (closes (the window of) this
CubeSpectrumAnalysisToolbox).

closeActiveTab()
Closes the active tab, if there is one.

closeAllTabs()
Closes all tabs, if there are any.

setEnabled(boolean enabled)

Disables/enables all tabs on the JTabbedPane and the menus for closing tabs and performing image analysis, of this CubeSpectrumAnalysisToolbox.

Argument

boolean **enabled** [INPUT, MANDATORY]

boolean isEnabled()

Returns whether the JTabbedPane and menus for closing tabs and performing image analysis, of this CubeSpectrumAnalysisToolbox are enabled/disabled.

Return

boolean

Returns true if the JTabbedPane and menus for closing tabs and performing image analysis, of this CubeSpectrumAnalysisToolbox are enabled; false otherwise.

DoubleId getWaves()

Method that returns the Float1d array of the wavelength of this CubeSpectrumAnalysisToolbox.

Return

[DoubleId](#)

The Float1d array of the wavelength of this CubeSpectrumAnalysisToolbox.

boolean isInHiPe()

Method that returns the way the CubeSpectrumAnalysisToolbox was opened.

true if it's open with the right menu in the variable list of hiPe false when opened with the classical command line call

Return

boolean

the way the CubeSpectrumAnalysisToolbox was opened. true if it's open with the right menu in the variable list of hiPe false when opened with the classical command line call

SimpleCube getSimpleCube()

Method that returns the Float1d array of the wavelength of this CubeSpectrumAnalysisToolbox.

Return

[SimpleCube](#)

The Float1d array of the wavelength of this CubeSpectrumAnalysisToolbox.

SpectralSimpleCube getSpectralSimpleCube()

Method that returns the Float1d array of the wavelength of this CubeSpectrumAnalysisToolbox.

Return

SpectralSimpleCube

The Float1d array of the wavelength of this CubeSpectrumAnalysisToolbox.

String getSpecDim()
Method that return The string indicating the physical dimension of the spectrum
Return
String
The string indicating the physical dimension of the spectrum
String getSpecUnit()
Method that return The string indicating the unit of the spectrum
Return
String
The string indicating the unit of the spectrum
JFrame getFrame()
Method that returns the JFrame of this CubeSpectrumAnalysisToolbox.
Return
JFrame
The JFrame of this CubeSpectrumAnalysisToolbox.
JPanel getPanel()
Method that returns the JPanel of this CubeSpectrumAnalysisToolbox.
Return
JPanel
The JPanel of this CubeSpectrumAnalysisToolbox.
JTabbedPane getTabbedPane()
Method that returns the JTabbedPane of this CubeSpectrumAnalysisToolbox.
Return
JTabbedPane
The JTabbedPane of this CubeSpectrumAnalysisToolbox.
SimpleImage getSimpleImage()
Returns the displayed image (SimpleImage).
Return
SimpleImage
Returns the displayed image (SimpleImage).
PlotXY getRtSpectrum()
Returns the displayed image (ImageDataset).
Return
PlotXY

PlotXY `getRtSpectrum()`

Returns the displayed image (ImageDataset).

Display `getDisplay()`

Returns the Display of the Cube you are analysing with this CubeSpectrumAnalysisToolbox.

Return

[Display](#)

Returns the display of the Cube you are analysing with this CubeSpectrumAnalysisToolbox.

int `getSelectedLayer()`

Returns the index of the shown layer, or the index of the selected tab on the JTabbedPane.

Return

int

Returns the index of the shown layer, or the index of the selected tab on the JTabbedPane.

Double `getMaxValue()`

Returns the maximal value of the "image" of the cube

Return

[Double](#)

Returns the maximal value of the "image" of the cube

Double `getMinValue()`

Returns the maximal value of the "image" of the cube

Return

[Double](#)

Returns the maximal value of the "image" of the cube

int `getSelectedTab()`

Returns the index of the tab, selected for the shown layer.

Return

int

The index of the tab, selected for the shown layer.

JTabbedPane `getTabOnPane()`

Returns the tab (JTabbedPane) on the JTabbedPane, that is selected.

Return

JTabbedPane <code>getTabOnPane()</code>
JTabbedPane Returns the tab (JTabbedPane) on the JTabbedPane, that is selected.
VelocityPosMapPlotting <code>getVelocityPosMap()</code>
Returns the VelocityPosMapPlotting object Return VelocityPosMapPlotting Returns the VelocityPosMapPlotting object
SpectrumPlotting <code>getSpecPlot()</code>
Returns the SpectrumPlotting to acces to all the values of single pixel spectrum extraction. Return SpectrumPlotting Returns the SpectrumPlotting to acces to all the values of single pixel spectrum extraction.
SpectrumAvgPlotting <code>getAvgSpecPlot()</code>
Returns the SpectrumAvgPlotting jframe to acces to all the values of single pixel spectrum extraction. Return SpectrumAvgPlotting Returns the SpectrumAvgPlotting to acces to all the values of single pixel spectrum extraction.
ChannelMapPlotting <code>getChannelMapPlot()</code>
Returns the ChannelMapPlotting jframe to acces to all the values of channel map extraction Return ChannelMapPlotting Returns the ChannelMapPlotting jframe to acces to all the values of channel map extraction
IntegratedMapDisplay <code>getIntegratedMapPlot()</code>
Returns the IntegratedMapDisplay jframe to acces to all the values of channel map extraction Return IntegratedMapDisplay Returns the IntegratedMapDisplay jframe to acces to all the values of channel map extraction

IntensityMapGui <code>getLineIntensityMap()</code>
Returns the IntegratedMapDisplay JFrame to access to all the values of channel map extraction
Return
IntensityMapGui
Returns the IntegratedMapDisplay JFrame to access to all the values of channel map extraction
SimpleCube <code>getExtractedCube()</code>
Returns the range extraction GUI JFrame to access to all the values of channel map extraction
Return
SimpleCube
Returns the range extraction GUI JFrame to access to all the values of channel map extraction
CubeCompareDisplay <code>getCubeCompare()</code>
Returns cube comparison GUI JFrame to access to all the values of the feature
Return
CubeCompareDisplay
Returns cube comparison GUI JFrame to access to all the values of the feature
Spectrum1d <code>getSinglePixelSpectrum()</code>
Returns the single pixel spectrum as Spectrum1d
Return
Spectrum1d
Returns the single pixel spectrum as Spectrum1D
Spectrum1d <code>getAvgspectrum()</code>
that returns the Averaged spectrum from the region spectrum extraction task
Return
Spectrum1d
that returns the Averaged spectrum from the region spectrum extraction task
SpectralSimpleCube <code>getRangeExtractedCube()</code>
returns the sub-range cube from the range extraction range feature
Return
SpectralSimpleCube
returns the sub-range cube from the range extraction range feature

SimpleImage getVelocityAxisImage()
returns the velocity position map for the "map mode" of the velocityposition map feature
Return
SimpleImage
returns the velocity position map for the "map mode" of the velocityposition map feature
SimpleCube getVelocityMapCube()
returns the velocity position map for the "map mode" of the velocityposition map feature
Return
SimpleCube
returns the velocity position map for the "map mode" of the velocityposition map feature
ArrayList getIntegratedMapImages()
Returns the integrated images as a list of Simpleimages
Return
ArrayList
Returns the integrated images as a list of Simpleimages
setWaves(Array3dData cube3d)
Method that initialize the Double1d array of the wavelength of this CubeSpectrumAnalysisToolbox.
Argument
Array3dData cube3d [INPUT, MANDATORY]
setWaves(Double1d waves1d)
Method that initialize the Double1d array of the wavelength of this CubeSpectrumAnalysisToolbox.
Argument
Double1d waves1d [INPUT, MANDATORY]
String getCubeName()
returns the name of the cube
Return
String
returns the name of the cube
setCubeName(String name)
set the name of the cube
Argument
String name [INPUT, MANDATORY]
boolean isInImage(MouseEvent e)
Checks whether the given MouseEvent has occurred within the image

boolean isInImage(MouseEvent e)

we are analysing with this CubeSpectrumAnalysisToolbox.

Argument

[MouseEvent](#) **e** [INPUT, MANDATORY]

Return

boolean

Returns true if the given MouseEvent has occurred within the image that is being analysed with this CubeSpectrumAnalysisToolbox; false otherwise.

boolean isInImage(double pixX, double pixY)

Checks whether the point with given PixelCoordinates (pixX, pixY)

lies in the image we are analysing with this CubeSpectrumAnalysisToolbox.

Arguments

double **pixX** [INPUT, MANDATORY]

double **pixY** [INPUT, MANDATORY]

Return

boolean

Returns true if the point with given PixelCoordinates (pixX, pixY) lies in the image we are analysing with this CubeSpectrumAnalysisToolbox; false otherwise.

boolean hasWCS()

Returns true if SkyCoordinates are available for the image,

analysed with this CubeSpectrumAnalysisToolbox; false otherwise.

Return

boolean

Returns true if SkyCoordinates are available for the image, analysed with this CubeSpectrum-AnalysisToolbox; false otherwise.

showlineIntensityMap()

Method which initialise and display the line intensity map GUI.

showSpectrum()

click on a spaxel in the image to stop the real time spectrum display

and keep the last one, when you are out of the image the spectrum is the last one which was read

showSpectrumAvg()

Drawn a region on the image, depending of the option selected

on the right side of the window and of the point clicked in the image with the mouse. and plots the averaged spectrum along that "cylinder". When you click or move outside of the image, no shape is defined.

showVelocityPosMap()

Draws a straight line on the image, starting at the point in the
 you clicked on with the mouse the 1st time, and ending at the current mouse position (in the image), or at the 2nd point (in the image) you clicked on, and plots the intensity along that straight line. When you click or move outside of the image, no straight line is drawn and no plot is shown.

showChannelMap()

Draws a straight line on the image, starting at the point in the
 you clicked on with the mouse the 1st time, and ending at the current mouse position (in the image), or at the 2nd point (in the image) you clicked on, and plots the intensity along that straight line. When you click or move outside of the image, no straight line is drawn and no plot is shown.

showIntegratedMap()

Draws a straight line on the image, starting at the point in the
 you clicked on with the mouse the 1st time, and ending at the current mouse position (in the image), or at the 2nd point (in the image) you clicked on, and plots the intensity along that straight line. When you click or move outside of the image, no straight line is drawn and no plot is shown.

guiRangeExtract()

Draws a straight line on the image, starting at the point in the
 you clicked on with the mouse the 1st time, and ending at the current mouse position (in the image), or at the 2nd point (in the image) you clicked on, and plots the intensity along that straight line. When you click or move outside of the image, no straight line is drawn and no plot is shown.

[ArrayList](#) getAllFigures()

Returns all ImageFigures used for this CubeSpectrumAnalysisToolbox.

Return

[ArrayList](#)

Returns all ImageFigures used for this CubeSpectrumAnalysisToolbox.

addFigures([ArrayList](#) figures)

Adds the ImageFigures belonging to the selected tab up front to
 the list of ImageFigures, used for this CubeSpectrumAnalysisToolbox.

Argument

[ArrayList](#) figures [INPUT, MANDATORY]

removeLastFigure(type figures The ImageFigures to add to the list of ImageFigures)

Adds the ImageFigures belonging to the selected tab up front to
 the list of ImageFigures, used for this CubeSpectrumAnalysisToolbox.

Argument

type figures The ImageFigures to add to the list of ImageFigures
 [INPUT, MANDATORY, default=no default value]

<code>removeLastFigure(type figures The ImageFigures to add to the list of ImageFigures)</code>

used for this CubeSpectrumAnalysisToolbox.
--

<code>updateImage()</code>

Updates the image, when another tab is made active.

<code>removeAllAnnotations()</code>

Makes all annotations invisible.

<code>createSpaceOnTabbedPane()</code>
--

Creates a free space at the end of the ArrayList of ArrayLists of ArrayLists of ImageFigures, that belong to the image analysis on the shown layer.

<code>createSpaceInTab()</code>

Creates a free space at index 0 in the ArrayList of CanvasFigures, belonging to a new image analysis on the shown layer.
--

<code>RangeExtractionGui getRangeextraction()</code>
--

Return


RangeExtractionGui

the RangeExtractionGui

See also

- [Display, PlotXY](#)

2.90. CubicSplineInterpolator

Full Name:	herschel.ia.numeric.toolbox.interp.CubicSplineInterpolator
Alias:	CubicSplineInterpolator
Type:	Java Class - 
Import:	from herschel.ia.numeric.toolbox.interp import CubicSplineInterpolator

Description

Given a set of knots (x/y data), this function computes values at arbitrary positions

by use of a cubic spline. It assumes that the second derivatives are zero at the endpoints (i.e. a natural spline). This can only be applied to numeric arrays of rank 1."

Example

Example 1: Create and apply a CubicSplineInterpolator on a Double1d

```
# create some knots
x=Double1d.range(10) # [0.0,1.0,2.0,3.0,4.0,5.0,6.0,7.0,8.0,9.0]
f=CubicSplineInterpolator(x,SQUARE(x))
u=Float1d([1,1.1,2,2.6,3,3.9])
print f(u) # [1.0,1.1970944,4.0,6.7656145,9.0,15.209593]
```

API Summary

Jython Syntax

```
<f>=CubicSplineInterpolator(<x>,<y>[,allowExtrapolation])
```

Properties

[Double1d x](#) [INPUT, MANDATORY, default=no default value]

[Double1d y](#) [INPUT, NOT MANDATORY, default=false]

[boolean allowExtrapolation](#) [INPUT, NOT MANDATORY, default=false]

API details

Properties

[Double1d x](#) [INPUT, MANDATORY, default=no default value]

The knots are Double1d

[Double1d y](#) [INPUT, NOT_MANDATORY, default=false]

The knots are Double1d

[boolean allowExtrapolation](#) [INPUT, NOT_MANDATORY, default=false]


Extrapolation is not allowed by default. Boolean.TRUE allows extrapolation.

See also

- [LinearInterpolator](#)

- [NearestNeighborInterpolator](#)

2.91. CubicSplinesModel

Full Name:	herschel.ia.numeric.toolbox.fit.CubicSplinesModel
Type:	Java Class - 
Import:	from herschel.ia.numeric.toolbox.fit import CubicSplinesModel

Description

General cubic splines model with arbitrary knot settings.

It is a linear model.

A number (<< datapoints) of knots are layed out on the x-axis at arbitrary position, generally more knots where the curvature is higher. The knots need to be monotonously increasing in x. Through these knots a cubic splines function is obtained which best fits the datapoints. One needs at least 2 knots, one smaller and one larger than all x-values in the dataset.

If the end knots are put in between the x-values in the dataset, a kind of extrapolating spline is obtained. It still works more or less.

This model is NOT for cubic spline interpolation.


For interpolation see: `herschel.ia.numeric.toolbox.interp.CubicSplineInterpolator`

A more complete worked out [example](#) is found in the HCSS-DRM (developers reference manual).

Example

Example 1: to use CubicSplinesModel
<pre>npt = 160 x = DoubleItd.range(npt -) knots = DoubleItd.range(17 -) * 10 # make knots from 0 to 160 csm = CubicSplinesModel(knots -) print csm.getNumberOfParameters() # 19 # ... fitter etc. see Fitter</pre>

2.92. cutLevels

Full Name:	herschel.ia.toolbox.image.CutLevelsTask
Alias:	cutLevels
Type:	Java Task - 
Import:	from herschel.ia.toolbox.image import CutLevelsTask
Category:	task/image

Description

A Task to calculate the cut levels of an image.

CutLevelsTask is a task which returns the cut levels of an Image. The cut levels can be calculated using 2 methods :

- PERCENT : A certain percentage of the pixels is calculated.
- MEDIAN_FILTER : A window of 7 pixels is taken. From this window, the median is taken. This median is then compared with the minimum and maximum value of the image. 21 pixels are skipped in the column dimension and 3 rows are skipped.

Examples

Example 1: Calculate the cut levels where 95% of the values are included.

```
cutLevelsTask(image = im, method=CutLevels.PERCENT, percent = 95.0)
```

Example 2: Calculate the cut levels using the median filter

```
cutLevelsTask(image = im, method=CutLevels.MEDIAN_FILTER)
```

API Summary

Jython Syntax

```
cutLevels(image, CutLevels.PERCENT, 95.0)
```

Properties

[Image image](#) [INPUT, MANDATORY, default=No default value]

[int method](#) [INPUT, MANDATORY, default=CutLevels.MEDIAN_FILTER]

[double percent](#) [INPUT, OPTIONAL, default=99.5]

[double\[\] cutLevels](#) [OUTPUT, MANDATORY, default=No default value]

API details

Properties

Image image [INPUT, MANDATORY, default=No default value]

The image to use for calculating the cut levels.

int method [INPUT, MANDATORY, default=CutLevels.MEDIAN_FILTER]

The method to use for calculating the cut levels (CutLevels.PERCENT or CutLevels.MEDIAN_FILTER).

double percent [INPUT, OPTIONAL, default=99.5]

The percentage of pixels to use in the calculation of the cut levels.

double[] cutLevels [OUTPUT, MANDATORY, default=No default value]

The minimum and maximum cut level.

2.93. CutLevelsTaskSignatureComponent

Full Name:	herschel.ia.toolbox.image.gui.CutLevelsTaskSignatureComponent
Type:	Java Class - 
Import:	from herschel.ia.toolbox.image.gui import CutLevelsTaskSignatureComponent

Description

A task dialog for the CutLevelsTask.

A task dialog to serve as GUI for the SourceFittingTask.

API Summary

Constructor
CutLevelsTaskSignatureComponent() The constructor of a new CutLevelsTaskSignatureComponent.
Methods
check() Validates the current modifications.
clear() Clears the current modifications.
Map getParameters() Returns a map of parameters and modifiers.
setSignature(SignatureApi sig) Setting the signature.
setVariableSelection(VariableSelection selection) Setting the variable selection.
JComponent getComponent() Returns the component.

API details

Constructor


<code>CutLevelsTaskSignatureComponent()</code>
The constructor of a new CutLevelsTaskSignatureComponent.
The constructor of a new CutLevelsTaskSignatureComponent.

Methods

<code>check()</code>
Validates the current modifications.
Validates the current modifications for this CutLevelsTaskSignatureComponent.

clear()
Clears the current modifications. Clears the current modifications for this CutLevelTaskSignatureComponent.
Map getParameters()
Returns a map of parameters and modifiers. Returns a map of parameters and modifiers storing the value selected by the user for this CutLevelTaskSignatureComponent. Return Map Returns a map of parameters and modifiers storing the value selected by the user for this CutLevelsTaskSignatureComponent.
setSignature(SignatureApi sig)
Setting the signature. Sets the given signature as the signature for this CutLevelsTaskSignatureComponent. Argument SignatureApi sig [INPUT, MANDATORY]
setVariableSelection(VariableSelection selection)
Setting the variable selection. Sets the given selection as the variable selection for this CutLevelsTaskSignatureComponent. Argument VariableSelection selection [INPUT, MANDATORY]
JComponent getComponent()
Returns the component. Returns the component for this CutLevelsTaskSignatureComponent. Return JComponent Returns the component for this CutLevelsTaskSignatureComponent.

2.94. DataFlow

Full Name:	herschel.ia.dataflow.DataFlow
Type:	Java Class - 
Import:	from herschel.ia.dataflow import DataFlow

Description

Groups several processes as one big process.

It is a process that may manage processes. Processes are created within the dataflow, and may be connected among them. DataFlow allows created processes outside to be added to it. A developer may inherit from this class in order to create a specific dataflow.

Example

Example 1: how to use a dataflow with given processes.

```
<pre>
from herschel.ia.dataflow import *
#creating dataflow and processes.
df = DataFlow("MyDataFlow")
df.createProcess("generator", -"myprocesses.ProcessGenerator")
df.createProcess("fft", -"myprocesses.ProcessFFT")
# adding a created process
p = myprocess.ProcessFFT("viewer")
df.addProcess(p)
#connecting processes
df.connect("generator.output", -"fft.input")
df.connect("fft.output", -"viewer.input")
# putting the viewer in a JFrame.
from javax.swing import *
frame = JFrame("MyFrame")
viewer = df.getProcess("viewer")
frame.getContentPane().add(viewer.getJComponent())
frame.setSize(600,600)
frame.setVisible(1)
# starting the dataflow.
df.start()
</pre>
```

Limitations

no limitation


See also

- [reference](#)

History

- 26-5-2005 jcg: change javadoc format for help support.

2.95. DataFlowManager

Full Name:	herschel.ia.dataflow.DataFlowManager
Type:	Java Class - 
Import:	from herschel.ia.dataflow import DataFlowManager

Description

Shows dataflows.

Allows to the user to see the dataflow in a graphical environment.

Example

Example 1: how to show a dataflow

```
<pre>
from herschel.ia.dataflow import *
df = DataFlow("mydf")
df.createProcess(...)
df.createProcess(...)
df.createProcess(...)
DataFlowManager(df)
</pre>
```

Limitations

no limitation


See also

- [reference](#)

History

- 26-5-2005 jcg: change javadoc format for help support.

2.96. DbFactory

Full Name:	herschel.ia.pal.pool.db.DbFactory
Type:	Java Class - 
Import:	from herschel.ia.pal.pool.db import DbFactory
Category:	PAL

Description

An implementation of a ProductPool which saves data to a versant database.

You need to define a property which will store the name of the database you wish the DbPool to use. Please check that the property of the same name points to a valid physical database.

Depending on your network performance and proximity to the database pool you may want to wrap a CachedPool around the DbPool. See the last example below.

Examples

Example 1: Create a DbPool passing it a map

```
Configuration.setProperty("dbProperty", -"&lt;dbname&gt;@&lt;hostname&gt;")
params = java.util.HashMap()<br>
params.put("id", -"dbpool")<br>
params.put("database", -"dbProperty")<br>
storage=ProductStorage();<br>
storage.register(DbFactory.createPool("db", params)
```

Example 2: Create a DbPool that points to the database property "hcss.ia.pal.pool.db.database"

```
storage=ProductStorage();<br>
storage.register(DbFactory.getStore())
```

Example 3: Create a DbPool that points to the database <dbname>@<hostname>

```
storage=ProductStorage() <br>
storage.register(DbFactory.getStore(poolName, &lt;dbname&gt;@&lt;hostname&gt;))
```

Example 4: Saving and Loading to/from a DbPool

```
ref=storage.save(myproduct) <br>
ref=storage.load(ref)
```

Example 5: Wrapping a CachedPool around a DbPool

```
storage.register(CachedPool(DbFactory.getStore()))
```

2.97. DbPool

Full Name:	herschel.ia.pal.pool.db.DbPool
Type:	Java Class - 
Import:	from herschel.ia.pal.pool.db import DbPool
Category:	PAL

Description

An implementation of a ProductPool which saves data to a versant database (to be exact, any object database that uses the HCSS ObjectStore interface).

New: The DbPool continues to support the direct creation of its pools, however it is now recommended that DbPool(s) be created using the DbFactory methods.

You need to define a property which will store the name of the database you wish the DbPool to use. By default this is "hcss.test.database". Please check that the property of the same name points to a valid physical database.

Depending on your network performance and proximity to the database pool you may want to wrap the a CachedPool around the DbPool. See the last example below.

Examples

Example 1: Create a DbPool that points to the default database property "hcss.test.database"

```
storage=ProductStorage() <br>
storage.register(DbPool.getInstance())
```

Example 2: Create a DbPool that points to the database property "my.database"

```
storage=ProductStorage() <br>
storage.register(DbPool.getInstance("my.database", poolName))
```

Example 3: Saving and Loading to/from a DbPool

```
ref=storage.save(myproduct) <br>
ref=storage.load(ref)
```

Example 4: Wrapping a CachedPool around a DbPool

```
storage.register(CachedPool(DbPool.getInstance()))
```

API Summary

Methods
DbPool getInstance(String databaseProperty, String poolName)
DbPool
DbPool getInstance(String databaseProperty)

Methods
DbPool
DbPool getInstance() Creates an instance of a DbPool connected to the logical database of default property named
DbPool createInstance(String databaseProperty) DbPool

API details

Methods

DbPool getInstance(String databaseProperty, String poolName)
DbPool
Creates an instance of a DbPool connected to a logical database
Arguments
String databaseProperty [INPUT, MANDATORY, default=no default value] The logical name of the database, for example "hcss.test.database"
String poolName [INPUT, MANDATORY, default=no default value] The actual name of the database pool
Return
DbPool
An instance of the DbPool.

DbPool getInstance(String databaseProperty)
DbPool
Creates an instance of a DbPool connected to a logical database
Argument
String databaseProperty [INPUT, MANDATORY, default=no default value] The logical name of the database, for example "hcss.test.database"
Return
DbPool
An instance of the DbPool.

DbPool getInstance()
Creates an instance of a DbPool connected to the logical database of default property named
"hcss.test.database"
Return
DbPool
An instance of the DbPool.

[DbPool](#) `createInstance(String databaseProperty)`

DbPool

Creates an instance of a DbPool connected to a logical database

Argument


[String](#) `databaseProperty` [INPUT, MANDATORY]

Return

[DbPool](#)

An instance of the DbPool.

2.98. DETERMINANT

Full Name:	herschel.ia.numeric.toolbox.matrix.MatrixDeterminant
Alias:	DETERMINANT
Type:	Java Class - 
Import:	from herschel.ia.numeric.toolbox.matrix import MatrixDeterminant

Description

Yields the determinant of a square matrix.

Example

Example 1: Apply a DETERMINANT to a Double2d matrix

```
A=Double2d([ [1,2],[3,4] -])
print DETERMINANT(A) # --2
```

API Summary

Jython Syntax

```
<y>=DETERMINANT(<x>)
```

Properties

[any square matrix x \[INPUT, MANDATORY, default=no default value\]](#)

[double y \[INPUT, NOT_MANDATORY, default=false\]](#)

Miscellaneous

Does not work for complex matrices.

API details

Properties

[any square matrix x \[INPUT, MANDATORY, default=no default value\]](#)

Any square matrix


[double y \[INPUT, NOT_MANDATORY, default=false\]](#)

Returns a double

See also

- [CubicSplineInterpolator](#)
- [LinearInterpolator](#)

2.99. DFT2dTask

Full Name:	herschel.ia.toolbox.image.DFT2dTask
Type:	Java Task - 
Import:	from herschel.ia.toolbox.image import DFT2dTask

Description

A Task for two dimensional Discrete Fourier Transforms.

A Task that calculates the 2D Discrete Fourier Transform and the power spectrum.

API Summary


Properties
Numeric2dData image [INPUT, MANDATORY, default=No default value]
Complex2d transform [OUTPUT, MANDATORY, default=No default value]
Double2d spectrum [OUTPUT, MANDATORY, default=No default value]

API details

Properties

Numeric2dData image [INPUT, MANDATORY, default=No default value]
The input image.
Complex2d transform [OUTPUT, MANDATORY, default=No default value]
The transform.
Double2d spectrum [OUTPUT, MANDATORY, default=No default value]
The spectrum.

2.100. Display

Full Name:	herschel.ia.gui.image.Display
Alias:	Display
Type:	Java Class - 
Import:	from herschel.ia.gui.image import Display
Category:	Image

Description

An Image display for DP. A class to display images.

This class can display a SimpleImage, SimpleCube or any numeric2d or numeric3d object. A status bar, color bar, a zoomed section of the image and an overview of the image are also available.

Examples

Example 1: Display a SimpleImage
<pre>d = Display(mySimpleImage)</pre>

Example 2: A basic example on how to Display a Double2d
<pre>im = Double2d(600, 400) for i in range(600): for j in range(400): im.set(i, j, i + j) d = Display(im)</pre>

API Summary

Constructors
Display() The standard constructor
Display(boolean useAsComponent, boolean imageVisible) A Display constructor
Display(boolean useAsComponent) A constructor to use Display as a component.
Display(Image imds) A constructor which displays an Image.
Display(Cube cube) A constructor which displays a Cube.
Display(Stack stack) A constructor which displays a Stack.
Display(RgbImage imds) A constructor which displays an RgbImage.
Display(Image imds, boolean imageVisible) A Display constructor
Display(Cube imds, boolean imageVisible) A Display constructor

Constructors	
Display(Stack stack, boolean imageVisible)	A Display constructor
Display(Array2dData data)	A constructor to display a Numeric2d.
Display(Array2dData data, boolean imageVisible)	A Display constructor
Display(Array3dData data)	A constructor to display a numeric3d
Display(Array3dData data, boolean imageVisible)	A display constructor
Methods	
setSize(int x, int y)	setSize
int getLayerShown()	Returns the number of the shown layer
setVisible(boolean imageVisible)	Toggles the display visible or invisible.
JPanel getComposedComponent()	Returns a panel with all components of this Display on it.
ImageColorbar getColorBarComponent()	Returns the color bar component.
DivaMainImageDisplay getComponent()	Returns the component where the image is displayed.
ImagePanner getPannerComponent()	Returns the panner component with the overview of the image.
StatusPanel getStatusPanelComponent()	Returns the component with the status panel.
setStatusPanelComponent(StatusPanel imageDisplayStatusPanel)	Set a new status panel for the display.
ImageZoom getZoomComponent()	Returns the component with the zoomed view.
setImage(Array2dData imageData)	Set a new numeric2d image.
setImage(Array3dData imageData)	Sets a new numeric3d image.
setImage(Image imds)	Sets a new Image to Display.
setImage(Cube cube)	Sets a new cube to display.
setImage(Stack stack)	Sets a new stack to display.

Methods	
setImage(RgbImage imds)	Sets a new RgbImage to Display.
addLayer(Layer layer)	Adds a new layer to the display.
addLayer(Array2dData imageData)	Adds a new layer with numeric2d data
addLayer(Array3dData imageData)	Adds a new layer with numeric3d data
addLayer(Image imds)	Adds a new layer with an Image to the display.
addLayer(RgbImage imds)	Adds a new layer with an RgbImage to the display.
addLayer(Cube cube)	Adds a new layer with a Cube to the display.
addLayer(Stack stack)	Adds a new layer with a Stack to the display.
Layer getLayer()	Returns the shown layer.
Layer getLayer(int i)	Returns the Layer object
showLayer(int i)	Shows a chosen layer.
removeLayer()	Removes the current layer.
removeLayer(int i)	Removes a layer.
updateImage(Array2dData imageData)	Updates the layer with numeric2d data.
updateImage(Image imds)	Updates the layer with an Image.
updateImage(RgbImage imds)	Updates the layer with an Image.
updateImage(Array3dData imageData)	Updates the layer with numeric3d data.
updateImage(Cube imds)	Updates the layer with a cube.
updateImage(Stack stack)	Updates the layer with a stack.
setBackground(Color bgColor)	Sets the background color.
annotationToolbox()	

Methods	
	Fires up the annotation toolbox.
CanvasFigure addAnnotation(String text, double row, double column)	Adds a text annotation.
CanvasFigure addAnnotationWorldCoordinates(String text, double ra, double decl)	Adds a text annotation
CanvasFigure addGreekAnnotation(String text, double row, double column)	Adds a Greek annotation
CanvasFigure addGreekAnnotationWorldCoordinates(String text, double ra, double decl)	Adds a Greek annotation
Font getAnnotationFont()	Returns the default font for annotations.
Font getAnnotationFont(double row, double column)	Returns the font for the specified annotation.
Font getAnnotationFontWorldCoordinates(double ra, double dec)	Returns the font for the specified annotation.
Color getAnnotationFontColor()	Returns the default color for annotations.
Color getAnnotationFontColor(double row, double column)	Returns the color of the specified annotation.
Color getAnnotationFontColorWorldCoordinates(double ra, double dec)	Returns the color of the specified annotation.
removeAnnotation(double row, double column)	Removes the specified annotation
removeAnnotationWorldCoordinates(double ra, double dec)	Remove the specified annotation.
removeAnnotation(CanvasFigure fig)	Removes the specified annotation.
removeAnnotations()	Removes all annotations.
setAnnotationFont(Font f)	Changes the font of all Annotations.
setAnnotationFont(double row, double column, Font f)	Changes the font of the specified annotations.
setAnnotationFontWorldCoordinates(double ra, double dec, Font f)	Changes the font of the specified annotations.
setAnnotationFont(int size)	Changes the font size of all annotations.
setAnnotationFont(double row, double column, int size)	

Methods
Changes the font size of the specified annotations.
<code>setAnnotationFontWorldCoordinates(double ra, double dec, int size)</code>
Changes the font size of the specified annotations.
<code>setAnnotationFontColor(Color color)</code>
Sets the default color for annotations.
<code>setAnnotationFontColor(double row, double column, Color color)</code>
Sets the font color for the specified annotation.
<code>setAnnotationFontColorWorldCoordinates(double ra, double dec, Color color)</code>
Sets the font color for the specified annotation.
<code>CanvasFigure addEllipse(double row, double column, double w, double h, float lineWidth, Color color)</code>
Adds an ellipse.
<code>addFigure(CanvasFigure fig)</code>
Adds a CanvasFigure
<code>CanvasFigure addEllipse(double row, double column, double w, double h, float lineWidth, Color color, double positionAngle)</code>
Adds an ellipse
<code>CanvasFigure addCircle(double row, double column, double radius, float lineWidth, Color color)</code>
Adds a circle
<code>CanvasFigure addLine(double row1, double column1, double row2, double column2, float lineWidth, Color color)</code>
Adds a line.
<code>CanvasFigure addPolygon(double[] coords, float lineWidth, Color color)</code>
Adds a polygon
<code>CanvasFigure addPolyline(double[] coords, float lineWidth, Color color)</code>
Adds a polyline.
<code>CanvasFigure addRectangle(double minRow, double minColumn, double w, double h, float lineWidth, Color color)</code>
Adds a rectangle.
<code>addCompass(double row, double column, double size, float lineWidth, Color color)</code>
Adds a compass with the north and east directions.
<code>addTenArcSecs(double row, double column, float lineWidth, Color color)</code>
Adds a line which is 10 arc seconds to the image.
<code>ArrayList addImageContour(ImageContour imageContour, ArrayList colors, int minLength)</code>
Adds an ImageContour.
<code>ArrayList addImageContour(ImageContour imageContour, int minLength)</code>

Methods
Adds an ImageContour.
ArrayList <code>addImageContour</code>(ImageContour imageContour, ArrayList colors)
Adds an ImageContour.
ArrayList <code>addImageContour</code>(ImageContour imageContour)
Adds an ImageContour.
ArrayList <code>addWcsImageContour</code>(ImageContour imageContour, ArrayList colors, int minLength)
Add an ImageContour.
ArrayList <code>addWcsImageContour</code>(ImageContour imageContour, int minLength)
Add an ImageContour.
ArrayList <code>addWcsImageContour</code>(ImageContour imageContour, ArrayList contourColors)
Adds an ImageContour.
ArrayList <code>addWcsImageContour</code>(ImageContour imageContour)
Adds an ImageContour.
ArrayList <code>addContourLevel</code>(ContourLevel contourLevel, Color contourColor, int minimumContourLength)
Adds a ContourLevel.
ArrayList <code>addContourLevel</code>(ContourLevel contourLevel, int minimumContourLength)
Adds a ContourLevel.
ArrayList <code>addContourLevel</code>(ContourLevel level, Color color)
Adds a ContourLevel.
ArrayList <code>addContourLevel</code>(ContourLevel level)
Adds a ContourLevel.
ArrayList <code>addContourLevel</code>(ContourLevel level, Wcs wcs, Color color, int minLength)
Adds a ContourLevel.
ArrayList <code>addContourLevel</code>(ContourLevel level, Wcs wcs, int minLength)
Adds a ContourLevel.
ArrayList <code>addContourLevel</code>(ContourLevel level, Wcs wcs, Color color)
<code>addContourLevel(ContourLevel level, Wcs wcs, Color color)</code>
ArrayList <code>addContourLevel</code>(ContourLevel level, Wcs wcs)
<code>addContourLevel(ContourLevel level, Wcs wcs, Color color)</code>
ImageFigure <code>addContour</code>(Contour contour, Color color, int minLength)
Adds a Contour.
ImageFigure <code>addContour</code>(Contour contour, Color color)
Adds a Contour

Methods	
ImageFigure addContour(Contour contour, Wcs wcs, Color color, int minLength)	Adds a Contour.
ImageFigure addContour(Contour contour, Wcs wcs, Color color)	Adds a Contour.
ArrayList addPositionList(PositionList positionList, Color color)	Overlays the given source list on this Display,
ArrayList addPositionList(PositionList positionList)	Overlays the given source list on this Display,
ArrayList addSourceFit(SourceFittingProduct source, Color color)	Visualises the given SourceFittingProduct on this Display in the given color.
ArrayList addSourceFit(SourceFittingProduct source)	Visualises the given SourceFittingProduct on this Display.
ArrayList showAxes(boolean showAxis)	Enables or disables the axes for the displayed image.
ImageAxis getLeftaxis()	Returns the left axis.
ImageAxis getRightaxis()	Returns the right axis.
ImageAxis getBottomaxis()	Returns the bottom axis.
ImageAxis getTopaxis()	Returns the top axis.
ArrayList getAxes()	Returns an array with the axes.
addAxis(String label, Position orientation)	Adds new axis.
addAxis(ImageAxis axis)	Adds new axis.
deleteAxis(ImageAxis axis)	Deletes the given axis.
setCenter(int x, int y)	setCenter
close()	Closes the display.
editColors()	Edit the colors using a popup.
editCutLevels()	Edit the cut levels using a popup.
String getColortable()	Returns the name of the color table.
double[] getCutLevels()	

Methods	
	Returns the cut levels.
Image getImage()	Returns the displayed image.
RgbImage getRgbImage()	Returns the displayed image.
double[] getPixelCoordinates(double c1, double c2)	Returns the image coordinates
Number getIntensity(int row, int column)	Returns the intensity of the pixel
Number getIntensityFromWorldCoordinates(double c1, double c2)	Returns the intensity of the pixel.
Unit getUnit()	Returns the unit.
float getZoomFactor()	Returns the used zoom factor.
printDialog()	Opens a printer dialog.
getPrintControl()	Returns the printControl.
createPrintJob()	Creates a print job.
PrinterJob getPrintJob()	Returns the printer job.
setPrintJob(PrinterJob job)	Sets a printer job.
print(String path)	Print the displayed image to a ps file.
print(HashPrintRequestAttributeSet attributes)	Print the displayed image to a ps file.
print(String path, String orientation)	Prints the displayed image to a ps file.
save()	Pops up a dialog to save your image.
saveAsJPG(String filename)	Saves the display as a jpg file.
saveAsPS(String filename)	Saves the display as a ps file.
saveScreenshot(String filename)	Save the file as a screenshot.
saveCurrentView(String filename)	Saves the current view of the image.

Methods
<code>setColortable(String colortableName)</code> Sets the color table.
<code>setColortable(String colortableName, String intensityName)</code> Sets the color table.
<code>setColortable(String colortableName, String intensityName, String scaleName)</code> Sets the color table.
<code>setCutLevels(double percent)</code> Sets the cut levels.
<code>setCutLevelsMin(double min)</code> Sets the minimum cut levels.
<code>setCutLevelsMax(double max)</code> Sets the maximum cut levels.
<code>setCutLevels(double min, double max)</code> Sets the cut levels.
<code>setCutLevels()</code> Sets the cut levels.
<code>setCutLevels(double[] minmax)</code> Sets the cut levels
<code>setUnit(Unit u)</code> Sets the unit.
<code>setZoomFactor(float zoomFactor)</code> Sets the zoom factor.
<code>zoom(double row, double column, float zoomFactor)</code> Zooms the image.
<code>zoomWorldCoordinates(double ra, double decl, float zoomFactor)</code> Zooms the image.
<code>zoomIn()</code> Zooms the image in.
<code>zoomOut()</code> Zoom the image out.
<code>zoomFit()</code> Zoom the image.
<code>flipYAxis()</code> Flips the y axis of the displayed image.
<code>setFlipYAxis(boolean flipAxis)</code> Sets whether the current image should be flipped.
<code>getFlipYAxis()</code> Returns whether the Y axis is flipped.
<code>isFlipped()</code> Returns whether the current layer is flipped.
<code>setDepthAxis(int depthAxis)</code>

Methods
Sets the depth axis for cubes.
getDepthAxis()
Returns the depth axis for cubes.

API details

Constructors

Display()
The standard constructor
The standard constructor for Display. This constructor shows an empty display window.

Display(boolean useAsComponent, boolean imageVisible)
A Display constructor
A constructor where you have the possibility to display the image in the background (which means the image will not be shown) or to use the display as a component to include in a gui in jide.
Arguments
boolean useAsComponent [INPUT, MANDATORY]
boolean imageVisible [INPUT, MANDATORY]

Display(boolean useAsComponent)
A constructor to use Display as a component.
A constructor where you have the possibility to use the display as a component to include in a gui in jide.
Argument
boolean useAsComponent [INPUT, MANDATORY]
Example
Example how to use Display as a component
<pre> from javax.swing import * from java.awt import BorderLayout dc = Display(True) frame = JFrame() frame.setTitle("Image display using Components") frame.setSize(800, 600) # The main component, with the image component = dc.getComponent() # The zoom, panner, status and colorbar components zoomComponent = dc.getZoomComponent() pannerComponent = dc.getPannerComponent() statusComponent = dc.getStatusPanelComponent() colorbarComponent = dc.getColorBarComponent() # Adding the components to the frame frame.getContentPane().add(component, BorderLayout.CENTER) frame.getContentPane().add(statusComponent, BorderLayout.NORTH) frame.getContentPane().add(colorbarComponent, BorderLayout.SOUTH) frame.show() dc.setImage(image) </pre>

Display(Image imds)
A constructor which displays an Image.

Display(Image imds)
A constructor which immediately displays the given Image.
Argument
Image imds [INPUT, MANDATORY]
Display(Cube cube)
A constructor which displays a Cube.
A constructor which immediately displays the given Cube.
Argument
Cube cube [INPUT, MANDATORY]
Display(Stack stack)
A constructor which displays a Stack.
A constructor which immediately displays the given Stack.
Argument
Stack stack [INPUT, MANDATORY]
Display(RgbImage imds)
A constructor which displays an RgbImage.
A constructor which immediately displays the given RgbImage.
Argument
RgbImage imds [INPUT, MANDATORY]
Display(Image imds, boolean imageVisible)
A Display constructor
A constructor which immediately displays the given Image. It is possible to not yet display the Image.
Arguments
Image imds [INPUT, MANDATORY]
boolean imageVisible [INPUT, MANDATORY]
Display(Cube imds, boolean imageVisible)
A Display constructor
A constructor which immediately displays the given Cube. It is possible to not yet display the Cube.
Arguments
Cube imds [INPUT, MANDATORY]
boolean imageVisible [INPUT, MANDATORY]
Display(Stack stack, boolean imageVisible)
A Display constructor

Display(Stack stack, boolean imageVisible)
A constructor which immediately displays the given Stack. It is possible to not yet display the Stack.
Arguments
Stack stack [INPUT, MANDATORY]
boolean imageVisible [INPUT, MANDATORY]
Display(Array2dData data)
A constructor to display a Numeric2d.
A constructor which immediately displays the given numeric2d
Argument
Array2dData data [INPUT, MANDATORY]
Display(Array2dData data, boolean imageVisible)
A Display constructor
A constructor which immediately displays the given numeric2d. It is possible to not yet display the image.
Arguments
Array2dData data [INPUT, MANDATORY]
boolean imageVisible [INPUT, MANDATORY]
Display(Array3dData data)
A constructor to display a numeric3d
A constructor which immediately displays the given numeric3d
Argument
Array3dData data [INPUT, MANDATORY]
Display(Array3dData data, boolean imageVisible)
A display constructor
A constructor which immediately displays the given numeric3d
Arguments
Array3dData data [INPUT, MANDATORY]
boolean imageVisible [INPUT, MANDATORY]

Methods

setSize(int x, int y)
setSize
Adapt the dimensions of the frame.
Arguments
int x [INPUT, MANDATORY]
int y [INPUT, MANDATORY]

int <code>getLayerShown()</code>
Returns the number of the shown layer
Returns the number of the layer which is shown.
Return
int
The number of the layer which is shown.
setVisible(boolean imageVisible)
Toggles the display visible or invisible.
Toggles the imageDisplay visible or invisible. True to make the image visible, False to hide the image.
Argument
boolean imageVisible [INPUT, MANDATORY]
JPanel <code>getComposedComponent()</code>
Returns a panel with all components of this Display on it.
Returns a panel with all components of this Display on it.
Return
JPanel
Returns a panel with all component of this Display on it.
ImageColorbar <code>getColorBarComponent()</code>
Returns the color bar component.
Returns a JComponent that contains the color bar. The color bar can be used to create your own component based Display.
Return
ImageColorbar
The ImageColorbar to use in you own components.
DivaMainImageDisplay <code>getComponent()</code>
Returns the component where the image is displayed.
Returns the component where the image is shown. This can be used to make your own GUI's with an image.
Return
DivaMainImageDisplay
The component to use in your own GUI's
Example
Example on how to integrate Display in your own GUI.

DivaMainImageDisplay getComponent()

```

from javax.swing import * from java.awt import BorderLayout
dc = Display(True)
frame = JFrame() frame.setTitle("Image display using Components")
frame.setSize(800, 600)
# The main component, with the image
component = dc.getComponent()
# The zoom, panner, status and colorbar components
zoomComponent = dc.getZoomComponent()
pannerComponent = dc.getPannerComponent()
statusComponent = dc.getStatusPanelComponent()
colorbarComponent = dc.getColorBarComponent()
# Adding the components to the frame
frame.getContentPane().add(component, BorderLayout.CENTER)
frame.getContentPane().add(statusComponent, BorderLayout.NORTH)
frame.getContentPane().add(colorbarComponent, BorderLayout.SOUTH)
frame.show()
dc.setImage(image)

```

ImagePanner getPannerComponent()

Returns the panner component with the overview of the image.

Returns the component which contains the overview of the image (100x100). This can be used to make your own GUI's with an image.

Return

ImagePanner

The component with the overview to use in your own GUI's

StatusPanel getStatusPanelComponent()

Returns the component with the status panel.

Returns the component which contains the status bar of the image. The status exists of zoom buttons, magnification, pixel coordinates, pixel intensity and sky coordinates. This can be used to make your own GUI's with an image.

Return

StatusPanel

The component with the status panel to use in your own GUI's

setStatusPanelComponent(StatusPanel imageDisplayStatusPanel)

Set a new status panel for the display.

Attaches a new status panel to the display.

Argument

StatusPanel **imageDisplayStatusPanel** [INPUT, MANDATORY]

ImageZoom getZoomComponent()

Returns the component with the zoomed view.

Returns the component which contains the a zoomed view of the image. The zoomed view has a zoom factor of 4. This component can be used to make your own GUI's with an image.

Return

ImageZoom <code>getZoomComponent()</code>
<p>ImageZoom</p> <p>The component with the zoomed view to use in your own GUI's</p>
setImage(Array2dData imageData)
<p>Set a new numeric2d image.</p> <p>Sets a new image on the display. A Bool2d, Int2d, Double2d, ... can be used.</p> <p>Argument</p> <p>Array2dData imageData [INPUT, MANDATORY]</p>
setImage(Array3dData imageData)
<p>Sets a new numeric3d image.</p> <p>Shows a new image on the display. A Bool3d, Int3d, Double3d, ... can be used. The first image is show, the other images are used as extra layers.</p> <p>Argument</p> <p>Array3dData imageData [INPUT, MANDATORY]</p>
setImage(Image imds)
<p>Sets a new Image to Display.</p> <p>Shows a new image on the display. The world-coordinate system of the Image is used.</p> <p>Argument</p> <p>Image imds [INPUT, MANDATORY]</p>
setImage(Cube cube)
<p>Sets a new cube to display.</p> <p>A Cube is shown on the display. The world-coordinate system of the Cube is used.</p> <p>Argument</p> <p>Cube cube [INPUT, MANDATORY]</p>
setImage(Stack stack)
<p>Sets a new stack to display.</p> <p>A Stack is shown on the display. The world-coordinate system of the Cube is used.</p> <p>Argument</p> <p>Stack stack [INPUT, MANDATORY]</p>
setImage(RgbImage imds)
<p>Sets a new RgbImage to Display.</p> <p>Shows a new image on the display.</p> <p>Argument</p> <p>RgbImage imds [INPUT, MANDATORY]</p>

addLayer(Layer layer)

Adds a new layer to the display.

Adds a new layer to the display. A Layer must be constructed before.

Argument

Layer **layer** [INPUT, MANDATORY]

addLayer(Array2dData imageData)

Adds a new layer with numeric2d data

Adds a new layer to the display. A Bool2d, Int2d, Double2d, ... can be used.

Argument

Array2dData **imageData** [INPUT, MANDATORY]

addLayer(Array3dData imageData)

Adds a new layer with numeric3d data

Adds new layers to the display. A Bool3d, Int3d, Double3d, ... can be used.

Argument

Array3dData **imageData** [INPUT, MANDATORY]

addLayer(Image imds)

Adds a new layer with an Image to the display.

Adds a new layer to the display. An Image is added as extra layer of the display.

Argument

Image **imds** [INPUT, MANDATORY]

addLayer(RgbImage imds)

Adds a new layer with an RgbImage to the display.

Adds a new layer to the display. An RgbImage is added as extra layer of the display.

Argument

RgbImage **imds** [INPUT, MANDATORY]

addLayer(Cube cube)

Adds a new layer with a Cube to the display.

Adds a new layer to the display. A Cube is added as extra layer of the display.

Argument

Cube **cube** [INPUT, MANDATORY]

addLayer(Stack stack)

Adds a new layer with a Stack to the display.

Adds a new layer to the display. A Stack is added as extra layer of the display.

Argument

addLayer(Stack stack)
Stack stack [INPUT, MANDATORY]
Layer getLayer()
Returns the shown layer.
Returns the Layer that is shown.
Return
Layer
The layer that is shown at the moment.
Layer getLayer(int i)
Returns the Layer object
The Layer with the given index is returned.
Argument
int i [INPUT, MANDATORY]
Return
Layer
The layer with the given index.
showLayer(int i)
Shows a chosen layer.
Shows the Layer with the given index. The Layer with the given index is shown.
Argument
int i [INPUT, MANDATORY]
removeLayer()
Removes the current layer.
Removes the Layer that is shown.
removeLayer(int i)
Removes a layer.
Removes the given Layer.
Argument
int i [INPUT, MANDATORY]
updateImage(Array2dData imageData)
Updates the layer with numeric2d data.
Updates the layer of the image that is displayed. A numeric2d will be used as input for the new layer. The annotations, cut levels and zoom factor are kept.
Argument

updateImage(Array2dData imageData)
Array2dData imageData [INPUT, MANDATORY]
updateImage(Image imds)
Updates the layer with an Image.
Updates the layer of the image that is displayed. An Image will be used as input for the new layer. The annotations, cut levels and zoom factor are kept.
Argument
Image imds [INPUT, MANDATORY]
updateImage(RgbImage imds)
Updates the layer with an Image.
Updates the layer of the image that is displayed. An RgbImage will be used as input for the new layer. The annotations, cut levels and zoom factor are kept.
Argument
RgbImage imds [INPUT, MANDATORY]
updateImage(Array3dData imageData)
Updates the layer with numeric3d data.
Updates the layer of the image that is displayed. A numeric3d will be used as input for the new layer. The annotations, cut levels and zoom factor are kept.
Argument
Array3dData imageData [INPUT, MANDATORY]
updateImage(Cube imds)
Updates the layer with a cube.
Updates the layer of the image that is displayed. A cube will be used as input for the new layer. The annotations, cut levels and zoom factor are kept.
Argument
Cube imds [INPUT, MANDATORY]
updateImage(Stack stack)
Updates the layer with a stack.
Updates the layer of the image that is displayed. A Stack will be used as input for the new layer. The annotations, cut levels and zoom factor are kept.
Argument
Stack stack [INPUT, MANDATORY]
setBackground(Color bgColor)
Sets the background color.
Changes the background of the image. The background can be black or white.
Argument
Color bgColor [INPUT, MANDATORY]

annotationToolbox()
Fires up the annotation toolbox.
Returns the AnnotationToolbox for the display. The AnnotationToolbox lets the user put annotation on the image, like ellipses, rectangles, text, ...
CanvasFigure addAnnotation(String text, double row, double column)
Adds a text annotation.
Adds a text annotation A text annotation will be placed, starting at the given pixel coordinates.
Arguments
String text [INPUT, MANDATORY]
double row [INPUT, MANDATORY]
double column [INPUT, MANDATORY]
Return
CanvasFigure
The annotation as a CanvasFigure
CanvasFigure addAnnotationWorldCoordinates(String text, double ra, double decl)
Adds a text annotation
A text annotation will be placed, starting at the given sky coordinates.
Arguments
String text [INPUT, MANDATORY]
double ra [INPUT, MANDATORY]
double decl [INPUT, MANDATORY]
Return
CanvasFigure
The annotation as a CanvasFigure
CanvasFigure addGreekAnnotation(String text, double row, double column)
Adds a Greek annotation
A greek text annotation will be placed, starting at the given pixel coordinates. All ascii text will be converted to greek characters : a becomes alpha, b becomes beta, ...
Arguments
String text [INPUT, MANDATORY]
double row [INPUT, MANDATORY]
double column [INPUT, MANDATORY]
Return
CanvasFigure
The annotation as a CanvasFigure

[CanvasFigure](#) addGreekAnnotationWorldCoordinates([String](#) text, double ra, double decl)

Adds a Greek annotation

A greek text annotation will be placed, starting at the given world coordinates. All ascii text will be converted to greek characters : a becomes alpha, b becomes beta, ...

Arguments

[String](#) **text** [INPUT, MANDATORY]

double **ra** [INPUT, MANDATORY]

double **decl** [INPUT, MANDATORY]

Return

[CanvasFigure](#)

The annotation as a CanvasFigure

[Font](#) getAnnotationFont()

Returns the default font for annotations.

Returns the font used for the annotations.

Return

[Font](#)

The font used for the annotations.

[Font](#) getAnnotationFont(double row, double column)

Returns the font for the specified annotation.

Returns the font used for the annotation that begins at the given pixel coordinates.

Arguments

double **row** [INPUT, MANDATORY]

double **column** [INPUT, MANDATORY]

Return

[Font](#)

The font used for the specified annotation.

[Font](#) getAnnotationFontWorldCoordinates(double ra, double dec)

Returns the font for the specified annotation.

Returns the font used for the annotation that begins at the given world coordinates.

Arguments

double **ra** [INPUT, MANDATORY]

double **dec** [INPUT, MANDATORY]

Return

[Font](#)

The font used for the specified annotation.

[Color](#) getAnnotationFontColor()

Returns the default color for annotations.

Returns the default color used for the text annotations.

Return**[Color](#)**

The default color used for the text annotations.

[Color](#) getAnnotationFontColor(double row, double column)

Returns the color of the specified annotation.

Returns the font color used for the annotation that begins at the given pixel coordinates.

Arguments

double **row** [INPUT, MANDATORY]

double **column** [INPUT, MANDATORY]

Return**[Color](#)**

The color used for the specified text annotation.

[Color](#) getAnnotationFontColorWorldCoordinates(double ra, double dec)

Returns the color of the specified annotation.

Returns the color used for the annotation that begins at the given world coordinates.

Arguments

double **ra** [INPUT, MANDATORY]

double **dec** [INPUT, MANDATORY]

Return**[Color](#)**

The color used for the specified text annotation.

removeAnnotation(double row, double column)

Removes the specified annotation

Removes the annotation at the given pixel coordinates.

Arguments

double **row** [INPUT, MANDATORY]

double **column** [INPUT, MANDATORY]

removeAnnotationWorldCoordinates(double ra, double dec)

Remove the specified annotation.

Removes the annotation at the given world coordinates.

Arguments

```
removeAnnotationWorldCoordinates(double ra, double dec)
```

```
double ra [INPUT, MANDATORY]
double dec [INPUT, MANDATORY]
```

```
removeAnnotation(CanvasFigure fig)
```

Removes the specified annotation.

Removes the given annotation.

Argument

```
CanvasFigure fig [INPUT, MANDATORY]
```

```
removeAnnotations()
```

Removes all annotations.

Removes all the annotations from the display.

```
setAnnotationFont(Font f)
```

Changes the font of all Annotations.

Changes the font of all annotations that are visible on the Display.

Argument

```
Font f [INPUT, MANDATORY]
```

```
setAnnotationFont(double row, double column, Font f)
```

Changes the font of the specified annotations.

Changes the font of the annotations which are located at the given pixel coordinates.

Arguments

```
double row [INPUT, MANDATORY]
double column [INPUT, MANDATORY]
Font f [INPUT, MANDATORY]
```

```
setAnnotationFontWorldCoordinates(double ra, double dec, Font f)
```

Changes the font of the specified annotations.

Changes the font of the annotations which are located at the given world coordinates.

Arguments

```
double ra [INPUT, MANDATORY]
double dec [INPUT, MANDATORY]
Font f [INPUT, MANDATORY]
```

```
setAnnotationFont(int size)
```

Changes the font size of all annotations.

Changes the font size of all annotations.

Argument

setAnnotationFont(int size)

int **size** [INPUT, MANDATORY]

setAnnotationFont(double row, double column, int size)

Changes the font size of the specified annotations.

Changes the font size of the annotations which are located at the given pixel coordinates.

Arguments

double **row** [INPUT, MANDATORY]

double **column** [INPUT, MANDATORY]

int **size** [INPUT, MANDATORY]

setAnnotationFontWorldCoordinates(double ra, double dec, int size)

Changes the font size of the specified annotations.

Changes the font size of the annotations which are located at the given world coordinates.

Arguments

double **ra** [INPUT, MANDATORY]

double **dec** [INPUT, MANDATORY]

int **size** [INPUT, MANDATORY]

setAnnotationFontColor(Color color)

Sets the default color for annotations.

Changes the font color of all annotations.

Argument

[Color](#) **color** [INPUT, MANDATORY]

setAnnotationFontColor(double row, double column, Color color)

Sets the font color for the specified annotation.

Changes the font color of the annotations which are located at the given pixel coordinates.

Arguments

double **row** [INPUT, MANDATORY]

double **column** [INPUT, MANDATORY]

[Color](#) **color** [INPUT, MANDATORY]

setAnnotationFontColorWorldCoordinates(double ra, double dec, Color color)

Sets the font color for the specified annotation.

Changes the font color of the annotations which are located at the given world coordinates.

Arguments

double **ra** [INPUT, MANDATORY]

double **dec** [INPUT, MANDATORY]

```
setAnnotationFontColorWorldCoordinates(double ra, double dec,
Color color)
```

```
Color color [INPUT, MANDATORY]
```

```
CanvasFigure addEllipse(double row, double column, double w, double h, float lineWidth, Color color)
```

Adds an ellipse.

Adds an ellipse to the image at a given place, with a given dimension, line width and color.

Arguments

```
double row [INPUT, MANDATORY]
double column [INPUT, MANDATORY]
double w [INPUT, MANDATORY]
double h [INPUT, MANDATORY]
float lineWidth [INPUT, MANDATORY]
Color color [INPUT, MANDATORY]
```

Return

[CanvasFigure](#)

The annotation as CanvasFigure

```
addFigure(CanvasFigure fig)
```

Adds a CanvasFigure

Adds the given CanvasFigure to the image.

Argument

```
CanvasFigure fig [INPUT, MANDATORY]
```

```
CanvasFigure addEllipse(double row, double column, double w, double h, float lineWidth, Color color, double positionAngle)
```

Adds an ellipse

Adds an ellipse to the image at a given place, with a given dimension, line width, color and position angle.

Arguments

```
double row [INPUT, MANDATORY]
double column [INPUT, MANDATORY]
double w [INPUT, MANDATORY]
double h [INPUT, MANDATORY]
float lineWidth [INPUT, MANDATORY]
Color color [INPUT, MANDATORY]
double positionAngle [INPUT, MANDATORY]
```

Return

[CanvasFigure](#)

The annotation as CanvasFigure

CanvasFigure addCircle(double row, double column, double radius, float lineWidth, [Color](#) color)

Adds a circle

Adds a circle to the image at a given place, with a given dimension, line width and color.

Arguments

double **row** [INPUT, MANDATORY]
double **column** [INPUT, MANDATORY]
double **radius** [INPUT, MANDATORY]
float **lineWidth** [INPUT, MANDATORY]
[Color](#) **color** [INPUT, MANDATORY]

Return

[CanvasFigure](#)

The annotation as CanvasFigure

CanvasFigure addLine(double row1, double column1, double row2, double column2, float lineWidth, [Color](#) color)

Adds a line.

Adds an line to the image at a given place, with a given line width and color.

Arguments

double **row1** [INPUT, MANDATORY]
double **column1** [INPUT, MANDATORY]
double **row2** [INPUT, MANDATORY]
double **column2** [INPUT, MANDATORY]
float **lineWidth** [INPUT, MANDATORY]
[Color](#) **color** [INPUT, MANDATORY]

Return

[CanvasFigure](#)

The annotation as CanvasFigure

CanvasFigure addPolygon(double[] coords, float lineWidth, [Color](#) color)

Adds a polygon

Adds a polygon to the image at a given place, with a given line width and color. The coordinates should be given as [row1, column1, row2, column2, ...]

Arguments

double[] **coords** [INPUT, MANDATORY]
float **lineWidth** [INPUT, MANDATORY]
[Color](#) **color** [INPUT, MANDATORY]

Return

[CanvasFigure](#)

The annotation as CanvasFigure

[CanvasFigure](#) addPolyline(double[] coords, float lineWidth, [Color](#) color)

Adds a polyline.

Adds a polyline to the image at a given place, with a given line width and color. The coordinates should be given as [row1, column1, row2, column2, ...]

Arguments

double[] **coords** [INPUT, MANDATORY]

float **lineWidth** [INPUT, MANDATORY]

[Color](#) **color** [INPUT, MANDATORY]

Return

[CanvasFigure](#)

The annotation as CanvasFigure

[CanvasFigure](#) addRectangle(double minRow, double minColumn, double w, double h, float lineWidth, [Color](#) color)

Adds a rectangle.

Adds an rectangle to the image at a given place, with a given dimension, line width and color.

Arguments

double **minRow** [INPUT, MANDATORY]

double **minColumn** [INPUT, MANDATORY]

double **w** [INPUT, MANDATORY]

double **h** [INPUT, MANDATORY]

float **lineWidth** [INPUT, MANDATORY]

[Color](#) **color** [INPUT, MANDATORY]

Return

[CanvasFigure](#)

The annotation as CanvasFigure

addCompass(double row, double column, double size, float lineWidth, [Color](#) color)

Adds a compass with the north and east directions.

Adds a compass with east and north.

Arguments

double **row** [INPUT, MANDATORY]

double **column** [INPUT, MANDATORY]

double **size** [INPUT, MANDATORY]

float **lineWidth** [INPUT, MANDATORY]

[Color](#) **color** [INPUT, MANDATORY]

addTenArcSecs(double row, double column, float lineWidth, [Color](#) color)

Adds a line which is 10 arc seconds to the image.

```
addTenArcSecs(double row, double column, float lineWidth, Color color)
```

Adds a line which is 10 arc seconds.

Arguments

```
double row [INPUT, MANDATORY]
double column [INPUT, MANDATORY]
float lineWidth [INPUT, MANDATORY]
Color color [INPUT, MANDATORY]
```

```
ArrayList addImageContour(ImageContour imageContour, ArrayList colors, int minLength)
```

Adds an ImageContour.

Draws the Contours of the given ImageContour in the given Color on the image (if their length is at least the given minimum length) and returns the drawn Contours as an ArrayList of ImageFigures.

Arguments

```
ImageContour imageContour [INPUT, MANDATORY]
ArrayList colors [INPUT, MANDATORY]
int minLength [INPUT, MANDATORY]
```

Return

[ArrayList](#)

The drawn Contours as an ArrayList of ImageFigures.

```
ArrayList addImageContour(ImageContour imageContour, int minLength)
```

Adds an ImageContour.

Draws the Contours of the given ImageContour on the image (if their length is at least the given minimum length) and returns the drawn Contours as an ArrayList of ImageFigures.

Arguments

```
ImageContour imageContour [INPUT, MANDATORY]
int minLength [INPUT, MANDATORY]
```

Return

[ArrayList](#)

The drawn Contours as an ArrayList of ImageFigures.

```
ArrayList addImageContour(ImageContour imageContour, ArrayList colors)
```

Adds an ImageContour.

Draws the Contours of the given ImageContour in the given Colors on the image and returns the drawn Contours as an ArrayList of ImageFigures.

Arguments

```
ImageContour imageContour [INPUT, MANDATORY]
```

[ArrayList](#) addImageContour([ImageContour](#) imageContour, [ArrayList](#) colors)

[ArrayList](#) colors [INPUT, MANDATORY]

Return

[ArrayList](#)

Returns the drawn Contours as an ArrayList of ImageFigures.

[ArrayList](#) addImageContour([ImageContour](#) imageContour)

Adds an ImageContour.

Draws the Contours of the given ImageContour on the image and returns the drawn Contours as an ArrayList of ImageFigures.

Argument

[ImageContour](#) imageContour [INPUT, MANDATORY]

Return

[ArrayList](#)

Returns the drawn Contours as an ArrayList of ImageFigures.

[ArrayList](#) addWcsImageContour([ImageContour](#) imageContour, [ArrayList](#) colors, int minLength)

Add an ImageContour.

Draws the Contours of the given ImageContour in the given Color on the image based on the Wcs (if their length is at least the given minimum length) and returns the drawn Contours as an ArrayList of ImageFigures.

Arguments

[ImageContour](#) imageContour [INPUT, MANDATORY]

[ArrayList](#) colors [INPUT, MANDATORY]

int minLength [INPUT, MANDATORY]

Return

[ArrayList](#)

The drawn Contours as an ArrayList of ImageFigures.

[ArrayList](#) addWcsImageContour([ImageContour](#) imageContour, int minLength)

Add an ImageContour.

Draws the Contours of the given ImageContour on the image based on the Wcs (if their length is at least the given minimum length) and returns the drawn Contours as an ArrayList of ImageFigures.

Arguments

[ImageContour](#) imageContour [INPUT, MANDATORY]

int minLength [INPUT, MANDATORY]

Return

[ArrayList](#)

```
ArrayList addWcsImageContour(ImageContour imageContour, int min-  
Length)
```

The drawn Contours as an ArrayList of ImageFigures.

```
ArrayList addWcsImageContour(ImageContour imageContour, ArrayList  
contourColors)
```

Adds an ImageContour.

Draws the Contours of the given ImageContour in the given Color on the image based on the Wcs and returns the drawn Contours as an ArrayList of ImageFigures.

Arguments

ImageContour imageContour [INPUT, MANDATORY]

ArrayList contourColors [INPUT, MANDATORY]

Return

ArrayList

The drawn Contours as an ArrayList of ImageFigures.

```
ArrayList addWcsImageContour(ImageContour imageContour)
```

Adds an ImageContour.

Draws the Contours of the given ImageContour on the image based on the Wcs and returns the drawn Contours as an ArrayList of ImageFigures.

Argument

ImageContour imageContour [INPUT, MANDATORY]

Return

ArrayList

The drawn Contours as an ArrayList of ImageFigures.

```
ArrayList addContourLevel(ContourLevel contourLevel, Color con-  
tourColor, int minimumContourLength)
```

Adds a ContourLevel.

Draws the Contours of the given ContourLevel in the given Color on the image (if their length is at least the given minimum length) and returns the drawn Contours as an ArrayList of ImageFigures.

Arguments

ContourLevel contourLevel [INPUT, MANDATORY]

Color contourColor [INPUT, MANDATORY]

int **minimumContourLength** [INPUT, MANDATORY]

Return

ArrayList

Returns the drawn Contours as an ArrayList of ImageFigures.

```
ArrayList addContourLevel(ContourLevel contourLevel, int minimum-  
ContourLength)
```

Adds a ContourLevel.

ArrayList addContourLevel(**ContourLevel** contourLevel, int minimumContourLength)

Draws the Contours of the given ContourLevel on the image (if their length is at least the given minimum length) and returns the drawn Contours as an ArrayList of ImageFigures.

Arguments

ContourLevel contourLevel [INPUT, MANDATORY]

int **minimumContourLength** [INPUT, MANDATORY]

Return

ArrayList

Returns the drawn Contours as an ArrayList of ImageFigures.

ArrayList addContourLevel(**ContourLevel** level, **Color** color)

Adds a ContourLevel.

Draws the Contours of the given ContourLevel in the given Color on the image and returns the drawn Contours as an ArrayList of ImageFigures.

Arguments

ContourLevel level [INPUT, MANDATORY]

Color color [INPUT, MANDATORY]

Return

ArrayList

The drawn Contours as an ArrayList of ImageFigures.

ArrayList addContourLevel(**ContourLevel** level)

Adds a ContourLevel.

Draws the Contours of the given ContourLevel on the image and returns the drawn Contours as an ArrayList of ImageFigures.

Argument

ContourLevel level [INPUT, MANDATORY]

Return

ArrayList

The drawn Contours as an ArrayList of ImageFigures.

ArrayList addContourLevel(**ContourLevel** level, **Wcs** wcs, **Color** color, int minLength)

Adds a ContourLevel.

Draws the Contours of the given ContourLevel in the given Color on the image (if their length is at least the given minimum length) and returns the drawn Contours as an ArrayList of ImageFigures.

Arguments

ContourLevel level [INPUT, MANDATORY]

Wcs wcs [INPUT, MANDATORY]

Color color [INPUT, MANDATORY]

ArrayList addContourLevel(**ContourLevel** level, **Wcs** wcs, **Color** color, int minLength)

int **minLength** [INPUT, MANDATORY]

Return

[ArrayList](#)

Returns the drawn Contours as an ArrayList of ImageFigures.

ArrayList addContourLevel(**ContourLevel** level, **Wcs** wcs, int minLength)

Adds a ContourLevel.

Draws the Contours of the given ContourLevel on the image (if their length is at least the given minimum length) and returns the drawn Contours as an ArrayList of ImageFigures.

Arguments

[ContourLevel](#) **level** [INPUT, MANDATORY]

[Wcs](#) **wcs** [INPUT, MANDATORY]

int **minLength** [INPUT, MANDATORY]

Return

[ArrayList](#)

Returns the drawn Contours as an ArrayList of ImageFigures.

ArrayList addContourLevel(**ContourLevel** level, **Wcs** wcs, **Color** color)

addContourLevel(ContourLevel level, Wcs wcs, Color color)

Draws the Contours of the given ContourLevel in the given Color on the image and returns the drawn Contours as an ArrayList of ImageFigures.

Arguments

[ContourLevel](#) **level** [INPUT, MANDATORY]

[Wcs](#) **wcs** [INPUT, MANDATORY]

[Color](#) **color** [INPUT, MANDATORY]

Return

[ArrayList](#)

Returns the drawn Contours as an ArrayList of ImageFigures.

ArrayList addContourLevel(**ContourLevel** level, **Wcs** wcs)

addContourLevel(ContourLevel level, Wcs wcs, Color color)

Draws the Contours of the given ContourLevel on the image and returns the drawn Contours as an ArrayList of ImageFigures.

Arguments

[ContourLevel](#) **level** [INPUT, MANDATORY]

[Wcs](#) **wcs** [INPUT, MANDATORY]

Return

ArrayList addContourLevel(ContourLevel level, Wcs wcs)
<p>ArrayList</p> <p>Returns the drawn Contours as an ArrayList of ImageFigures.</p>
ImageFigure addContour(Contour contour, Color color, int minLength)
<p>Adds a Contour.</p> <p>Draws the given Contour on the image (if its length is at least the given minimum length) and returns the drawn Contour as an ImageFigure.</p> <p>Arguments</p> <p>Contour contour [INPUT, MANDATORY]</p> <p>Color color [INPUT, MANDATORY]</p> <p>int minLength [INPUT, MANDATORY]</p> <p>Return</p> <p>ImageFigure</p> <p>Returns the drawn Contour as an ImageFigure.</p>
ImageFigure addContour(Contour contour, Color color)
<p>Adds a Contour</p> <p>Draws the given Contour in the given Color on the image and returns the drawn Contour as an ImageFigure.</p> <p>Arguments</p> <p>Contour contour [INPUT, MANDATORY]</p> <p>Color color [INPUT, MANDATORY]</p> <p>Return</p> <p>ImageFigure</p> <p>Returns the drawn Contour as an ImageFigure.</p>
ImageFigure addContour(Contour contour, Wcs wcs, Color color, int minLength)
<p>Adds a Contour.</p> <p>Draws the given Contour in the given Color on the image (if its length is at least the given minimum length) based on the Wcs and returns the drawn Contour as an ImageFigure.</p> <p>Arguments</p> <p>Contour contour [INPUT, MANDATORY]</p> <p>Wcs wcs [INPUT, MANDATORY]</p> <p>Color color [INPUT, MANDATORY]</p> <p>int minLength [INPUT, MANDATORY]</p> <p>Return</p> <p>ImageFigure</p> <p>Returns the drawn Contour as an ImageFigure.</p>

ImageFigure addContour(**Contour** contour, **Wcs** wcs, **Color** color)

Adds a Contour.

Draws the given Contour in the given Color on the image based on the Wcs and returns the drawn Contour as an ImageFigure.

Arguments

Contour contour [INPUT, MANDATORY]

Wcs wcs [INPUT, MANDATORY]

Color color [INPUT, MANDATORY]

Return

ImageFigure

Returns the drawn Contour as an ImageFigure.

ArrayList addPositionList(**PositionList** positionList, **Color** color)

Overlays the given source list on this Display,

in the given color.

Arguments

PositionList positionList [INPUT, MANDATORY]

Color color [INPUT, MANDATORY]

Return

ArrayList

Returns a list of sources.

ArrayList addPositionList(**PositionList** positionList)

Overlays the given source list on this Display,

in the given color.

Argument

PositionList positionList [INPUT, MANDATORY]

Return

ArrayList

Returns a list of sources.

ArrayList addSourceFit(**SourceFittingProduct** source, **Color** color)

Visualises the given SourceFittingProduct on this Display in the given color.

Returns a list of ellipses/circles.

Arguments

SourceFittingProduct source [INPUT, MANDATORY]

Color color [INPUT, MANDATORY]

Return

ArrayList

[ArrayList](#) addSourceFit([SourceFittingProduct](#) source, [Color](#) color)

Returns a list of ellipses/circles.

[ArrayList](#) addSourceFit([SourceFittingProduct](#) source)

Visualises the given SourceFittingProduct on this Display.

Returns a list of ellipses/circles.

Argument

[SourceFittingProduct](#) **source** [INPUT, MANDATORY]

Return

[ArrayList](#)

Returns a list of ellipses/circles.

[ArrayList](#) showAxes(boolean showAxis)

Enables or disables the axes for the displayed image.

Enables or disables the axes for the displayed image. If the parameter is true, two axes are drawn, one on the left side and one at the bottom. An array with the axes is returned.

Argument

boolean **showAxis** [INPUT, MANDATORY]

Return

[ArrayList](#)

An ArrayList with the axes.

[ImageAxis](#) getLeftaxis()

Returns the left axis.

Returns the left axis of the Display. If there is no left axis, the standard left axis is made, shown and returned.

Return

[ImageAxis](#)

The left axis

[ImageAxis](#) getRightaxis()

Returns the right axis.

Returns the right axis of the Display. If there is no right axis, the standard right axis is made, shown and returned.

Return

[ImageAxis](#)

The right axis

[ImageAxis](#) getBottomaxis()

Returns the bottom axis.

[ImageAxis](#) getBottomaxis()

Returns the bottom axis of the Display. If there is no bottom axis, the standard bottom axis is made, shown and returned.

Return

[ImageAxis](#)

The bottom axis

[ImageAxis](#) getTopaxis()

Returns the top axis.

Returns the top axis of the Display. If there is no top axis, the standard top axis is made, shown and returned.

Return

[ImageAxis](#)

The top axis

[ArrayList](#) getAxes()

Returns an array with the axes.

Returns an array with the axes. Using methods on the Axes, the appearance can be changed.

Return

[ArrayList](#)

An ArrayList with the Axes.

addAxis([String](#) label, Position orientation)

Adds new axis.

Adds a new axis.

Arguments

[String](#) label [INPUT, MANDATORY]

Position **orientation** [INPUT, MANDATORY]

addAxis([ImageAxis](#) axis)

Adds new axis.

Adds a new axis.

Argument

[ImageAxis](#) axis [INPUT, MANDATORY]

deleteAxis([ImageAxis](#) axis)

Deletes the given axis.

Deletes the given axis.

Argument

deleteAxis(ImageAxis axis)
ImageAxis axis [INPUT, MANDATORY]
setCenter(int x, int y)
setCenter
Sets the (geometrical, not as defined in the Wcs) center of the image to a certain pixel on the display
Arguments
int x [INPUT, MANDATORY]
int y [INPUT, MANDATORY]
close()
Closes the display.
Closes the display and frees the memory.
editColors()
Edit the colors using a popup.
A windows opens where you can change the color table, intensity and scale.
editCutLevels()
Edit the cut levels using a popup.
A windows opens where you can set the cut levels of the image.
String getColortable()
Returns the name of the color table.
Returns the name of the color table
Return
String
The name of the color table
double[] getCutLevels()
Returns the cut levels.
Returns an array with the cut levels of the displayed image.
Return
double[]
The cut levels of the displayed image.
Image getImage()
Returns the displayed image.
Returns the displayed Image.

Image <code>getImage()</code>
Return Image The displayed Image
RgbImage <code>getRgbImage()</code>
Returns the displayed image. Returns the displayed Image. Return RgbImage The displayed Image
double[] <code>getPixelCoordinates(double c1, double c2)</code>
Returns the image coordinates Returns the image coordinates of the given world coordinates Arguments double c1 [INPUT, MANDATORY] double c2 [INPUT, MANDATORY] Return double[] The corresponding image coordinates as a 2 dimensional array of doubles, the first one describing the row and the second the column.
Number <code>getIntensity(int row, int column)</code>
Returns the intensity of the pixel Returns the intensity of the pixel with given pixel coordinates Arguments int row [INPUT, MANDATORY] int column [INPUT, MANDATORY] Return Number The intensity of the given pixel
Number <code>getIntensityFromWorldCoordinates(double c1, double c2)</code>
Returns the intensity of the pixel. Returns the intensity of the pixel with given world coordinates. Arguments double c1 [INPUT, MANDATORY] double c2 [INPUT, MANDATORY]

Number `getIntensityFromWorldCoordinates(double c1, double c2)`

Return

Number

double The intensity of the given pixel.

Unit `getUnit()`

Returns the unit.

Returns the unit of the image

Return

Unit

The unit of the image.

float `getZoomFactor()`

Returns the used zoom factor.

Returns the used zoom factor. The returned zoomfactor is bigger than 1 if the image is zoomed in, otherwise, the zoomfactor is between 0 and 1

Return

float

The zoomfactor of the displayed image.

printDialog()

Opens a printer dialog.

A dialog is opened where you can choice the printer, number of copies, page setup and appearance.

getPrintControl()

Returns the printControl.

Returns the printControl to make it possible to print using the command line.

Example

How to print from the command line

```
job = d.getPrintJob()
job.setCopies(2) # Set the number of copies
pservices = job.lookupPrintServices()
job.setPrintService(pservices[3]) #Use the printer of choice
d.setPrintJob(job)
job.print()
```

createPrintJob()

Creates a print job.

Creates a print job.

PrinterJob `getPrintJob()`

Returns the printer job.

PrinterJob getPrintJob()

Returns the printer job.

Return

[PrinterJob](#)

The printerJob.

setPrintJob([PrinterJob](#) job)

Sets a printer job.

Set a given printer job.

Argument

[PrinterJob](#) job [INPUT, MANDATORY]

print([String](#) path)

Print the displayed image to a ps file.

Prints the displayed image to a ps file. The settings can be changed with the method setPrintJob(job) or with the GUI obtained from the method createPrintJob()

Argument

[String](#) path [INPUT, MANDATORY]

print([HashPrintRequestAttributeSet](#) attributes)

Print the displayed image to a ps file.

Prints the displayed image to a ps file.

Argument

[HashPrintRequestAttributeSet](#) attributes [INPUT, MANDATORY]

print([String](#) path, [String](#) orientation)

Prints the displayed image to a ps file.

Prints the displayed image to a ps file. The settings can be changed with the method setPrintJob(job) or with the GUI obtained from the method createPrintJob() The orientation of the page can be "landscape" (or "l"), "portrait" (or "p"), "reverse_landscape" (or "rl"), "reverse_portrait" (or "rp")

Arguments

[String](#) path [INPUT, MANDATORY]

[String](#) orientation [INPUT, MANDATORY]

save()

Pops up a dialog to save your image.

Pops up a file chooser and saves the image as fits, jpeg, tiff, png or bmp file.

saveAsJPG([String](#) filename)

Saves the display as a jpg file.

saveAsJPG(String filename)
Saves the display as a jpg file.
Argument
String filename [INPUT, MANDATORY]
saveAsPS(String filename)
Saves the display as a ps file.
Saves the display as a ps file.
Argument
String filename [INPUT, MANDATORY]
saveScreenshot(String filename)
Save the file as a screenshot.
Saves the file as screenshot. Saves the display as jpg, png, tiff, bmp or FITS.
Argument
String filename [INPUT, MANDATORY]
saveCurrentView(String filename)
Saves the current view of the image.
Saves the view as screenshot. The annotations are also saved!
Argument
String filename [INPUT, MANDATORY]
setColortable(String colortableName)
Sets the color table.
Sets the color table of the displayed image.
Argument
String colortableName [INPUT, MANDATORY]
setColortable(String colortableName, String intensityName)
Sets the color table.
Sets the colortable of the displayed image.
Arguments
String colortableName [INPUT, MANDATORY]
String intensityName [INPUT, MANDATORY]
setColortable(String colortableName, String intensityName, String scaleName)
Sets the color table.
Sets the colortable Sets the colortable of the displayed image

```
setColortable(String colortableName, String intensityName, String
scaleName)
```

Arguments

```
String colortableName [INPUT, MANDATORY]
```

```
String intensityName [INPUT, MANDATORY]
```

```
String scaleName [INPUT, MANDATORY]
```

```
setCutLevels(double percent)
```

Sets the cut levels.

Sets the cut level of the displayed image.

Argument

```
double percent [INPUT, MANDATORY]
```

```
setCutLevelsMin(double min)
```

Sets the minimum cut levels.

Sets the minimum cut level of the displayed image.

Argument

```
double min [INPUT, MANDATORY]
```

```
setCutLevelsMax(double max)
```

Sets the maximum cut levels.

Sets the maximum cut level of the displayed image.

Argument

```
double max [INPUT, MANDATORY]
```

```
setCutLevels(double min, double max)
```

Sets the cut levels.

Sets the cut level of the displayed image.

Arguments

```
double min [INPUT, MANDATORY]
```

```
double max [INPUT, MANDATORY]
```

```
setCutLevels()
```

Sets the cut levels.

Sets the cut level of the displayed image.

```
setCutLevels(double[] minmax)
```

Sets the cut levels

Sets the cut level of the displayed image.

Argument

```
double[] minmax [INPUT, MANDATORY]
```



setUnit(Unit u)
Sets the unit. Sets the unit of the displayed image. Argument Unit u [INPUT, MANDATORY]
setZoomFactor(float zoomFactor)
Sets the zoom factor. Sets the zoomfactor of the displayed image. Argument float zoomFactor [INPUT, MANDATORY]
zoom(double row, double column, float zoomFactor)
Zooms the image. Sets the zoomfactor of the displayed image and the coordinates which should appear in the center of the image. Arguments double row [INPUT, MANDATORY] double column [INPUT, MANDATORY] float zoomFactor [INPUT, MANDATORY]
zoomWorldCoordinates(double ra, double decl, float zoomFactor)
Zooms the image. Sets the zoomfactor of the displayed image and the coordinates which should appear in the center of the image. Arguments double ra [INPUT, MANDATORY] double decl [INPUT, MANDATORY] float zoomFactor [INPUT, MANDATORY]
zoomIn()
Zooms the image in. Zooms the displayed image. The zoom factor changes the following way : ...1/8, 1/4, 1/2, 1, 2, 4, 8, ...
zoomOut()
Zoom the image out. Zooms the displayed image out. The zoom factor changes the following way : ..., 8, 4, 2, 1, 1/2, 1/4, 1/8, ...
zoomFit()
Zoom the image.

zoomFit()
Zooms the image to a zoomfactor so that the image is shown in total on the screen. Axes are taken into account
flipYAxis()
Flips the y axis of the displayed image. If this method is called, the current axis is flipped. If the displayed image has the WCS keyword flipy, this method will have no effect.
setFlipYAxis(boolean flipAxis)
Sets whether the current image should be flipped. The current image will be flipped. Argument boolean flipAxis [INPUT, MANDATORY]
getFlipYAxis()
Returns whether the Y axis is flipped. Returns true if the current image is flipped.
isFlipped()
Returns whether the current layer is flipped. Returns true if the current layer is flipped.
setDepthAxis(int depthAxis)
Sets the depth axis for cubes. Set the depth axis for the current and newly added cubes. The first axis (0) is the standard depth axis. Argument int depthAxis [INPUT, MANDATORY]
getDepthAxis()
Returns the depth axis for cubes. Returns which axis is the depth axis.

See also

- [SimpleImage](#)
- [SimpleCube](#)

2.101. displayLog

Full Name:	herschel.ia.toolbox.util.DisplayLogTask
Alias:	displayLog
Type:	Java Task - 
Import:	from herschel.ia.toolbox.util import DisplayLogTask
Category:	task

Description

Open a window displaying log messages

The display task is used to open a log window handler and assign it to a log

Examples

Example 1: attach the window to a logger

```
# create a logger
log=java.util.logging.Logger.getLogger("")
# display the logger in the window
displayLog(log)
# sending logs
log.info("ahh, an info message")
log.warning("ehm, a warning message")
log.severe("oops! a severe message")
```

Example 2: remove the window attached to a logger (JIDE only)

```
# create a logger
log=java.util.logging.Logger.getLogger("")
# display the logger in the window
displayLog(log)
# sending logs
log.info("ahh, an info message")
# stop displaying messages from log by removing it from the window
displayLog(log, option="remove")
# no displayed any longer
log.info("ahh, an new info message")
```

Example 3: change the size (number of lines to be displayed) in the window (JIDE only)

```
displayLog(option="change", size=8000)
```

API Summary

Jython Syntax

```
displayLog(logger)
```

Properties

Logger `logger` [INPUT, No, default=`java.util.logging.Logger.getLogger("")`]

String `option` [INPUT, No, default=`"add"`]

Integer `size` [INPUT, No, default=`4096`]

Limitations

No time buffering, logs could clutter the java window system

Miscellaneous

No miscellaneous

API details

Properties

<code>Logger logger [INPUT, No, default=java.util.logging.Logger.getLogger("")]</code>
--

The log to display the messages into the window

<code>String option [INPUT, No, default="add"]</code>

Specify the action to be taken, default value is "add" which enables the display for the specified logger, the "remove" option stops displaying messages for an attached logger, the "change" option works in pair with parameter size.


<code>Integer size [INPUT, No, default=4096]</code>

Specify the number of lines to be displayed in the window, the parameter is effectively set up *only* if the "change" value for the parameter "option" is specified

History

- 2007-01-26 - NdC: first release

2.102. DivideSpectrum

Full Name:	herschel.ia.toolbox.spectrum.DivideSpectrum
Type:	Java Task - 
Import:	from herschel.ia.toolbox.spectrum import DivideSpectrum

Description

Task for dividing the flux data included in a spectrum container by a scalar or for dividing two spectrum containers on a spectrum-by-spectrum basis by each other.

For the scalar mode, 'ds' and 'param' need to be specified - for the pair-wise mode, 'ds1' and 'ds2' need to be set. In the pairwise mode, the task iterates through the two containers and computes the pair-operation. In case the size of the two containers is different, the iteration stops at the minimum size. Hence, the remaining spectra in the larger container are ignored. In case one of the datasets only contains a single spectrum, the task always refers to this single spectrum while iterating over all the spectra in the other dataset.

Different selection schemes are available for selecting the point spectra or the segments to be processed. The most simple scheme is to specify lists of point spectrum indices (`selection=[1, 3, 4, 2]`). In this case, the operation (scalar- or pair-operation) is just taken over the point spectra with corresponding indices. Instead of specifying a list of indices (which requires the knowledge of these) other recipes to specify a selection are available. Typical recipes test attribute values to match certain values or to be included in a given interval. These recipes we refer to as selection models. The different options to specify selections can be combined. Here, an AND logic is adopted.

Similarly, you can specify what segments to consider. In the most simple case, you can just specify the indices of segments to be considered (for the scalar operation `segments=[1, 3, 4]` and for the pair operation `segments1=[1, 3, 4]`, `segments2=[3, 6, 5]`). In more advanced situations you can use a `SegmentSelection` object. Note that the pairs of segments to be processed need to be consistent - otherwise the processing will fail.

You can specify whether the original datasets should be overwritten with the 'overwrite' flag. By default no data is overwritten. In the pair-wise mode, it is not always possible to overwrite the data - in particular if non-trivial segment selections are specified.

For the pair-wise mode, there are further options that can be selected by the 'variant' (e.g. whether to use flags and weights).

Example

Example 1: from Jide:

```
from herschel.ia.toolbox.spectrum import DivideSpectrum
divide = DivideSpectrum()
dividedByFactor = divide(ds=spectra, param=2.1)
dividedByFactor = divide(ds=spectra, selection={"bbtype": [6031]}, param=2.1)
dividedByFactor = divide(ds=spectra, selection=[0,1,2,3], param=2.1)
dividedByFactor = divide(ds=spectra, selection={"Chopper":
([-4.4, 5.9], 0.2), -"bbtype": [6031]}, param=2.1)
dividedByFactor = divide(ds=spectra, selection={"LoFrequency":
(4000.0, 5000.0)}, param=2.1)
dividedPairWise = divide(ds1=spectral1, ds2=spectra2)
dividedPairWise = divide(ds1=spectral1, ds2=spectra2, selection={"bbtype":
[6031]})
dividedPairWise = divide(ds1=spectral1, ds2=spectra2, selection=[0,1,2,3])
dividedPairWise = divide(ds1=spectral1, ds2=spectra2, selection={"Chopper":
([-4.4, 5.9], 0.2), -"bbtype": [6031]})
```

Example 1: from Jide:

```
dividedPairWise = divide(ds1=spectral, ds2=spectra2, selection={"LoFrequency":
(4000.0,5000.0)})
```

API Summary

Properties
SpectrumContainer ds [INPUT, OPTIONAL, default=no default value.]
Double param [INPUT, OPTIONAL, default=no default value.]
Object selection [INPUT, OPTIONAL, default=None.]
PyDictionary Map> selection_lookup [INPUT, OPTIONAL, default=no default value.]
PyList selection_index [INPUT, OPTIONAL, default=No default value.]
Boolean overwrite [INPUT, OPTIONAL, default=False.]
SpectrumContainer ds1 [INPUT, OPTIONAL, default=no default value.]
SpectrumContainer ds2 [INPUT, OPTIONAL, default=no default value.]
Object segments [INPUT, OPTIONAL, default=no default value.]
Object segments1 [INPUT, OPTIONAL, default=no default value.]
Object segments2 [INPUT, OPTIONAL, default=no default value.]
String variant [INPUT, OPTIONAL, default=no default value.]
SpectrumContainer result [OUTPUT, OPTIONAL, default=no default value.]

API details

Properties

SpectrumContainer ds [INPUT, OPTIONAL, default=no default value.]
Input container to be processed by the task in case the task is used as a scalar operation.
Double param [INPUT, OPTIONAL, default=no default value.]
The scalar parameter to be considered when the task is used as scalar operation.
Object selection [INPUT, OPTIONAL, default=None.]
Specify what point spectra the operation should be applied to. Different ways to specify these selections are possible: <ul style="list-style-type: none"> Specify a list of indices (in jython) of the point spectra for which the add should be applied. Use a dictionary (in jython) to specify a selection model (lookup for attributes, filter attributes to be included in ranges or intervals). Pass a string representation of a jython list or a jython dictionary. The string will be parsed and the processing delegated to the two cases above. Pass any java instance that implements the SelectionModel interface (for the advanced user).

<code>PyDictionary Map> selection_lookup [INPUT, OPTIONAL, default=no default value.]</code>
Specify a PyDictionary with keys given by the attribute name to look up and a list of admissible values. Behind the scenes, a DiscreteValueSelectionModel is instantiated.
<code>PyList selection_index [INPUT, OPTIONAL, default=No default value.]</code>
Specify a PyList with the indices of the point spectra to be considered.
<code>Boolean overwrite [INPUT, OPTIONAL, default=False.]</code>
Specify whether the input data container can be reused - the values found therein is overwritten.
<code>SpectrumContainer ds1 [INPUT, OPTIONAL, default=no default value.]</code>
First input container for pair-wise operations.
<code>SpectrumContainer ds2 [INPUT, OPTIONAL, default=no default value.]</code>
Second input container for pair-wise operations.
<code>Object segments [INPUT, OPTIONAL, default=no default value.]</code>
Specify what segments the operation should be applied to. There are two options available: <ul style="list-style-type: none"> • Either pass an instance of a SegmentSelection which gives the information on what segments for each point spectrum included in the container. • Specify a PyList of segment indices.
<code>Object segments1 [INPUT, OPTIONAL, default=no default value.]</code>
Specify the segment selection to be associated with 'ds1'.
<code>Object segments2 [INPUT, OPTIONAL, default=no default value.]</code>
Specify the segment selection to be associated with 'ds2'.
<code>String variant [INPUT, OPTIONAL, default=no default value.]</code>
Specify the variant of processing mode you would like to run (including flags / weights / ...). HCSS defaults include: "flux" / "flux-wave" / "flux-wave-weight" / "flux-flag-wave" / "flux-weight-flag-wave"
<code>SpectrumContainer result [OUTPUT, OPTIONAL, default=no default value]</code>
Result object containing the results of the operation applied.

See also


- [SpectrumTask](#)

History

- 2007-08-17 - meli: initial.

- 2008-03-16 - meli: major refactoring of the toolbox.
- 2008-10-03 - meli: segments, javadoc

2.103. Double1d


Full Name:	herschel.ia.numeric.Double1d
Type:	Java Class - 
Import:	from herschel.ia.numeric import Double1d

Description

A rectangular numeric double array of rank 1.

Numeric arrays are part of the Numeric library. An explanation on how to use them, you can find in chapter **Numeric** -> **Arrays**. (Yes, it should be a link)

2.104. Double2d


Full Name:	herschel.ia.numeric.Double2d
Type:	Java Class - 
Import:	from herschel.ia.numeric import Double2d

Description

A rectangular numeric double array of rank 2.

Numeric arrays are part of the Numeric library. An explanation on how to use them, you can find in chapter **Numeric** -> **Arrays**. (Yes, it should be a link)

2.105. Double3d


Full Name:	herschel.ia.numeric.Double3d
Type:	Java Class - 
Import:	from herschel.ia.numeric import Double3d

Description

A rectangular numeric double array of rank 3.

Numeric arrays are part of the Numeric library. An explanation on how to use them, you can find in chapter **Numeric** -> **Arrays**. (Yes, it should be a link)

2.106. Double4d


Full Name:	herschel.ia.numeric.Double4d
Type:	Java Class - 
Import:	from herschel.ia.numeric import Double4d

Description

A rectangular numeric double array of rank 4.

Numeric arrays are part of the Numeric library. An explanation on how to use them, you can find in chapter **Numeric** -> **Arrays**. (Yes, it should be a link)

2.107. Double5d


Full Name:	herschel.ia.numeric.Double5d
Type:	Java Class - 
Import:	from herschel.ia.numeric import Double5d

Description

A rectangular numeric double array of rank 5.

Numeric arrays are part of the Numeric library. An explanation on how to use them, you can find in chapter **Numeric** -> **Arrays**. (Yes, it should be a link)

2.108. EigenvalueDecomposition

Full Name:	herschel.ia.numeric.toolbox.matrix.EigenvalueDecomposition
Alias:	EigenvalueDecomposition
Type:	Java Class - 
Import:	from herschel.ia.numeric.toolbox.matrix import EigenvalueDecomposition

Example

Example 1: A=Double2d([[1.,1.,1.],[1.,2.,3.],[1.,3.,6.]])
<pre> evd=A.apply(EigenvalueDecomposition()) print evd.d print evd.v </pre>

API Summary

Jython Syntax
<pre> A=Double2d() evd=A.apply(EigenvalueDecomposition()) # get the eigenvalue matrix D D = evd.d # get the eigenvector matrix V V = evd.v # Get the imaginary eigenvalues imagEigenvalues = evd.imagEigenvalues # Get the real eigenvalues realEigenvalues = evd.realEigenvalues </pre>
Property
Double2d A [INPUT, MANDATORY, default=no default value]

API details

Property

Double2d A [INPUT, MANDATORY, default=no default value]
Input must be a Double2d or Float2d array.

2.109. EllipseHistogramExplorer

Full Name:	herschel.ia.gui.image.EllipseHistogramExplorer
Type:	Java Class - 
Import:	from herschel.ia.gui.image import EllipseHistogramExplorer

Description

An explorer for EllipseHistogramProducts.

An explorer for EllipseHistogramProducts.

API Summary

Constructors
EllipseHistogramExplorer() The construction of a new EllipseHistogramExplorer.
EllipseHistogramExplorer(Object object) The construction of a new EllipseHistogramExplorer associated with the given object.
Methods
String getName() Returns the name.
String getDescription() Returns the description.
boolean canHandle(Class className) Checks whether this EllipseHistogramExplorer can handle objects of the given class.
setObject(Object object) Sets the object.
EllipseHistogramProduct getObject() Returns the object.
Class getVariableType() Returns the expected variable type.
JScrollPane getParameterTable() Returns the parameter table.

API details

Constructors

EllipseHistogramExplorer() The construction of a new EllipseHistogramExplorer. The construction of a new EllipseHistogramExplorer.
EllipseHistogramExplorer(Object object) The construction of a new EllipseHistogramExplorer associated with the given object.

EllipseHistogramExplorer([Object](#) object)

The construction of a new EllipseHistogramExplorer associated with the given object.

Argument

[Object](#) object [INPUT, MANDATORY]

Methods

[String](#) getName()

Returns the name.

Returns the name for this EllipseHistogramExplorer.

Return

[String](#)

Returns the name for this EllipseHistogramExplorer.

[String](#) getDescription()

Returns the description.

Returns the description for this EllipseHistogramExplorer.

Return

[String](#)

Returns the description for this EllipseHistogramExplorer.

boolean canHandle([Class](#) className)

Checks whether this EllipseHistogramExplorer can handle objects of the given class.

Returns true if this EllipseHistogramExplorer can handle objects of the given class; false otherwise.

Argument

[Class](#) className [INPUT, MANDATORY]

Return

boolean

Returns true if this EllipseHistogramExplorer can handle objects of the given class; false otherwise.

setObject([Object](#) object)

Sets the object.

Sets the object for this EllipseHistogramExplorer to the given object.

Argument

[Object](#) object [INPUT, MANDATORY]

[EllipseHistogramProduct](#) getObject()

Returns the object.

[EllipseHistogramProduct](#) getObject()

Returns the object for this EllipseHistogramExplorer.

Return

[EllipseHistogramProduct](#)

Returns the object for this EllipseHistogramExplorer.

[Class](#) getVariableType()

Returns the expected variable type.

Returns the expected variable type for this EllipseHistogramExplorer.

Return

[Class](#)

Returns the expected variable type for this EllipseHistogramExplorer.

[JScrollPane](#) getParameterTable()

Returns the parameter table.


Returns the parameter table for this EllipseHistogramExplorer.

Return

[JScrollPane](#)

Returns the parameter table for this EllipseHistogramExplorer.

2.110. EllipseHistogramPanel

Full Name:	herschel.ia.toolbox.image.gui.EllipseHistogramPanel
Type:	Java Class - 
Import:	from herschel.ia.toolbox.image.gui import EllipseHistogramPanel

Description

A panel for the EllipseHistogramTask.

A panel to serve as GUI for the EllipseHistogramTask.

API Summary

Constructor
EllipseHistogramPanel() The construction of a new EllipseHistogramPanel.
Methods
drawFigure() Draws the ellipse on the image.
updateFigure() Updates the associated ellipse.
trigger() Triggers the execution of the associated task.
updateHistogram() Updates the associated histogram.

API details

Constructor

<code>EllipseHistogramPanel()</code>
The construction of a new EllipseHistogramPanel.
The construction of a new EllipseHistogramPanel

Methods

<code>drawFigure()</code>
Draws the ellipse on the image.
Draws the ellipse on the image associated with this EllipseHistogramPanel.
<code>updateFigure()</code>
Updates the associated ellipse.
Updates the ellipse associated with this EllipseHistogramPanel.

trigger()
Triggers the execution of the associated task.
Triggers the execution of the task associated with this EllipseHistogramPanel.

updateHistogram()
Updates the associated histogram.
Updates the histogram associated with this EllipseHistogramPanel.

2.111. EllipseHistogramProduct

Full Name:	herschel.ia.dataset.image.EllipseHistogramProduct
Type:	Java Class - 
Import:	from herschel.ia.dataset.image import EllipseHistogramProduct

Description

A class to deal with the results of an ellipse histogram.

API Summary

Constructor
<p>EllipseHistogramProduct()</p> <p>The constructor of a new EllipseHistogramProduct.</p>
Methods
<p>setCenter(double centerX, double centerY)</p> <p>Sets the center for this EllipseHistogramProduct</p>
<p>setCenter(double centerX, double centerY, String centerRA, String centerDec)</p> <p>Sets the center for this EllipseHistogramProduct</p>
<p>setDimensions(double widthPixels, double heightPixels)</p> <p>Sets the dimensions for this EllipseHistogramProduct to the</p>
<p>setDimensions(double widthPixels, double heightPixels, double widthArcsec, double heightArcsec)</p> <p>Sets the dimensions for this EllipseHistogramProduct to the</p>
<p>DoubleId getCenterPixelCoordinates()</p> <p>Returns the center for this EllipseHistogramProduct in</p>
<p>StringId getCenterSkyCoordinates()</p> <p>Returns the center for this EllipseHistogramProduct in</p>
<p>double getWidthPixels()</p> <p>Returns the width for this EllipseHistogramProduct in pixels.</p>
<p>double getWidthArcsec()</p> <p>Returns the width for this EllipseHistogramProduct in arcsec.</p>
<p>double getHeightPixels()</p> <p>Returns the height for this EllipseHistogramProduct in pixels.</p>
<p>double getHeightArcsec()</p> <p>Returns the height for this EllipseHistogramProduct in arcsec.</p>

API details

Constructor

EllipseHistogramProduct()
The constructor of a new EllipseHistogramProduct.

Methods

setCenter(double centerX, double centerY)

Sets the center for this EllipseHistogramProduct to the given pixel coordinates.

Arguments

double **centerX** [INPUT, MANDATORY]
double **centerY** [INPUT, MANDATORY]

setCenter(double centerX, double centerY, [String](#) centerRA, [String](#) centerDec)

Sets the center for this EllipseHistogramProduct to the given pixel and sky coordinates.

Arguments

double **centerX** [INPUT, MANDATORY]
double **centerY** [INPUT, MANDATORY]
[String](#) **centerRA** [INPUT, MANDATORY]
[String](#) **centerDec** [INPUT, MANDATORY]

setDimensions(double widthPixels, double heightPixels)

Sets the dimensions for this EllipseHistogramProduct to the given dimensions in pixels.

Arguments

double **widthPixels** [INPUT, MANDATORY]
double **heightPixels** [INPUT, MANDATORY]

setDimensions(double widthPixels, double heightPixels, double widthArcsec, double heightArcsec)

Sets the dimensions for this EllipseHistogramProduct to the given dimensions in pixels and arcsec.

Arguments

double **widthPixels** [INPUT, MANDATORY]
double **heightPixels** [INPUT, MANDATORY]
double **widthArcsec** [INPUT, MANDATORY]
double **heightArcsec** [INPUT, MANDATORY]

[DoubleId](#) **getCenterPixelCoordinates()**

Returns the center for this EllipseHistogramProduct in pixel coordinates.

Return

[DoubleId](#)

[Double1d](#) getCenterPixelCoordinates()

Returns the center for this EllipseHistogramProduct in pixel coordinates.

[String1d](#) getCenterSkyCoordinates()

Returns the center for this EllipseHistogramProduct in sky coordinates.

Return

[String1d](#)

Returns the center for this EllipseHistogramProduct in sky coordinates.

double getWidthPixels()

Returns the width for this EllipseHistogramProduct in pixels.

Return

double

Returns the width for this EllipseHistogramProduct in pixels.

double getWidthArcsec()

Returns the width for this EllipseHistogramProduct in arcsec.

Return

double

Returns the width for this EllipseHistogramProduct in arcsec.

double getHeightPixels()

Returns the height for this EllipseHistogramProduct in pixels.

Return

double

Returns the height for this EllipseHistogramProduct in pixels.

double getHeightArcsec()


Returns the height for this EllipseHistogramProduct in arcsec.

Return

double

Returns the height for this EllipseHistogramProduct in arcsec.

2.112. EllipseHistogramTask

Full Name:	herschel.ia.toolbox.image.EllipseHistogramTask
Type:	Java Task - 
Import:	from herschel.ia.toolbox.image import EllipseHistogramTask

Description

A Task to make a histogram within an ellipse.

A Task to make a histogram of a region of interest, which is bounded by an ellipse.

API Summary

Properties
Image image [INPUT, MANDATORY, default=No default value]
Double lowCut [INPUT, OPTIONAL, default=Default value : NaN]
Double highCut [INPUT, OPTIONAL, default=Default value : NaN]
Integer bins [INPUT, MANDATORY, default=Default value : new Integer(2000)]
Double centerX [INPUT, OPTIONAL, default=Default value : NaN]
Double centerY [INPUT, OPTIONAL, default=Default value : NaN]
String centerRA [INPUT, OPTIONAL, default=Default value : "center-RA"]
String centerDec [INPUT, OPTIONAL, default=Default value : "centerDec"]
Double widthPixels [INPUT, OPTIONAL, default=Default value : NaN]
Double heightPixels [INPUT, OPTIONAL, default=Default value : NaN]
Double widthArcsec [INPUT, OPTIONAL, default=Default value : NaN]
Double heightArcsec [INPUT, OPTIONAL, default=Default value : NaN]

API details

Properties

Image image [INPUT, MANDATORY, default=No default value]
The image.
Double lowCut [INPUT, OPTIONAL, default=Default value : NaN]
The lower cut level.
Double highCut [INPUT, OPTIONAL, default=Default value : NaN]
The upper cut level.
Integer bins [INPUT, MANDATORY, default=Default value : new Integer(2000)]
The number of bins.

Double centerX [INPUT, OPTIONAL, default=Default value : NaN]

The x-pixel-coordinate of the center of the ellipse.

Double centerY [INPUT, OPTIONAL, default=Default value : NaN]

The y-pixel-coordinate of the center of the ellipse.

String centerRA [INPUT, OPTIONAL, default=Default value : "centerRA"]

The right ascension of the center of the ellipse.

String centerDec [INPUT, OPTIONAL, default=Default value : "centerDec"]

The declination of the center of the ellipse.

Double widthPixels [INPUT, OPTIONAL, default=Default value : NaN]

The width of the ellipse in pixels.

Double heightPixels [INPUT, OPTIONAL, default=Default value : NaN]

The height of the ellipse in pixels.


Double widthArcsec [INPUT, OPTIONAL, default=Default value : NaN]

The width of the ellipse in arcsec.

Double heightArcsec [INPUT, OPTIONAL, default=Default value : NaN]

The height of the ellipse in arcsec.

2.113. Erfc

Full Name:	herschel.ia.numeric.toolbox.basic.Erfc
Type:	Java Class - 
Import:	from herschel.ia.numeric.toolbox.basic import Erfc

Description

Computes the Complementary Error function.

Example

Example 1: Apply ERFC on a Double1d
<pre>x = Double1d([-5.0,-1.0,-0.5,0.0,0.5,1.0,5.0]) erc = ERFC(x) print erc # [1.999999999984626,1.8427007900291827,1.520499876068384, 1.0,0.4795001239316159,0.15729920997081737,1.5374597939680894E-12]</pre>

API Summary

Jython Syntax
<y>=ERFC(<x>)

Properties
an double array or number x [INPUT, MANDATORY, default=.]
returns an array (or a number if the input is a number) y [OUTPUT, MANDATORY, default=no default value]

API details


Properties

<code>an double array or number x [INPUT, MANDATORY, default=.]</code>
<code>returns an array (or a number if the input is a number) y [OUTPUT, MANDATORY, default=no default value]</code>
where each element is the Complementary Error function of the corresponding element of the input array.

See also

- [Erf](#)

2.114. Erf

Full Name:	herschel.ia.numeric.toolbox.basic.Erf
Type:	Java Class - 
Import:	from herschel.ia.numeric.toolbox.basic import Erf

Description

Computes the Error function.

Example

Example 1: Apply ERF on a Double1d
<pre>x = Double1d([-5.0,-1.0,-0.5,0.0,0.5,1.0,5.0]) er = ERF(x) print er # [-0.9999999999984626,-0.8427007900291826,-0.5204998760683841, 0.0,0.5204998760683841,0.8427007900291826,0.9999999999984626]</pre>

API Summary

Jython Syntax
<y>=ERF(<x>)
Properties
an double array or number x [INPUT, MANDATORY, default=.]
returns an array (or a number if the input is a number) y [OUTPUT, MANDATORY, default=no default value]

API details


Properties

an double array or number x [INPUT, MANDATORY, default=.]
returns an array (or a number if the input is a number) y [OUTPUT, MANDATORY, default=no default value]
where each element is the Error function of the corresponding element of the input array.

See also

- [Erfc](#)

2.115. EXP10

Full Name:	herschel.ia.numeric.toolbox.basic.Exp10
Alias:	EXP10
Type:	Java Class - 
Import:	from herschel.ia.numeric.toolbox.basic import Exp10

Description

Gives 10 raised to the power of array x

Example

Example 1: Apply EXP10 on a Int1d

```
x=Double1d([1,2,3,4])
print EXP10(x) # [10.0, 100.0, 1000.0, 10000.0]
```

API Summary

Jython Syntax

```
<y>=EXP10(<x>)
```

Properties

[any number or array **x** \[INPUT, MANDATORY, default=no default value\]](#)

[a number or an array **y** \[OUTPUT, MANDATORY, default=no default value\]](#)

API details

Properties

any number or array **x [INPUT, MANDATORY, default=no default value]**


a number or an array **y [OUTPUT, MANDATORY, default=no default value]**

The result is a number or an array depending on the input

See also

- [EXP10](#)

2.116. EXP

Full Name:	herschel.ia.numeric.toolbox.basic.Exp
Alias:	EXP
Type:	Java Class - 
Import:	from herschel.ia.numeric.toolbox.basic import Exp

Description

Gives the exponential function applied to a number or array.

Example

Example 1: Apply EXP on a Int1d

```
x=Double1d([0,1])
print LOG(EXP(x)) # [0.0,1.0]
```

API Summary

Jython Syntax

```
<y>=EXP(<x>)
```

Properties

any number or array **x** [INPUT, MANDATORY, default=no default value]

a number or an array **y** [OUTPUT, MANDATORY, default=no default value]

API details

Properties

any number or array **x** [INPUT, MANDATORY, default=no default value]


a number or an array **y** [OUTPUT, MANDATORY, default=no default value]

The result is a number or an array depending on the input

See also

- [LOG](#)

2.117. ExpModel

Full Name:	herschel.ia.numeric.toolbox.fit.ExpModel
Type:	Java Class - 
Import:	from herschel.ia.numeric.toolbox.fit import ExpModel

Description

Exponential Model.

$$f(x;p) = p_0 * \exp(p_1 * x)$$

where p_0 = amplitude, p_1 = slope and As always x = input.

The parameters are initialized at {1.0, -1.0}. It is a non-linear model.

Beware of a positive 2nd parameter. It is going off to Infinity quickly.


See [example](#)

Example

Example 1: ExpModel

```
em = ExpModel()           # exponential
print em.getNumberOfParameters() # 2
```

2.118. ExpN

Full Name:	herschel.ia.numeric.toolbox.basic.ExpN
Type:	Java Class - 
Import:	from herschel.ia.numeric.toolbox.basic import ExpN

Description

Gives the specified base raised to each item array: $y = \text{ExpN base}^{(x)}$.

Example

Example 1: Apply ExpN on a Int1d
<pre>x=Double1d([1,2,3,4]) print ExpN(2)(x) # [2.0,4.0,8.0,16.0] print x.apply(ExpN(2)) # [2.0,4.0,8.0,16.0]</pre>

API Summary

Jython Syntax
<code><y>=ExpN(<base>)(<x>)</code>
Properties
any array or number x [INPUT, MANDATORY, default=no default value]
a number base [INPUT, MANDATORY, default=no default value]
a number or an array y [OUTPUT, MANDATORY, default=no default value]

API details

Properties

any array or number x [INPUT, MANDATORY, default=no default value]
An array or a number
a number base [INPUT, MANDATORY, default=no default value]
The base n
a number or an array y [OUTPUT, MANDATORY, default=no default value]
The result is a number or an array depending on the input

See also

- [ExpN](#)

2.119. exportPalToUfDir

Full Name:	herschel.ia.toolbox.util.ExportPalToUfDirTask
Alias:	exportPalToUfDir
Type:	Java Task - 
Import:	from herschel.ia.toolbox.util import ExportPalToUfDirTask
Category:	task

Description

Exports observations related to an URN from a pool to a user friendly directory structure

The exportPalToUfDir task is used to export observations to a user friendly directory structure, so that they can be easily inspected, used and shared outside of HIPE.

Please note that only LocalStores and HsaReadPools can export their observations.

Example

Example 1: Simple example

```
mypool = PoolManager.getPool("mib")
exportPalToUfDir(pool=mypool -,
urn="urn:poolid:herschel.ia.obs.ObservationContext:0", dirout="/exportdir1")
```

API Summary

Jython Syntax

```
exportPalToUfDir(<pool> , <urn>, <dirout> [, <warn>, compress,
moveTo])
```

Properties

[ProductPool pool](#) [INPUT, MANDATORY, default=No default value]

[String urn](#) [INPUT, MANDATORY, default=No default value]

[String dirout](#) [INPUT, MANDATORY, default=No default value]

[Boolean warn](#) [INPUT, OPTIONAL, default=true]

[String compression](#) [INPUT, OPTIONAL, default="NONE"]

[String moveTo](#) [INPUT, OPTIONAL, default=No default value]

Limitations

moveTo only works inside the same physical disk structure: check your shares and mount points. If in doubt, either move inside the same directory or use an external copy file command (more time).

Miscellaneous

No miscellaneous

API details

Properties

ProductPool pool [INPUT, MANDATORY, default=No default value]

The pool to export products from.

String urn [INPUT, MANDATORY, default=No default value]

The urn that defines what to export

String dirout [INPUT, MANDATORY, default=No default value]

The directory where the observations will be exported with a user friendly structure. Will not be deleted but can be moved via moveTo.

Boolean warn [INPUT, OPTIONAL, default=true]

If true, asks confirmation if the dirout directory already exists and it is not empty.

String compression [INPUT, OPTIONAL, default="NONE"]

The type of compression in the output file. Valid values are ""NONE", "TAR" and "TGZ" The output file(s) will be dirout with .tar or .tgz respectively when no moveTo is present. The intermediate tar will be removed when the "TGZ" is created. They should not exist.

String moveTo [INPUT, OPTIONAL, default=No default value]

The final path of the resulting Ufdir (possibly archived), should not exist, even as a temporal file produced by compression by this task! Note that if you do not compress dirout will be moved to moveTo.


See also

- [???](#)

History

- 2009-01-16 - JDS: initial version
- 2009-09-30 - JSS: modifiers are created through getCustomModifiers (SCR HCSS-8253)

2.120. ExportSpectrumToASCII

Full Name:	herschel.ia.toolbox.spectrum.ExportSpectrumToASCII
Alias:	ExportSpectrumToASCII
Type:	Java Task - 
Import:	from herschel.ia.toolbox.spectrum import ExportSpectrumToASCII

Description

Task for exporting spectrum data to ASCII.

API Summary

Properties
SpectrumContainer ds [INPUT, MANDATORY, default=no default value.]
Object selection [INPUT, OPTIONAL, default=no default value.]
PyArray segments [INPUT, OPTIONAL, default=no default value.]
String file [INPUT, OPTIONAL, default=False.]
Boolean meta [INPUT, OPTIONAL, default=False.]
Boolean flags [INPUT, OPTIONAL, default=False.]
Boolean weights [INPUT, OPTIONAL, default=False.]
Boolean concat [INPUT, OPTIONAL, default=False.]

API details

Properties

SpectrumContainer ds [INPUT, MANDATORY, default=no default value.]
Data to be exported.
Object selection [INPUT, OPTIONAL, default=no default value.]
Specify a selection of the segments / point spectra to be considered.
PyArray segments [INPUT, OPTIONAL, default=no default value.]
Specify the list of segments to be considered.
String file [INPUT, OPTIONAL, default=False.]
The location and name of the file to write the output to.
Boolean meta [INPUT, OPTIONAL, default=False.]
Specifies whether the meta data found in the input data are also included as header information in the output file.
Boolean flags [INPUT, OPTIONAL, default=False.]
Specifies whether the per channel flags also are exported.


Boolean weights [INPUT, OPTIONAL, default=False.]

Specifies whether the per channel weights also are exported.

Boolean concat [INPUT, OPTIONAL, default=False.]

Specifies whether segments are concatenated before exporting them. In general the spectra may be organized in several, possibly disjoint ranges. When setting the concat flag to True it means that all these segments are just glued. This may lead to frequencies not monotonously in- or decreasing or with gaps.

2.121. ExtractFreqRanges

Full Name:	herschel.ia.toolbox.spectrum.ExtractFreqRanges
Type:	Java Task - 
Import:	from herschel.ia.toolbox.spectrum import ExtractFreqRanges

Description

Task for extracting a range or ranges from the segments of one or several SpectrumContainer.

Different possibilities are available for specifying the ranges (see the `ranges`-parameter below. Specifying frequency intervals means that all the spectra are tested for the suitable ranges. In presence of frequency drift these ranges may be different from spectrum to spectrum. For the spectrum container the enveloping range is taken so that the all the PointSpectra included in the result container have again the same width of the segments. Here, the number of frequency intervals and the number of segments does not need to coincide. For each of the segments with a non-trivial overlap with one of the frequency intervals will lead to a segment in the result container.

Example

Example 1: from Jide
<pre>slices = extract(ds=spectra, segments=[1,3]) slices = extract(ds=spectra, ranges=[(4100,4400),(4600,4900),(5100,5900), (6100,6900),(7100,7900)]) slices = extract(ds=spectra, ranges=[(4100,4400),(4600,4900),(5100,5900), (6100,6900),(7100,7900)], selection={"bbtype":[6031]}) slices = extract(ds=spectra, ranges=[(4100,4400),(4600,4900),(5100,5900), (6100,6900),(7100,7900)], selection={"Chopper":([-4.4,5.9],0.2)}) slices = extract(ds=spectra, ranges=[(4100,4400),(4600,4900),(5100,5900), (6100,6900),(7100,7900)], selection=[0,1,2,3]) slices = extract(ds=spectra, ranges=[(4100,4400),(4600,4900),(5100,5900), (6100,6900),(7100,7900)], selection={"LoFrequency":(4000.0,5000.0)}) slices = extract(ds=spectra, ranges=[(4100,4600,5100,6100,7100), [4400,4900,5900,6900,7900]]) # or in -"the old way": min = Double1d([4100,4600,5100,6100,7100]) max = Double1d([4400,4900,5900,6900,7900]) slices = extract(ds=spectra, min=min, max=max)</pre>

API Summary

Properties
Object <code>ds</code> [INPUT, OPTIONAL, default=no default value.]
SpectrumContainer <code>result</code> [OUTPUT, OPTIONAL, default=no default value.]
Object <code>ranges</code> [INPUT, OPTIONAL, default=no default value.]
Double1d <code>minFreq</code> [INPUT, OPTIONAL, default=no default value.]
Double1d <code>maxFreq</code> [INPUT, OPTIONAL, default=no default value.]
Object <code>segments</code> [INPUT, OPTIONAL, default=no default value.]
Object <code>selection</code> [INPUT, OPTIONAL, default=None.]

API details

Properties

Object ds [INPUT, OPTIONAL, default=no default value.]
<p>Input data to be processed by the task. Several types are possible:</p> <ul style="list-style-type: none"> • SpectrumContainer • Array of SpectrumContainer, i.e. SpectrumContainer[] • List of SpectrumContainer, i.e. List • Any product with implementations of SpectrumContainer inside.
SpectrumContainer result [OUTPUT, OPTIONAL, default=no default value.]
The resulting container created by this task.
Object ranges [INPUT, OPTIONAL, default=no default value.]
<p>Specify the range to be extracted - either by specifying ranges of channel numbers or frequency ranges:</p> <ul style="list-style-type: none"> • Type Range[]: Range of channel numbers for each sub-segment to be extracted. • Type Range: Range of channel numbers to be extracted (if only one segment in the data). • Type PyList or PyTuple: Each PyTuple defines an interval on the frequency scale. • Type PyTuple of two PyLists: The first list with the minima and the second with the maxima of the intervals on the frequency scale.
DoubleId minFreq [INPUT, OPTIONAL, default=no default value.]
For each of the ranges to be extracted the minimum frequency.
DoubleId maxFreq [INPUT, OPTIONAL, default=no default value.]
For each of the ranges to be extracted the maximum frequency.
Object segments [INPUT, OPTIONAL, default=no default value.]
<p>Specify what segments to restrict on. There are two options available:</p> <ul style="list-style-type: none"> • Specify a PyList of segment indices or • pass an instance of a SegmentSelection which gives the information on what segments for each point spectrum included in the container.
Object selection [INPUT, OPTIONAL, default=None.]
Specify what point spectra to restricted to. Different ways to specify these selections are possible:

Object selection [INPUT, OPTIONAL, default=None.]

- Specify a list of indices (in jython) of the point spectra for which the processing should be applied.
- Use a dictionary (in jython) to specify a selection model (lookup for attributes, filter attributes to be included in ranges or intervals).
- Pass a string representation of a jython list or a jython dictionary. The string will be parsed and the processing delegated to the two cases above.
- Pass any java instance that implements the SelectionModel interface (for the advanced user).


See also

- [ManyToOneSpectrumTask](#)

History

- 2008-01-29 - meli: initial.
- 2008-03-16 - meli: major refactoring of the toolbox.
- 2009-05-18 - meli: more ways to specify selections and frequency ranges.

2.122. ExtractMultiplePixelSpectrumPanel

Full Name:	herschel.ia.gui.cube.ExtractMultiplePixelSpectrumPanel
Type:	Java Class - 
Import:	from herschel.ia.gui.cube import ExtractMultiplePixelSpectrumPanel

Description

A panel for the SinglePixelSpectrum extraction task

API Summary

Constructor
ExtractMultiplePixelSpectrumPanel() The constructor of a new ExtractSinglePixelSpectrumPanel.
Methods
JPanel getPixelsPanel() Returns the panel where to specify the radii for the
drawFigures() Draws the circles on the image for this
clearSpectrumExtraction() Clears the annular sky aperture for this
setClicked(Double center) Sets the center of the rectangle, associated with this

API details

Constructor

ExtractMultiplePixelSpectrumPanel()
The constructor of a new ExtractSinglePixelSpectrumPanel.

Methods

JPanel getPixelsPanel() Returns the panel where to specify the radii for the apertures for this ExtractSinglePixelSpectrumPanel. Return JPanel Returns the panel where to specify the radii for the apertures for this AnnularSkyAerturePhotometryPanel.
drawFigures() Draws the circles on the image for this

drawFigures()

ExtractSinglePixelSpectrumPanel.

clearSpectrumExtraction()

Clears the annular sky aperture for this

ExtractSinglePixelSpectrumPanel.

setClicked([Double](#) center)


Sets the center of the rectangle, associated with this

ExtractSinglePixelSpectrumPanel to the given point (in UserCoordinates).

Argument

[Double](#) center [INPUT, MANDATORY]

2.123. ExtractRegionPixelSpectrumTask

Full Name:	herschel.ia.toolbox.cube.ExtractRegionPixelSpectrumTask
Type:	Java Task - 
Import:	from herschel.ia.toolbox.cube import ExtractRegionPixelSpectrumTask

Description

Task which Extract an averaged Spectrum from a set of pixels, from a simplecube.

In the CubeSpectrumAnalysisToolbox (via the SpectrumAvgPlotting GUI), this task process a set of contiguous pixels and can be used in association with some FILTERRING, the output of this task in this kind of usage is therefore sent directly to the next tasks.

The command line can be used to retrieve manually some spectrum or to include it in scripts. this task use SimpleCube and return

Example

Example 1: How to use the Single pixel spectrum extraction in jide

```
regionExtraction = ExtractRegionPixelSpectrumTask
regionExtraction.setValue("simplecube",Cube)
regionExtraction.setValue("wholeImg",False)
regionExtraction.setValue("posArray",arrayofpixel)
regionExtraction.execute()
spectrum =(DoubleId) regionExtraction.getValue("spectrum")
SpectrumId spectrumId =(DoubleId) regionExtraction.getValue("spectrum")
{@link #__abs__()} futur
```

API Summary

Properties
SimpleCube simplecube [INPUT, MANDATORY, default=no default value]
Boolean wholeImg [INPUT, no default value, default=no default value]
Double2d posArray [INPUT, MANDATORY, default=no default value]
Integer nbPixels [INPUT, MANDATORY, default=no default value]
Integer arithmetics [INPUT, MANDATORY, default=no default value]
DoubleId spectrum [OUTPUT, MANDATORY, default=no default value]
SpectrumId finalspectrum [OUTPUT, MANDATORY, default=no default value]
float totalWeight [OUTPUT, MANDATORY, default=no default value]

API details

Properties

SimpleCube simplecube [INPUT, MANDATORY, default=no default value]
The Cube containing the spectra to extract

Boolean wholeImg [INPUT, no default value, default=no default value]

A flag to define the usage whole image or region of this task

Double2d posArray [INPUT, MANDATORY, default=no default value]

The Array of pixels to be read in the cube must contain the X,Y and weight information [X,Y,weight]

Integer nbPixels [INPUT, MANDATORY, default=no default value]
--

The number of pixel to be read

Integer arithmetics [INPUT, MANDATORY, default=no default value]

a value defining the kind of arithmetic to use for the resulting spectra: choice between SUM(1) AVG(2) MEDIAN(3) default = avg
--

Double1d spectrum [OUTPUT, MANDATORY, default=no default value]
--

The flux of the spectrum extracted at the given position
--

Spectrum1d finalspectrum [OUTPUT, MANDATORY, default=no default value]

The spectrum extracted at the given position in a Spectrum1d
--

float totalWeight [OUTPUT, MANDATORY, default=no default value]
--

the 'weight' of the spectrum , ne exact number of pixel from which the spectrum was read
--


See also

- [???](#)
- [???](#)
- herschel.ia.dataset.spectrum.Spectrum1d:

History

- 2008-06-17 - AG: first complet description
- 2008-07-03 - AG: Modification of the input Parameters
- 2008-09-08 - AG: add the error computation
- 2009-06-08

2.124. ExtractSinglePixelSpectrumPanel

Full Name:	herschel.ia.gui.cube.ExtractSinglePixelSpectrumPanel
Type:	Java Class - 
Import:	from herschel.ia.gui.cube import ExtractSinglePixelSpectrumPanel

Description

A panel for the SinglePixelSpectrum extraction task.

API Summary

Constructor
ExtractSinglePixelSpectrumPanel() The constructor of a new ExtractSinglePixelSpectrumPanel.
Methods
JPanel getPixelsPanel() Returns the panel where to specify the X,Y pixel
drawFigures() Draws the circles on the image for this
clearSpectrumExtraction() Clears tthe spectrum extraction interface for this
setClicked(Double center) Sets the center of the rectacle, associated with this

API details

Constructor

<code>ExtractSinglePixelSpectrumPanel()</code>
The constructor of a new ExtractSinglePixelSpectrumPanel.

Methods

JPanel getPixelsPanel() Returns the panel where to specify the X,Y pixel position for this ExtractSinglePixelSpectrumPanel. Return JPanel Returns the panel where to specify the X,Y pixel position for this ExtractSinglePixel
drawFigures() Draws the circles on the image for this

drawFigures()

ExtractSinglePixelSpectrumPanel.

clearSpectrumExtraction()

Clears tthe spectrum extraction interface for this

ExtractSinglePixelSpectrumPanel.

setClicked([Double](#) center)


Sets the center of the rectancele, associated with this

ExtractSinglePixelSpectrumPanel to the given point (in UserCoordinates).

Argument

[Double](#) center [INPUT, MANDATORY]

2.125. ExtractSinglePixelSpectrumTask

Full Name:	herschel.ia.toolbox.cube.ExtractSinglePixelSpectrumTask
Type:	Java Task - 
Import:	from herschel.ia.toolbox.cube import ExtractSinglePixelSpectrumTask

Description

Task which extract a Spectrum of a Simplecube from a Single pixel position given as parameter.

This task is used in the GUI CubeSpectrumAnalysisToolbox for the Real time single spectrum visualization and in the the Single pixel Spectrum Extraction.

*

This task can be used in common with some filtering and fitting tasks. in this case the input parameters for these tasks are partly the result of this one. In the main GUI or in the JIDE the user will decide and define of the filter to apply.

Example

Example 1: How to use the Single pixel spectrum extraction

```
singlePixExtraction = ExtractSinglePixelSpectrumTask();
singlePixExtraction.setValue()
```

API Summary

Properties
SimpleCube simplecube [INPUT, MANDATORY, default=no default value]
Integer posX [INPUT, MANDATORY, default=no default value]
Integer posY [INPUT, MANDATORY, default=no default value]
SpectrumId finalspectrum [OUTPUT, MANDATORY, default=no default value]
DoubleId spectrum [OUTPUT, MANDATORY, default=no default value]

API details

Properties

SimpleCube simplecube [INPUT, MANDATORY, default=no default value]
The Cube containing the spectra
Integer posX [INPUT, MANDATORY, default=no default value]
The X coordinate of the pixel to be read.
Integer posY [INPUT, MANDATORY, default=no default value]
The Y coordinate of the pixel to be read.

<code>SpectrumId finalspectrum [OUTPUT, MANDATORY, default=no default value]</code>

The spectrum extracted at the given position in a <code>SpectrumId</code> , default output parameter
--


<code>DoubleId spectrum [OUTPUT, MANDATORY, default=no default value]</code>
--

The spectrum extracted at the given position
--

History

- 2008-06-17 - AG: first complete description
- 2008-07-03 - AG: Modification of the input Parameters
- 2008-07-24 - AG: added a new output parameters
- 2008-10-3 - AG: temporary removed the unit management
- 2009-01-31 - AG: inverted the output parameters order

2.126. FFT2dTask

Full Name:	herschel.ia.toolbox.image.FFT2dTask
Type:	Java Task - 
Import:	from herschel.ia.toolbox.image import FFT2dTask

Description

A Task for two dimensional Fast Fourier Transforms.

A Task that calculates the 2D Fast Fourier Transform and the power spectrum.

API Summary


Properties
Numeric2dData image [INPUT, MANDATORY, default=No default value]
Complex2d transform [OUTPUT, MANDATORY, default=No default value]
Double2d spectrum [OUTPUT, MANDATORY, default=No default value]

API details

Properties

Numeric2dData image [INPUT, MANDATORY, default=No default value]
The input image.
Complex2d transform [OUTPUT, MANDATORY, default=No default value]
The transform.
Double2d spectrum [OUTPUT, MANDATORY, default=No default value]
The spectrum.

2.127. FFT

Full Name:	herschel.ia.numeric.toolbox.xform.FFT
Alias:	FFT
Type:	Java Class - 
Import:	from herschel.ia.numeric.toolbox.xform import FFT

Description

Gives the Fast Fourier Transform `<![CDATA[]]>`.

Example

Example 1: Apply FFT
<pre> from java.lang.Math import PI # Frequency modulated signal: parameters ts = 1E-6 # Sampling period (sec) fc = 200000 # Carrier frequency (Hz) fm = 2000 # Modulation frequency (Hz) beta = -.0003 # Modulation index (Hz) n = 5000 # Number of samples # Create signal in complex form t = Double1d.range(n) * ts signal = SIN(2 * PI * fc * t * (1 + beta * COS(2 * PI * fm * t))) z_signal=Complex1d(signal) # Get spectrum spectrum = ABS(FFT(z_signal)) # Repeat with apodizing z_signal=Complex1d(HAMMING(signal)) spectrum2 = ABS(FFT(z_signal)) </pre>

API Summary

Jython Syntax
<code><y>=FFT(<x>)</code>
Properties
Complex1d x [INPUT, MANDATORY, default=no default value]
Double1d y [INPUT, MANDATORY, default=no default value]

API details


Properties

Complex1d x [INPUT, MANDATORY, default=no default value]
Complex1d arrays only.
Double1d y [INPUT, MANDATORY, default=no default value]
Returns a Double1d

See also

- [IFFT](#)

2.128. FitFringeData

Full Name:	herschel.ia.toolbox.spectrum.standingwaves.FitFringeData
Type:	Java Class - 
Import:	from herschel.ia.toolbox.spectrum.standingwaves import FitFringeData

Description

Class that wraps Double1d's into a SpectralSegment.

It can be used to generate a data object used with the FitFringe and SmoothBaselie in the standingwaves package.

Example

Example 1: Description

```
from herschel.ia.toolbox.spectrum.standingwaves import FitFringeData
from herschel.ia.toolbox.spectrum.standingwaves import FitFringe
myFreq = Double1d(freq_column.data)
myFlux = Double1d(flux_column.data)
myFlag = Int1d(flag_column.data)
myWeight = Double1d(weight_column.data)
ffData1 = FitFringeData(myFreq,myFlux)
ffData2 = FitFringeData(myFreq,myFlux,myFlag,myWeight)
ffData3 = FitFringeData(spectrumContainer,6,2) # WbsSpectrumDataSet, 6=point
spectrum index, 2=segment index
```

API Summary

Properties
Double1d freq [INPUT, MANDATORY, default=no default value]
Double1d flux [INPUT, MANDATORY, default=no default value]
Int1d flag [INPUT, OPTIONAL, default=no default value]
Double1d weight [INPUT, OPTIONAL, default=no default value]
int scindex [INPUT, NOT_MANDATORY, default=point spectrum index]
int segmentindex [INPUT, NOT_MANDATORY, default=segment index]
Boolean convertGHzToMicron [INPUT, NOT_MANDATORY, default=convert GHz or MHz to micron]

API details

Properties

Double1d freq [INPUT, MANDATORY, default=no default value]
- 1d array of wavelength.
Double1d flux [INPUT, MANDATORY, default=no default value]
- 1d array of flux.
Int1d flag [INPUT, OPTIONAL, default=no default value]
- 1d array of flag.

Double1d weight [INPUT, OPTIONAL, default=no default value]


- 1d array of weight.

int scindex [INPUT, NOT_MANDATORY, default=point spectrum index]

int segmentindex [INPUT, NOT_MANDATORY, default=segment index]

Boolean convertGHzToMicron [INPUT, NOT_MANDATORY, default=convert GHz or MHz to micron]

2.129. FitFringe

Full Name:	herschel.ia.toolbox.spectrum.standingwaves.FitFringe
Type:	Java Task - 
Import:	from herschel.ia.toolbox.spectrum.standingwaves import FitFringe

Description

Class that fits and removes fringes from spectra.

Example

Example 1: Description
<pre> from herschel.ia.toolbox.spectrum.standingwaves import FitFringeData from herschel.ia.toolbox.spectrum.standingwaves import FitFringe myFreq = Double1d(freq_column.data) myFlux = Double1d(flux_column.data) myFlag = Int1d(flag_column.data) myWeight = Double1d(weight_column.data) swData = FitFringeData(myFreq,myFlux,myFlag,myWeight) ff = FitFringe() improvedData = ff(ffData,nfringes=2) baseline = ff.baseline fringelist = ff.fringelist </pre>

API Summary

Jython Syntax
<pre> ff = FitFringe() improvedData = ff(myData,nfringes=2) </pre>
Properties
FitFringeData data [INPUT, MANDATORY, default=no default value]
Integer nfringes [INPUT, MANDATORY, default=no default value]
Boolean mhz [INPUT, MANDATORY, default=no default value]
Double cycle [INPUT, MANDATORY, default=no default value]
Double cstep [INPUT, MANDATORY, default=no default value]
Integer ncycle [INPUT, MANDATORY, default=no default value]
Double midcycle [INPUT, MANDATORY, default=no default value]
Double tolerance [INPUT, MANDATORY, default=no default value]
Boolean weight [INPUT, MANDATORY, default=no default value]
Boolean auto [INPUT, MANDATORY, default=no default value]
Integer plot [INPUT, MANDATORY, default=no default value]
Boolean expert [INPUT, MANDATORY, default=no default value]
Object usermask [INPUT, MANDATORY, default=no default value]
Double1d fixfreq [INPUT, MANDATORY, default=no default value]
FitFringeData improvedData [OUTPUT, MANDATORY, default=no default value]

Properties
FitFringeData baseline [OUTPUT, MANDATORY, default=no default value]
DoubleId mask [OUTPUT, MANDATORY, default=no default value]
TableDataset fringelist [OUTPUT, MANDATORY, default=no default value]

API details

Properties

FitFringeData data [INPUT, MANDATORY, default=no default value]
Input FitFringeData which contains: 1d array of wavelength, 1d array of flux, 1d array of flag (optional), 1d array of weight (optional).
Integer nfringes [INPUT, MANDATORY, default=no default value]
- Number of fringes to be removed.
Boolean mhz [INPUT, MANDATORY, default=no default value]
- Unit in mhz.
Double cycle [INPUT, MANDATORY, default=no default value]
- Start of cycle to search for fringes.
Double cystep [INPUT, MANDATORY, default=no default value]
- Step in the cycles.
Integer ncycle [INPUT, MANDATORY, default=no default value]
- Number of cycles to check.
Double midcycle [INPUT, MANDATORY, default=no default value]
- Typical cycle frequency used for smoothing.
Double tolerance [INPUT, MANDATORY, default=no default value]
- Reduce the chisq until reduction is less than tol.
Boolean weight [INPUT, MANDATORY, default=no default value]
- Set all weights to 1.
Boolean auto [INPUT, MANDATORY, default=no default value]
- Automatic determination of the number of fringes to extract.
Integer plot [INPUT, MANDATORY, default=no default value]
- Plot the result.
Boolean expert [INPUT, MANDATORY, default=no default value]
- Expert plot option.

Object usermask [INPUT, MANDATORY, default=no default value]

- Input frequencies of a set of wavelengths to disregard during fringe fitting (eg. at emission lines).

DoubleId fixfreq [INPUT, MANDATORY, default=no default value]

- Fix periods to these values.

FitFringeData improvedData [OUTPUT, MANDATORY, default=no default value]

- Fringe removed data.

FitFringeData baseline [OUTPUT, MANDATORY, default=no default value]

- Smoothed background.


DoubleId mask [OUTPUT, MANDATORY, default=no default value]

- Output mask.

TableDataset fringelist [OUTPUT, MANDATORY, default=no default value]

- Table dataset which contains a cumulative list of extracted fringes. fringeNum - fringe number. cycle - fringe cycle frequency. cycle_in_MHz - fringe cycle in MHz. sinAmp - amplitude of sine component. cosAmp - amplitude of cosine component. chisq - chisq. chiRed - total chisq reduction.

2.130. FitsArchive

Full Name:	herschel.ia.io.fits.FitsArchive
Alias:	FitsArchive
Type:	Java Class - 
Import:	from herschel.ia.io.fits import FitsArchive
Category:	class

Description

A class for reading and writing FITS files.

The FitsArchive provides a transparent way to store and retrieve Products as defined within the Herschel Data Processing software. FitsArchive is thread tolerant. It is not thread safe: do not share a FitsArchive object between several threads.

Examples

Example 1: default usage:

```
from herschel.ia.io.fits import FitsArchive
# read/write a Product in HCSS decorated format.
fits=FitsArchive()
product=fits.load("input.fits")
fits.save("output.fits",product)
```

Example 2: reading a HCSS FITS file which contains a removed class:

```
fits=FitsArchive()
product=fits.load("input.fits",FitsArchive.HANDLE_MISSING_CLASSES)
```

Example 3: reading a externally generated FITS file into a Product:

```
from herschel.ia.io.fits import FitsArchive
# setup a new FitsArchive, but change the reader type
from herschel.ia.io.fits.reader.standard import StandardFitsReader
fits = FitsArchive(reader = StandardFitsReader())
product=fits.load("input.fits")
# Saving the product will be done in HCSS decorated format.
fits.save("output.fits",product)
```

Example 4: saving an empty ArrayDataset is allowed:

```
from herschel.ia.io.fits import FitsArchive
fits=FitsArchive()
ads=ArrayDataset()
product=Product("with empty ArrayDataset")
product['ads'] = ads
fits.save("product.fits", product) # works: image extension without image
```

Example 5: saving an empty column in a TableDataset is NOT allowed:

```
from herschel.ia.io.fits import FitsArchive
fits=FitsArchive()
tds=TableDataset()
tds.addColumn(Column())
product=Product("with empty Column in TableDataset")
product['tds'] = tds
fits.save("product.fits", product) # fails: binary table extensions require data
```

Example 6: saving the product minimizing product metadata keyword translations and making keywords duplicity.:

```

from herschel.ia.io.fits import FitsArchive
fits=FitsArchive()
product=Product("My Product")
fits.save("product.fits", product, 1, 1)

```

API Summary

Fields
FitsReader_HCSS_READER HCSS FITS Reader
FitsReader_STANDARD_READER Generic FITS Reader
Methods
Product load(String name) Loads a Product from a FITS file.
Product load(String name, boolean handleMissingClasses) Loads a Product from a FITS file.
Product load(String file) Loads a Product from a FITS file.
Product load(String file, boolean handleMissingClasses) Loads a Product from a FITS file.
Product load(InputStream is) Loads a Product from a FITS InputStream.
Product load(InputStream is, boolean handleMissingClasses) Loads a Product from a FITS InputStream.
save(String name, [Optionally derived] Product. product) Saves a Product to a FITS file.
save(String name, [Optionally derived] Product. product, Boolean.minimizeTranslations, [Optionally derived] Product. keysDuplicity) Saves a Product to a FITS file.
save(File file, [Optionally derived] Product. product) Saves a Product to a FITS file.
save(File file, [Optionally derived] Product. product, Boolean.minimizeTranslations, [Optionally derived] Product. keysDuplicity) Saves a Product to a FITS file.
save(OutputStream os, [Optionally derived] Product. product) Saves a Product to an OutputStream (creating a FITS file).
save(OutputStream os, [Optionally derived] Product. product, Boolean.minimizeTranslations, [Optionally derived] Product. keysDuplicity) Saves a Product to an OutputStream (creating a FITS file).

Methods
MetaData <code>loadMeta(String name)</code> Loads a MetaData product object from a FITS file.
MetaData <code>loadMeta(File file)</code> Loads a MetaData product object from a FITS file.
MetaData <code>loadMeta(InputStream is)</code> Loads a MetaData product object from a FITS file.
MetaData <code>loadMeta(String name, integer hduIndex)</code> Loads a MetaData object from a FITS file at the specified HDU index.
MetaData <code>loadMeta(String name, integer hduIndex, integer commentExtraSpacesToRemove)</code> Loads a MetaData object from a FITS file at the specified HDU index.
MetaData <code>loadMeta(File file, integer hduIndex)</code> Loads a MetaData object from a FITS file at the specified HDU index.
MetaData <code>loadMeta(File file, integer hduIndex, integer commentExtraSpacesToRemove)</code> Loads a MetaData object from a FITS file at the specified HDU index.
MetaData <code>loadMeta(InputStream is, integer hduIndex)</code> Loads a MetaData object from a FITS file at the specified HDU index.
MetaData <code>loadMeta(InputStream is, integer hduIndex, integer commentExtraSpacesToRemove)</code> Loads a MetaData object from a FITS file at the specified HDU index.
MetaData <code>loadMeta(String name, integer hduIndex, boolean handleMissingClasses)</code> Loads a MetaData object from a FITS file at the specified HDU index.
MetaData <code>loadMeta(String name, integer hduIndex, integer commentExtraSpacesToRemove, boolean handleMissingClasses)</code> Loads a MetaData object from a FITS file at the specified HDU index.
MetaData <code>loadMeta(File file, integer hduIndex, boolean handleMissingClasses)</code> Loads a MetaData object from a FITS file at the specified HDU index.
MetaData <code>loadMeta(File file, integer hduIndex, integer commentExtraSpacesToRemove, boolean handleMissingClasses)</code> Loads a MetaData object from a FITS file at the specified HDU index.
MetaData <code>loadMeta(InputStream is, integer hduIndex, boolean handleMissingClasses)</code> Loads a MetaData object from a FITS file at the specified HDU index.
MetaData <code>loadMeta(InputStream is, integer hduIndex, integer commentExtraSpacesToRemove, boolean handleMissingClasses)</code> Loads a MetaData object from a FITS file at the specified HDU index.

API details

Fields

FitsReader HCSS_READER
<p>HCSS FITS Reader</p> <p>A FITS reader that expects a FITS format that was produced by this archive implementation. This is a shared object among any FitsArchive instance. In a multithread environment, you should use a new instance of a reader.</p> <p>It can handle nested structures, derived datasets and quantities of the data within FITS.</p> <p>This is the default reader when you create a FitsArchive.</p> <p>Examples</p> <p>creating a new FitsArchive</p> <pre>fits=FitsArchive() # default reader=FitsArchive.HCSS_READER</pre> <p>creating a new FitsArchive</p> <pre>fits=FitsArchive(reader=FitsArchive.HCSS_READER)</pre>

FitsReader STANDARD_READER
<p>Generic FITS Reader</p> <p>A generic FITS reader for FITS files that are not created by the HCSS FitsArchive. This is a shared object among any FitsArchive instance. In a multithread environment, you should use a new instance of a reader.</p> <p>This reader can read FITS files that were generated by other software as long as it complies to the FITS standard and translates the contents into a Product.</p> <p>Examples</p> <p>import data from an externally generated FITS file:</p> <pre>fits=FitsArchive(reader=FitsArchive.STANDARD_READER) product=fits.load("external.fits")</pre> <p>reusing this FitsArchive but changing the reader:</p> <pre>fits.reader=fits.HCSS_READER product=fits.load("hcss.fits")</pre>

Methods

Product load (String name)
<p>Loads a Product from a FITS file.</p> <p>Loads a Product from a FITS file using the current reader.</p> <p>Argument</p> <p>String name [INPUT, MANDATORY, default=no default value] Name of the FITS file</p>

Product load(String name)
<p>Return</p> <p>Product</p> <p>An object of the herschel.ia.dataset.Product family.</p>
Product load(String name, boolean handleMissingClasses)
<p>Loads a Product from a FITS file.</p> <p>Loads a Product from a FITS file using the current reader.</p> <p>Arguments</p> <p>String name [INPUT, MANDATORY, default=no default value] Name of the FITS file</p> <p>boolean handleMissingClasses [INPUT, MANDATORY, default=no default value] For creating dummy classes when a Hcss class is not found.</p> <p>Return</p> <p>Product</p> <p>An object of the herschel.ia.dataset.Product family.</p>
Product load(String file)
<p>Loads a Product from a FITS file.</p> <p>Loads a Product from a FITS file using the current reader.</p> <p>Argument</p> <p>String file [INPUT, MANDATORY, default=no default value] FITS file</p> <p>Return</p> <p>Product</p> <p>An object of the herschel.ia.dataset.Product family.</p>
Product load(String file, boolean handleMissingClasses)
<p>Loads a Product from a FITS file.</p> <p>Loads a Product from a FITS file using the current reader.</p> <p>Arguments</p> <p>String file [INPUT, MANDATORY, default=no default value] FITS file</p> <p>boolean handleMissingClasses [INPUT, MANDATORY, default=no default value] For creating dummy classes when a Hcss class is not found.</p> <p>Return</p> <p>Product</p> <p>An object of the herschel.ia.dataset.Product family.</p>

Product load(InputStream is)
Loads a Product from a FITS InputStream.
Loads a Product from a FITS InputStream using the current reader.
Argument
InputStream is [INPUT, MANDATORY, default=no default value] InputStream to the FITS file
Return
Product
An object of the herschel.ia.dataset.Product family.

Product load(InputStream is, boolean handleMissingClasses)
Loads a Product from a FITS InputStream.
Loads a Product from a FITS InputStream using the current reader.
Arguments
InputStream is [INPUT, MANDATORY, default=no default value] InputStream to the FITS file
boolean handleMissingClasses [INPUT, MANDATORY, default=no default value] For creating dummy classes when a Hcss class is not found.
Return
Product
An object of the herschel.ia.dataset.Product family.

save(String name, [Optionally derived] Product. product)
Saves a Product to a FITS file.
Saves a Product to a FITS file with sufficient information to preserve the quantities of the data as well as the original dataset type and contents.
Arguments
String name [INPUT, MANDATORY, default=no default value] Name of the FITS file
[Optionally derived] Product. product [INPUT, MANDATORY, default=no default value] An object of the herschel.ia.dataset.Product family.

save(String name, [Optionally derived] Product. product, Boolean. minimizeTranslations, [Optionally derived] Product. keysDuplicity)
Saves a Product to a FITS file.
Saves a Product to a FITS file with sufficient information to preserve the quantities of the data as well as the original dataset type and contents.
Arguments
String name [INPUT, MANDATORY, default=no default value]

save([String](#) name, [Optionally derived] Product. product, Boolean. minimizeTranslations, [Optionally derived] Product. keysDuplicity)

Name of the FITS file

[Optionally derived] Product. **product** [INPUT, MANDATORY, default=no default value]

An object of the herschel.ia.dataset.Product family.

Boolean. **minimizeTranslations** [INPUT, MANDATORY, default=no default value]

Specifies if some keywords that are not in any dictionary will be translated without using 'META' following some rules.

[Optionally derived] Product. **keysDuplicity** [INPUT, MANDATORY, default=no default value]

Specifies if a keyword will be translated to several FITS keywords.

save([File](#) file, [Optionally derived] Product. product)

Saves a Product to a FITS file.

Saves a Product to a FITS file with sufficient information to preserve the quantities of the data as well as the original dataset type and contents.

Arguments

[File](#) **file** [INPUT, MANDATORY, default=no default value]

FITS file

[Optionally derived] Product. **product** [INPUT, MANDATORY, default=no default value]

An object of the herschel.ia.dataset.Product family.

save([File](#) file, [Optionally derived] Product. product, Boolean. minimizeTranslations, [Optionally derived] Product. keysDuplicity)

Saves a Product to a FITS file.

Saves a Product to a FITS file with sufficient information to preserve the quantities of the data as well as the original dataset type and contents.

Arguments

[File](#) **file** [INPUT, MANDATORY, default=no default value]

FITS file.

[Optionally derived] Product. **product** [INPUT, MANDATORY, default=no default value]

An object of the herschel.ia.dataset.Product family.

Boolean. **minimizeTranslations** [INPUT, MANDATORY, default=no default value]

Specifies if some keywords that are not in any dictionary will be translated without using 'META' following some rules.

[Optionally derived] Product. **keysDuplicity** [INPUT, MANDATORY, default=no default value]

Specifies if a keyword will be translated to several FITS keywords.

save(OutputStream os, [Optionally derived] Product. product)

Saves a Product to an OutputStream (creating a FITS file).

save(OutputStream os, [Optionally derived] Product. product)

Saves a Product to an OutputStream, creating a FITS file with sufficient information to preserve the quantities of the data as well as the original dataset type and contents. The OutputStream is closed by this method.

Arguments

OutputStream **os** [INPUT, MANDATORY, default=no default value]

OutputStream.

[Optionally derived] Product. **product** [INPUT, MANDATORY, default=no default value]

An object of the herschel.ia.dataset.Product family.

save(OutputStream os, [Optionally derived] Product. product, Boolean. minimizeTranslations, [Optionally derived] Product. keysDuplicity)

Saves a Product to an OutputStream (creating a FITS file).

Saves a Product to an OutputStream, creating a FITS file with sufficient information to preserve the quantities of the data as well as the original dataset type and contents. The OutputStream is closed by this method.

Arguments

OutputStream **os** [INPUT, MANDATORY, default=no default value]

OutputStream.

[Optionally derived] Product. **product** [INPUT, MANDATORY, default=no default value]

An object of the herschel.ia.dataset.Product family.

Boolean. **minimizeTranslations** [INPUT, MANDATORY, default=no default value]

Specifies if some keywords that are not in any dictionary will be translated without using 'META' following some rules.

[Optionally derived] Product. **keysDuplicity** [INPUT, MANDATORY, default=no default value]

Specifies if a keyword will be translated to several FITS keywords.

MetaData loadMeta(String name)

Loads a MetaData product object from a FITS file.

Loads a MetaData product object using the current reader.

Argument

[String](#) **name** [INPUT, MANDATORY, default=no default value]

Name of the FITS file

Return

MetaData

A MetaData product object.

MetaData loadMeta(File file)

Loads a MetaData product object from a FITS file.

MetaData loadMeta([File](#) file)

Loads a MetaData product object using the current reader.

Argument

[File](#) **file** [INPUT, MANDATORY, default=no default value]
Name of the FITS file

Return**MetaData**

A MetaData product object.

MetaData loadMeta([InputStream](#) is)

Loads a MetaData product object from a FITS file.

Loads a MetaData product object using the current reader. The user must close the InputStream.

Argument

InputStream **is** [INPUT, MANDATORY, default=no default value]
InputStream to the FITS file.

Return**MetaData**

A MetaData product object.

MetaData loadMeta([String](#) name, integer hduIndex)

Loads a MetaData object from a FITS file at the specified HDU index.

Loads a MetaData object using the current reader for HDU index 0. Loads a MetaData object using STANDARD_READER for HDU index greater than 0.

Arguments

[String](#) **name** [INPUT, MANDATORY, default=no default value]
Name of the FITS file
integer **hduIndex** [INPUT, MANDATORY, default=no default value]
HDU index.

Return**MetaData**

A MetaData object at the specified HDU index.

MetaData loadMeta([String](#) name, integer hduIndex, integer commentExtraSpacesToRemove)

Loads a MetaData object from a FITS file at the specified HDU index.

Loads a MetaData object using the current reader for HDU index 0. Loads a MetaData object using STANDARD_READER for HDU index greater than 0.

Arguments

[String](#) **name** [INPUT, MANDATORY, default=no default value]
Name of the FITS file
integer **hduIndex** [INPUT, MANDATORY, default=no default value]

MetaData loadMeta([String](#) name, integer hduIndex, integer commentExtraSpacesToRemove)

HDU index.

integer **commentExtraSpacesToRemove** [INPUT, MANDATORY, default=no default value]

Comment extra spaces to be removed.

Return

MetaData

A MetaData object at the specified HDU index.

MetaData loadMeta([File](#) file, integer hduIndex)

Loads a MetaData object from a FITS file at the specified HDU index.

Loads a MetaData object using the current reader for HDU index 0. Loads a MetaData object using STANDARD_READER for HDU index greater than 0.

Arguments

[File](#) **file** [INPUT, MANDATORY, default=no default value]

Name of the FITS file

integer **hduIndex** [INPUT, MANDATORY, default=no default value]

HDU index.

Return

MetaData

A MetaData object at the specified HDU index.

MetaData loadMeta([File](#) file, integer hduIndex, integer commentExtraSpacesToRemove)

Loads a MetaData object from a FITS file at the specified HDU index.

Loads a MetaData object using the current reader for HDU index 0. Loads a MetaData object using STANDARD_READER for HDU index greater than 0.

Arguments

[File](#) **file** [INPUT, MANDATORY, default=no default value]

Name of the FITS file

integer **hduIndex** [INPUT, MANDATORY, default=no default value]

HDU index.

integer **commentExtraSpacesToRemove** [INPUT, MANDATORY, default=no default value]

Comment extra spaces to be removed.

Return

MetaData

A MetaData object at the specified HDU index.

MetaData loadMeta(InputStream is, integer hduIndex)

Loads a MetaData object from a FITS file at the specified HDU index.

MetaData loadMeta(InputStream is, integer hduIndex)

Loads a MetaData object using the current reader for HDU index 0. Loads a MetaData object using STANDARD_READER for HDU index greater than 0. The user must close the InputStream.

Arguments

InputStream **is** [INPUT, MANDATORY, default=no default value]

InputStream to the FITS file.

integer **hduIndex** [INPUT, MANDATORY, default=no default value]

HDU index.

Return**MetaData**

A MetaData object at the specified HDU index.

MetaData loadMeta(InputStream is, integer hduIndex, integer commentExtraSpacesToRemove)

Loads a MetaData object from a FITS file at the specified HDU index.

Loads a MetaData object using the current reader for HDU index 0. Loads a MetaData object using STANDARD_READER for HDU index greater than 0. The user must close the InputStream.

Arguments

InputStream **is** [INPUT, MANDATORY, default=no default value]

InputStream to the FITS file.

integer **hduIndex** [INPUT, MANDATORY, default=no default value]

HDU index.

integer **commentExtraSpacesToRemove** [INPUT, MANDATORY, default=no default value]

Comment extra spaces to be removed.

Return**MetaData**

A MetaData object at the specified HDU index.

MetaData loadMeta(String name, integer hduIndex, boolean handleMissingClasses)

Loads a MetaData object from a FITS file at the specified HDU index.

Loads a MetaData object using the current reader for HDU index 0. Loads a MetaData object using STANDARD_READER for HDU index greater than 0.

Arguments

[String](#) **name** [INPUT, MANDATORY, default=no default value]

Name of the FITS file

integer **hduIndex** [INPUT, MANDATORY, default=no default value]

HDU index.

boolean **handleMissingClasses** [INPUT, MANDATORY, default=no default value]

For creating dummy classes when a Hcss class is not found.

MetaData loadMeta([String](#) name, integer hduIndex, boolean handleMissingClasses)

Return

MetaData

A MetaData object at the specified HDU index.

MetaData loadMeta([String](#) name, integer hduIndex, integer commentExtraSpacesToRemove, boolean handleMissingClasses)

Loads a MetaData object from a FITS file at the specified HDU index.

Loads a MetaData object using the current reader for HDU index 0. Loads a MetaData object using STANDARD_READER for HDU index greater than 0.

Arguments

[String](#) **name** [INPUT, MANDATORY, default=no default value]

Name of the FITS file

integer **hduIndex** [INPUT, MANDATORY, default=no default value]

HDU index.

integer **commentExtraSpacesToRemove** [INPUT, MANDATORY, default=no default value]

Comment extra spaces to be removed.

boolean **handleMissingClasses** [INPUT, MANDATORY, default=no default value]

For creating dummy classes when a Hcss class is not found.

Return

MetaData

A MetaData object at the specified HDU index.

MetaData loadMeta([File](#) file, integer hduIndex, boolean handleMissingClasses)

Loads a MetaData object from a FITS file at the specified HDU index.

Loads a MetaData object using the current reader for HDU index 0. Loads a MetaData object using STANDARD_READER for HDU index greater than 0.

Arguments

[File](#) **file** [INPUT, MANDATORY, default=no default value]

Name of the FITS file

integer **hduIndex** [INPUT, MANDATORY, default=no default value]

HDU index.

boolean **handleMissingClasses** [INPUT, MANDATORY, default=no default value]

For creating dummy classes when a Hcss class is not found.

Return

MetaData

A MetaData object at the specified HDU index.

```
MetaData loadMeta(File file, integer hduIndex, integer commentExtraSpacesToRemove, boolean handleMissingClasses)
```

Loads a MetaData object from a FITS file at the specified HDU index.

Loads a MetaData object using the current reader for HDU index 0. Loads a MetaData object using STANDARD_READER for HDU index greater than 0.

Arguments

File file [INPUT, MANDATORY, default=no default value]

Name of the FITS file

integer **hduIndex** [INPUT, MANDATORY, default=no default value]

HDU index.

integer **commentExtraSpacesToRemove** [INPUT, MANDATORY, default=no default value]

Comment extra spaces to be removed.

boolean **handleMissingClasses** [INPUT, MANDATORY, default=no default value]

For creating dummy classes when a Hcss class is not found.

Return

MetaData

A MetaData object at the specified HDU index.

```
MetaData loadMeta(InputStream is, integer hduIndex, boolean handleMissingClasses)
```

Loads a MetaData object from a FITS file at the specified HDU index.

Loads a MetaData object using the current reader for HDU index 0. Loads a MetaData object using STANDARD_READER for HDU index greater than 0. The user must close the InputStream.

Arguments

InputStream **is** [INPUT, MANDATORY, default=no default value]

InputStream to the FITS file.

integer **hduIndex** [INPUT, MANDATORY, default=no default value]

HDU index.

boolean **handleMissingClasses** [INPUT, MANDATORY, default=no default value]

For creating dummy classes when a Hcss class is not found.

Return

MetaData

A MetaData object at the specified HDU index.

```
MetaData loadMeta(InputStream is, integer hduIndex, integer commentExtraSpacesToRemove, boolean handleMissingClasses)
```

Loads a MetaData object from a FITS file at the specified HDU index.

Loads a MetaData object using the current reader for HDU index 0. Loads a MetaData object using STANDARD_READER for HDU index greater than 0. The user must close the InputStream.

Arguments

```
MetaData loadMeta(InputStream is, integer hduIndex, integer commentExtraSpacesToRemove, boolean handleMissingClasses)
```

```
InputStream is [INPUT, MANDATORY, default=no default value]
```

InputStream to the FITS file.

```
integer hduIndex [INPUT, MANDATORY, default=no default value]
```

HDU index.

```
integer commentExtraSpacesToRemove [INPUT, MANDATORY, default=no default value]
```

Comment extra spaces to be removed.


```
boolean handleMissingClasses [INPUT, MANDATORY, default=no default value]
```

For creating dummy classes when a Hcss class is not found.

Return**MetaData**

A MetaData object at the specified HDU index.

2.131. fitsReader

Full Name:	herschel.ia.toolbox.util.FitsReaderTask
Alias:	fitsReader
Type:	Java Task - 
Import:	from herschel.ia.toolbox.util import FitsReaderTask
Category:	task

Description

The FitsReaderTask task.

Creates a "suitable" product from a FITS file, not necessarily from Herschel Currently it deals with images, (partially, only wave based) spectra, cubes and spectral cubes.

Examples

Example 1: FitsReaderTask with a file parameter

```
filepath = -"path_to_file"
product=fitsReader(file=filepath)
```

Example 2: FitsReaderTask to read an image

```
filepath = -"path_to_file"
product=fitsReader(file=filepath,
fitstype=herschel.ia.toolbox.util.FitsReaderTask.FitsType.IMAGE)
```

API Summary

Jython Syntax

```
product=fitsReader(<file>[, <FitsType>])
```

Properties

[String file](#) [INPUT, MANDATORY, default=null]

[FitsReaderTask.FitsType fitstype](#) [INPUT, OPTIONAL, default=FitsType.UNKNOWN]

[Object product](#) [OUTPUT, MANDATORY, default=null]

API details

Properties

[String file](#) [INPUT, MANDATORY, default=null]

The path of the FITS file to be read.

[FitsReaderTask.FitsType fitstype](#) [INPUT, OPTIONAL, default=FitsType.UNKNOWN]

The type of object the fits file contains (default is unknown).


<code>Object product [OUTPUT, MANDATORY, default=null]</code>

The Product (dataset for spectra) read from the FITS file.
--

History

- 2008-08-16 - JDS: initial commit (not allowed dependencies)
- 2009-09-04 - JDS: initial release (only images can be displayed, graphical editors fail to show)
- 2009-09-30 - JSS: modifiers are created through `getCustomModifiers` (SCR HCSS-8253)
- 2009-10-22 - JDS: release to be backported to 1.2 (no general solution for Spectra yet)

2.132. FitterFunction

Full Name:	herschel.ia.toolbox.fit.FitterFunction
Alias:	FitterFunction
Type:	Java Class - 
Import:	from herschel.ia.toolbox.fit import FitterFunction

Description

Specialization of RealFunction that fits some given data.

Example

Example 1: Implicit fitting

```
x = DoubleIcd([ 0,0.5,1,2, 3, 4, 5, 6, 10, 12])
y = DoubleIcd([-3, 2,4,5,12,37,92,189,1237,2325])
f = FitterFunction(x, y, PolynomialModel(3))
i = SimpsonIntegrator(1, 10)
print i.integrate(f) # 2668.499999999972
Explicit fitting
x = DoubleIcd([ 0,0.5,1,2, 3, 4, 5, 6, 10, 12])
y = DoubleIcd([-3, 2,4,5,12,37,92,189,1237,2325])
model = CubicSplinesModel(x)
fitter = AmoebaFitter(x, model)
fitter.setSimplex(params, range) # customize the fitter as you want
fitter.fit(y)
f = FitterFunction(model) # or f = FitterFunction(fitter)
# remaining code like before
```

API Summary

Jython Syntax

```
<f>=FitterFunction(<x>,<y>,<m>[,<c>])
<f>=FitterFunction(<m>)
<f>=FitterFunction(<F>)
```

Properties

[DoubleIcd of abscissas **x** \[INPUT, MANDATORY, default=no default value\]](#)

[DoubleIcd of values **y** \[INPUT, MANDATORY, default=no default value\]](#)

[AbstractModel **m** \[INPUT, MANDATORY, default=no default value\]](#)

[Class of Fitter **c** \[INPUT, OPTIONAL, default=Fitter.class\]](#)

[Fitter **F** \[INPUT, MANDATORY, default=no default value\]](#)

[RealFunction **f** \[OUTPUT, MANDATORY, default=no default value\]](#)

API details

Properties

DoubleIcd of abscissas **x [INPUT, MANDATORY, default=no default value]**

Abcissas; it is only mandatory for the constructors where it appears in the synopsis.

DoubleId of values y [INPUT, MANDATORY, default=no default value]

Values corresponding to the abscissas; it is only mandatory for 1st constructors.

[AbstractModel](#) m [INPUT, MANDATORY, default=no default value]

Unitialized model for 1st constructors, or already customized model for 2nd constructor.

Class of Fitter c [INPUT, OPTIONAL, default=Fitter.class]

Allows specifying the fitter class to be used in 1st constructor. FitterFunction provides some useful constants: LINEAR (for Fitter.class), AMOEBA (for AmoebaFitter.class) and LEVENBERG (for LevenbergMarquardtFitter.class).


Fitter F [INPUT, MANDATORY, default=no default value]

It is only mandatory for 3rd constructor.

RealFunction f [OUTPUT, MANDATORY, default=no default value]

Function object that fits the provided data with the specified model.

2.133. Fitter

Full Name:	herschel.ia.numeric.toolbox.fit.Fitter
Type:	Java Class - 
Import:	from herschel.ia.numeric.toolbox.fit import Fitter

Description

Fitter for linear models.

The Fitter class is to be used in conjunction with *Model classes.

The Fitter class and its descendants fit data to a model. Fitter itself is the variant for linear models, ie. models linear in its parameters.

For both linear and nonlinear models it holds that once the optimal estimate of the parameters is found, a variety of calculations is exactly the same: standard deviations, noise scale, evidence and model errors. They all derive more or less from the inverse Hessian matrix (aka the covariance matrix). All these calculations are in this Fitter class. Other Fitter classes relegate their calculation in these issues to this one.

An introduction to the use of fit package can be found [here](#).

At least once have a look at the [background](#) on the organization and structure of the package. The fit/demo directory contains worked [examples](#) on almost all aspects of of the package.

Example

Example 1: Fitter (for models which are linear in its parameters.)

```
# assume x and y are DoubleIcd data arrays.
poly = PolynomialModel( 1 -)           # line
fitter = Fitter( x, poly -)
param = fitter.fit( y -)
stdev = fitter.getStandardDeviation()  # stdevs on the parameters
chisq = fitter.getChiSquared()
scale = fitter.getScale()              # noise scale
yfit = fitter.getResult()             # fitted values
yfit = poly( x -)                     # fitted values (same as previous)
yband = fitter.monteCarloError()      # 1 sigma confidence region
```

Limitations

1. The Fitter does **not** work with limits.
2. The calculation of the evidence is an Gaussian approximation which is only exact for linear models with a fixed scale.

Miscellaneous

In case of problems look at the [trouble shooting](#) section.

2.134. FitterTask

Full Name:	herschel.ia.toolbox.fit.FitterTask
Alias:	FitterTask
Type:	Java Task - 
Import:	from herschel.ia.toolbox.fit import FitterTask
Category:	generic task

Description

generic module: FitterTask

Task shell around the package herschel.ia.numeric.toolbox.fit. Both a command line and a GUI interface are provided. With the GUI interface there is more of the Fitter package accessible than on this tasks command line. However not everything possible within the fitter package is accessible via this task, GUI or otherwise.

Most text fields in the GUI accept either numerics or JIDE variables or JIDE expressions.

Deprecated. Use ModelFitTask in stead.

In ModelFitTask 4 output parameters have disappeared: stdev, yfit, yband and evidence.

At the same time they reappeared as get-methods of the task: getStdev(), getYfit(), getYband() and getEvidence().

Because of the jython-stub convention the calls within HIPE (JIDE) stay the same. Parameters can be addressed in HIPE as taskname.item while get-methods can be addressed in the same way: taskname.item. Because of this fortuitous coincidence HIPE user will not be affected by this change.

Java developers do need to change their code.

Example

Example 1: FitterTask

```
<pre>
from herschel.hifi.generic.task import FitterTask
# Assume that `tt` and/or `data` are DoubleI'd's
# usage on command line:
ft = FitterTask()( x=tt, y=data, modelname="GaussModel" -)
ft.fittername = -"LevenbergMarquardtFitter"
ft()          # performs the execute method
print ft.result      # parameters of the fit
print ft.stdev       # standard deviations
print ft.fittername  # name of the fitter
# Etcetera, etc.
# use the GUI.
ft = FitterTask()
ft.gui = 1          # start the GUI
# from the GUI select the data, model, fitter, properties etc and run.
print ft.result      # parameters of the fit
print ft.stdev       # standard deviations
# Etcetera as before.
</pre>
```

API Summary

Jython Syntax

see example below

Properties
NumericData x [INPUT, OPTIONAL, default=null]
DoubleArray y [INPUT, MANDATORY, default=no]
Boolean indexview [INPUT, OPTIONAL, default=true]
DoubleArray weights [INPUT, OPTIONAL, default=null]
AbstractModel model [INPUT, OPTIONAL, default=null]
String modelname [INPUT, OPTIONAL, default=no]
ArrayIdData modelarg [INPUT, OPTIONAL, default=null]
Fitter fitter [INPUT, OPTIONAL, default=null]
String fittername [INPUT, OPTIONAL, default=no]
DoubleId result [OUTPUT, OPTIONAL, default=n/a]
DoubleId parameters [OUTPUT, OPTIONAL, default=n/a]
DoubleId stdev [OUTPUT, OPTIONAL, default=n/a]
Double chisq [OUTPUT, OPTIONAL, default=n/a]
DoubleArray yfit [OUTPUT, OPTIONAL, default=n/a]
DoubleArray yband [OUTPUT, OPTIONAL, default=n/a]
DoubleId prior [INPUT, OPTIONAL, default=null]
Double evidence [OUTPUT, OPTIONAL, default=n/a]
Double scale [OUTPUT, OPTIONAL, default=n/a]
Boolean auto [INPUT, OPTIONAL, default=true]
Double fixed [INPUT, OPTIONAL, default=1.0]
Double mixed [INPUT, OPTIONAL, default=0.0]
Double tolerance [INPUT, OPTIONAL, default=0.01]
Integer iterations [INPUT, OPTIONAL, default=10000]
Double temperature [INPUT, OPTIONAL, default=0.0]
Double cooling [INPUT, OPTIONAL, default=0.95]
Integer tempsteps [INPUT, OPTIONAL, default=100]
DoubleId initialpars [INPUT, OPTIONAL, default=from model]
DoubleId highlimits [INPUT, OPTIONAL, default=null]
DoubleId lowlimits [INPUT, OPTIONAL, default=null]
IntId keepfixed [INPUT, OPTIONAL, default=null]
DoubleId fixedvalues [INPUT, OPTIONAL, default=null]
Boolean gui [INPUT, OPTIONAL, default=false]

Limitations

Not everything which is possible using the package directly is accessible via this task.

API details

Properties

NumericData x [INPUT, OPTIONAL, default=null]
The independent variable(s) of the fit problem. If not provided, it will use indices ranging from 0 to len(y). If y is more-dimensional the proper 2-d indices are provided.


DoubleArray y [INPUT, MANDATORY, default=no]
The data to be fitted. It can be more-dimensional. E.g. a 2-dimensional map.
Boolean indexview [INPUT, OPTIONAL, default=true]
Applicable parameters appear in the same order as in the y DoubleArray This parameter is only active when y has 2 dimensions or more.
DoubleArray weights [INPUT, OPTIONAL, default=null]
The weights to be used in the fit. Can be more-dimensional. When provided it should be the same size as parameter y.
AbstractModel model [INPUT, OPTIONAL, default=null]
The model to be fitted. One of "model" or "modelname" is MANDATORY
String modelname [INPUT, OPTIONAL, default=no]
The name of the model to be fitted. One of "model" or "modelname" is MANDATORY
ArrayIdData modelarg [INPUT, OPTIONAL, default=null]
Arguments, if any, needed in the Constructor of the model.
Fitter fitter [INPUT, OPTIONAL, default=null]
The fitter to be used. One of "fitter" or "fittername" is MANDATORY
String fittername [INPUT, OPTIONAL, default=no]
The name of the fitter to be used. One of "fitter" or "fittername" is MANDATORY
DoubleId result [OUTPUT, OPTIONAL, default=n/a]
The parameters of the fit. Also the default result of the task.
DoubleId parameters [OUTPUT, OPTIONAL, default=n/a]
The parameters of the fit. (same as result)
DoubleId stdev [OUTPUT, OPTIONAL, default=n/a]
The standard deviations pertaining to the parameters.
Double chisq [OUTPUT, OPTIONAL, default=n/a]
Chi-squared of the fit.
DoubleArray yfit [OUTPUT, OPTIONAL, default=n/a]
An array of the same size as the data, containing the fitted values.
DoubleArray yband [OUTPUT, OPTIONAL, default=n/a]
An array of the same size as the data, containing a 1-# MonteCarlo error at each point.
DoubleId prior [INPUT, OPTIONAL, default=null]
Prior ranges for the parameters, needed when the evidence is requested. When noise scaling is set to auto=true (default), the prior needs 1 extra value as prior for the noise scale.

Double evidence [OUTPUT, OPTIONAL, default=n/a]
Relative evidence the model carries w.r.t. the data It needs values for prior, as many as the number of parameters and one more if auto-scaling is set.
Double scale [OUTPUT, OPTIONAL, default=n/a]
Noise scale of the data (when auto or mixed is selected)
Boolean auto [INPUT, OPTIONAL, default=true]
Select automatic noise scaling. The noise level in the data is not exactly known.
Double fixed [INPUT, OPTIONAL, default=1.0]
Fixed noise scale. The noise level is known.
Double mixed [INPUT, OPTIONAL, default=0.0]
Automatic noise scaling with a minimum. The noise level is not exactly known, but a minimum level is known.
Double tolerance [INPUT, OPTIONAL, default=0.01]
Stopping criterion for iterative fitting.
Integer iterations [INPUT, OPTIONAL, default=10000]
Maximum number of iterations in iterative fitters.
Double temperature [INPUT, OPTIONAL, default=0.0]
Starting temperature in annealing amoeba fitting.
Double cooling [INPUT, OPTIONAL, default=0.95]
Cooling step in annealing amoeba fitting.
Integer tempsteps [INPUT, OPTIONAL, default=100]
Number of exploratory steps at each temperature.
DoubleId initialpars [INPUT, OPTIONAL, default=from model]
Initial parameters in iterative fitters.
DoubleId highlimits [INPUT, OPTIONAL, default=null]
Upper limits of the parameters (only in AmoebaFitter)
DoubleId lowlimits [INPUT, OPTIONAL, default=null]
Lower limits of the parameters (only in AmoebaFitter)
IntId keepfixed [INPUT, OPTIONAL, default=null]
Keep these parameters fixed. (Parameter numbering starts at 0)
DoubleId fixedvalues [INPUT, OPTIONAL, default=null]
Values at which the parameters should be kept.
Boolean gui [INPUT, OPTIONAL, default=false]
Allows to handle this task using a gui.

History

- 15-10-2006 DK

2.135. FixedMask

Full Name:	herschel.ia.numeric.toolbox.mask.FixedMask
Type:	Java Class - 
Import:	from herschel.ia.numeric.toolbox.mask import FixedMask

Description

FixedMask represents the more "traditional" mask form, where a mask is defined at a fixed bit offset position.

The class represents only the definition, and does not allocate any data array itself. Defining a single mask array requires a minimum of 8 bits per data element (if byte is used) up to 64 bits (if long is used). All integer types are supported (byte/short/int/long). However, masks can be combined in the same array, so if 64 masks were stored in a long, it would be fully packed, with no memory wastage. This also means that there is a practical limit of 64 on the number of masks that can be defined. This mask still makes perfect sense when exported to an external format. It also integrates well with standard numeric library functionality. Note that the mask data itself is not stored in this class.

Examples

Example 1: Define some mask bits/setters

```
MASTER = FixedMask (0, -"Master")
NOISY = FixedMask (2, -"Noisy")
GLITCH = FixedMask (4, -"Glitch")
#where 0, 2, and 4 are bit offset
#Since a long integer can only holds 64 bits,an exception will be thrown out
if
    #TOOBIG=FixedMask(100,"Too Big")
#java.lang.IllegalArgumentException: bit offset must be >= 0 and less than
64, not 100
```

Example 2: Mask bits can be printed (continued from the previous example):

```
print MASTER
# Master @ 0 = 1
print NOISY
# Noisy @ 2 = 4
print GLITCH
# Glitch @ 4 = 16
print NOISY.getName()
#Noisy
print GLITCH.getValue()
#16
```

Example 3: Set/Unset mask data (continued from the previous example):

```
#A mask data x can be set/unset by different mask setters:
x = 0
x = GLITCH.set (x)
print x
#16
x = MASTER.set (x)
print x
# 17
x = GLITCH.unset (x)
print x
```

Example 3: Set/Unset mask data (continued from the previous example):

```
# 1
```

Example 4: Check mask data (continued from the previous example):

```
#The status of mask data x can be checked for different mask setters:
print MASTER.isSet(x)
# 1
print GLITCH.isSet(x)
# 0
# The mask checking can be ignored:
MASTER.setIgnore(1)
print MASTER.isSet(x)
#0
# The mask checking can be resumed:
MASTER.setIgnore(0)
print MASTER.isSet(x)
#1
```

Example 5: Use mask bits on a mask data array (continued from the previous example):

```
#One can also use mask bits/setters to work on a mask data array a:
a = Int1d([1,2,3,4,5,6,7,8,9])
print a.where(MASTER)
#[0,2,4,6,8] (Note. The method -"where" returns the indices where the mask
MASTER is set.)
print MASTER.isSet(a)
#[true,false,true,false,true,false,true,false,true]
print MASTER.unset(a)
#[0,2,2,4,4,6,6,8,8]
print MASTER.set(a)
#[1,3,3,5,5,7,7,9,9]
MASTER.setIgnore(1)
print MASTER.isSet(a)
#[false,false,false,false,false,false,false,false,false]
print a.where(MASTER)
#[[]]
MASTER.setIgnore(0)
```

Example 6: More examples to show mask is set, is not set, and the combined masks are set (continued from the previous example):

```
b = Int1d.range(20)
#Check where Master is set
print b.where(MASTER.isSet(b))
#[1,3,5,7,9,11,13,15,17,19]
#Check where Glitch is set
print b.where(GLITCH.isSet(b))
#[16,17,18,19]
#Check where Glitch is not set
print b.where(~GLITCH.isSet(b))
#[0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15]
#Check where Glitch or Master are set
print b.where(GLITCH.isSet(b) -| MASTER.isSet(b))
#[1,3,5,7,9,11,13,15,16,17,18,19]
#Check where Glitch and Master are set
print b.where(GLITCH.isSet(b) & MASTER.isSet(b))
#[17,19]
```

Example 7: The mask data array can be in high dimension (continued from the previous example):

```
aa = Int2d([[1,2,3],[4,5,6]])
print NOISY.isSet(aa)
#[ [false,false,false], [true,true,true] -]
```

Example 7: The mask data array can be in high dimension (continued from the previous example):

```
print aa.where(NOISY)
#[3,4,5]
```


Example 8: Mask bits/setters can be combined (continued from the previous example):

```
M_or_N = MASTER|NOISY
print M_or_N
#Master -| Noisy @ --1 = 5
b=Int1d([0,0,0,0,0])
print M_or_N.set(b)
#[5,5,5,5,5]
#which is equivalent to set MASTER and then NOISY:
M_or_N.unset(b)
print MASTER.set(b)
#[1,1,1,1,1]
print NOISY.set(b)
#[5,5,5,5,5]
#
M_and_N=MASTER & NOISY
print M_and_N
#Master & Noisy @ --1 = 0
print M_and_N.set(b)
#[5,5,5,5,5]
#Nothing is set because MASTER_AND_NOISY's value is 0 (they are different
single bit setters)
#
```

History

- June 2006 Original prototype
- September 2006 second go not using enum, subclass AbstractArrayPredicate
- February 2007 third iteration
- July 2007 fourth iteration
- May 2009 SPR-6833, SCR-4734
- November 2009 SCR-6541: registration

2.136. FixedSkyAperturePhotometryExplorer

Full Name:	herschel.ia.gui.image.FixedSkyAperturePhotometryExplorer
Type:	Java Class - 
Import:	from herschel.ia.gui.image import FixedSkyAperturePhotometryExplorer

Description

An explorer for FixedSkyAperturePhotometryProducts.

An explorer for FixedSkyAperturePhotometryProducts.

API Summary

Constructors
FixedSkyAperturePhotometryExplorer() The construction of a new FixedSkyAperturePhotometryExplorer.
FixedSkyAperturePhotometryExplorer(Object object) The construction of a new FixedSkyAperturePhotometryExplorer associated with the given object.

Methods
boolean canHandle(Class className) Checks whether this FixedSkyAperturePhotometryExplorer can handle objects of the given class.
setObject(Object object) Sets the object.
FixedSkyAperturePhotometryProduct getObject() Returns the object.
Class getVariableType() Returns the expected variable type.
JScrollPane getParameterTable() Returns the parameter table.
JScrollPane getResultsTable() Returns the results table.
JPanel getPlotPanel() Returns the plot panel.

API details

Constructors

FixedSkyAperturePhotometryExplorer() The construction of a new FixedSkyAperturePhotometryExplorer.
The construction of a new FixedSkyAperturePhotometryExplorer.

FixedSkyAperturePhotometryExplorer(Object object)

The construction of a new FixedSkyAperturePhotometryExplorer associated with the given object.

The construction of a new FixedSkyAperturePhotometryExplorer associated with the given object.

Argument

[Object](#) **object** [INPUT, MANDATORY]

Methods

boolean canHandle(Class className)

Checks whether this FixedSkyAperturePhotometryExplorer can handle objects of the given class.

Returns true if this FixedSkyAperturePhotometryExplorer can handle objects of the given class.

Argument

[Class](#) **className** [INPUT, MANDATORY]

Return

boolean

Returns true if this FixedSkyAperturePhotometryExplorer can handle objects of the given class; false otherwise.

setObject(Object object)

Sets the object.

Sets the object for this AperturePhotometryExplorer to the given object.

Argument

[Object](#) **object** [INPUT, MANDATORY]

FixedSkyAperturePhotometryProduct getObject()

Returns the object.

Returns the object for this FixedSkyAperturePhotometryExplorer.

Return

[FixedSkyAperturePhotometryProduct](#)

Returns the object for this FixedSkyAperturePhotometryExplorer.

Class getVariableType()

Returns the expected variable type.

Returns the expected variable type for this FixedSkyAperturePhotometryExplorer.

Return

[Class](#)

Returns the expected variable type for this FixedSkyAperturePhotometryExplorer.

JScrollPane getParameterTable()

Returns the parameter table.

[JScrollPane](#) getParameterTable()

Constructs and returns the parameter table for this FixedSkyAperturePhotometryExplorer.

Return

[JScrollPane](#)

Returns the parameter table for this FixedSkyAperturePhotometryExplorer.

[JScrollPane](#) getResultsTable()

Returns the results table.

Constructs and returns the results table for this AperturePhotometryExplorer.

Return

[JScrollPane](#)

Returns the results table for this AperturePhotometryExplorer.

[JPanel](#) getPlotPanel()

Returns the plot panel.

Constructs and returns the plot panel for this FixedSkyAperturePhotometryExplorer.

Return

[JPanel](#)

Returns the plot panel for this FixedSkyAperturePhotometryExplorer.

2.137. FixedSkyAperturePhotometryPanel

Full Name:	herschel.ia.toolbox.image.gui.FixedSkyAperturePhotometryPanel
Type:	Java Class - 
Import:	from herschel.ia.toolbox.image.gui import FixedSkyAperturePhotometryPanel

Description

A panel for the FixedSkyAperturePhotometryTask.

A panel to serve as GUI for the FixedSkyAperturePhotometryTask.

API Summary

Constructor
FixedSkyAperturePhotometryPanel() The construction of a new FixedSkyAperturePhotometryPanel.
Methods
JPanel getAperturesPanel() Returns the panel where to specify the target radius and the sky value.
JPanel getTargetAperturePanel() Returns the panel where to specify the target radius
JComponent getSkyApertureComponent() Returns the component where to specify the sky value.
drawFigures() Draws the circle on the image.
moveConfirmedFigures() Allows to move/resize the figures bounding the apertures.
transferFromModifierToSelectionMap() Transfers the information from the modifier to the selection map.
clearSkyAperture() Clears the sky aperture.

API details

Constructor


FixedSkyAperturePhotometryPanel()
The construction of a new FixedSkyAperturePhotometryPanel.
The construction of a new FixedSkyAperturePhotometryPanel.

Methods

JPanel getAperturesPanel()
Returns the panel where to specify the target radius and the sky value.

JPanel getAperturesPanel ()
Returns the panel where to specify the target radius and the sky value for this FixedSkyAperturePhotometryPanel.
Return
JPanel
Returns the panel where to specify the target radius and the sky value for this FixedSkyAperturePhotometryPanel.
JPanel getTargetAperturePanel ()
Returns the panel where to specify the target radius
Returns the panel where to specify the target radius for this FixedAperturePhotometryPanel.
Return
JPanel
Returns the panel where to specify the target radius for this FixedAperturePhotometryPanel.
JComponent getSkyApertureComponent ()
Returns the component where to specify the sky value.
Returns the component where to specify the sky value for this FixedSkyAperturePhotometryPanel.
Return
JComponent
Returns the component where to specify the sky value for this FixedSkyAperturePhotometryPanel.
drawFigures ()
Draws the circle on the image.
Draws the circle on the image for this FixedSkyAperturePhotometryPanel.
moveConfirmedFigures ()
Allows to move/resize the figures bounding the apertures.
Allows to move/resize the figures bounding the apertures for this FixedAperturePhotometryPanel.
transferFromModifierToSelectionMap ()
Transfers the information from the modifier to the selection map.
Transfers the information for this FixedSkyAperturePhotometryPanel from the modifier to the selection map.
clearSkyAperture ()
Clears the sky aperture.
Clears the sky aperture for this FixedAperturePhotometryPanel.

2.138. FixedSkyAperturePhotometryProduct

Full Name:	herschel.ia.dataset.image.FixedSkyAperturePhotometryProduct
Type:	Java Class - 
Import:	from herschel.ia.dataset.image import FixedSkyAperturePhotometryProduct

Description

A class to deal with the results of aperture photometry with a circular target aperture and a fixed sky value.

API Summary

Constructor
<p>FixedSkyAperturePhotometryProduct()</p> <p>The constructor of a new FixedSkyAperturePhotometryProduct.</p>
Methods
<p>setSkyValue(double sky)</p> <p>Sets the sky value for this FixedSkyAperturePhotometryProduct to the</p>
<p>setResultsTable(Double2d resultsTable)</p> <p>Sets the results table for this AperturePhotometryProduct to the</p>
<p>double getSkyValue()</p> <p>Returns the sky value for this FixedSkyAperturePhotometryProduct.</p>
<p>double getIntensityPerSkyPixel()</p> <p>Returns the intensity per pixel for the sky for this FixedSkyAperturePhotometryProduct.</p>
<p>double getTargetTotal()</p> <p>Returns the total flux for the target (sky subtracted) for this</p>
<p>double getNbOfTargetPixels()</p> <p>Returns the number of pixels for the target (sky subtracted) for this</p>
<p>double getIntensityPerTargetPixel()</p> <p>Returns the intensity per pixel for the target (sky subtracted) for this</p>
<p>double getTargetError()</p> <p>Returns the error for the target (sky subtracted) for this</p>

Miscellaneous

Aperture specific information is stored in meta data, all other data (results of calculations) are stored in TableDatasets.

API details

Constructor

FixedSkyAperturePhotometryProduct()
The constructor of a new FixedSkyAperturePhotometryProduct.

Methods

setSkyValue(double sky)
Sets the sky value for this FixedSkyAperturePhotometryProduct to the given sky value.
Argument
double sky [INPUT, MANDATORY]
setResultsTable(Double2d resultsTable)
Sets the results table for this AperturePhotometryProduct to the given table.
Argument
Double2d resultsTable [INPUT, MANDATORY]
double getSkyValue()
Returns the sky value for this FixedSkyAperturePhotometryProduct.
Return
double
Returns the sky value for this FixedSkyAperturePhotometryProduct.
double getIntensityPerSkyPixel()
Returns the intensity per pixel for the sky for this FixedSkyAperturePhotometryProduct.
Return
double
Returns the intensity per pixel for the sky for this FixedSkyAperturePhotometryProduct.
double getTargetTotal()
Returns the total flux for the target (sky subtracted) for this FixedSkyAperturePhotometryProduct.
Return
double
Returns the total flux for the target (sky subtracted) for this FixedSkyAperturePhotometryProduct.
double getNbOfTargetPixels()
Returns the number of pixels for the target (sky subtracted) for this FixedSkyAperturePhotometryProduct.
Return
double

double getNbOfTargetPixels()

Returns the number of pixels for the target (sky subtracted) for this FixedSkyAperturePhotometryProduct.

double getIntensityPerTargetPixel()

Returns the intensity per pixel for the target (sky subtracted) for this FixedSkyAperturePhotometryProduct.

Return

double

Returns the intensity per pixel for the target (sky subtracted) for this FixedSkyAperturePhotometryProduct.

double getTargetError()


Returns the error for the target (sky subtracted) for this FixedSkyAperturePhotometryProduct.

Return

double

Returns the error for the target (sky subtracted) for this FixedSkyAperturePhotometryProduct.

2.139. FixedSkyAperturePhotometryTask

Full Name:	herschel.ia.toolbox.image.FixedSkyAperturePhotometryTask
Type:	Java Task - 
Import:	from herschel.ia.toolbox.image import FixedSkyAperturePhotometryTask

Description

A Task for aperture photometry.

A Task for aperture photometry with a circular target aperture and a fixed value for the sky.

API Summary

Properties
Image image [INPUT, MANDATORY, default=No default value]
Double centerX [INPUT, OPTIONAL, default=Default value : NaN]
Double centerY [INPUT, OPTIONAL, default=Default value : NaN]
String centerRA [INPUT, OPTIONAL, default=Default value : "center-RA"]
String centerDEC [INPUT, OPTIONAL, default=Default value : "centerDec"]
Double radiusPixels [INPUT, OPTIONAL, default=Default value : NaN]
Double radiusArcsec [INPUT, OPTIONAL, default=Default value : NaN]
Double sky [INPUT, MANDATORY, default=Default value : 0.0]
boolean fractional [INPUT, OPTIONAL, default=Default value : true]
AperturePhotometryProduct result [OUTPUT, MANDATORY, default=No default value]

Miscellaneous

The calculation of the error on the fluxes is calculated in the same way as in the aper.pro routine of IDL. The error on the target flux including the background is the sqrt of the (absolute value of the) total flux in the target aperture (including the background). The error on the target flux (with the background subtracted) is defined as the sqrt of the (absolute value of the) total flux in the target aperture (with the background subtracted).

API details

Properties

Image image [INPUT, MANDATORY, default=No default value]
The image.
Double centerX [INPUT, OPTIONAL, default=Default value : NaN]
The x-pixel-coordinate of the target center.
Double centerY [INPUT, OPTIONAL, default=Default value : NaN]
The y-pixel-coordinate of the target center.

String centerRA [INPUT, OPTIONAL, default=Default value : "centerRA"]

The right ascension of the target center.

String centerDEC [INPUT, OPTIONAL, default=Default value : "centerDec"]

The declination of the target center.

Double radiusPixels [INPUT, OPTIONAL, default=Default value : NaN]

The target radius (i.e. the radius of the circular target aperture) in pixels.

Double radiusArcsec [INPUT, OPTIONAL, default=Default value : NaN]

The target radius in arcsec.

Double sky [INPUT, MANDATORY, default=Default value : 0.0]

The value for the sky.


boolean fractional [INPUT, OPTIONAL, default=Default value : true]

The type of pixels.

AperturePhotometryProduct result [OUTPUT, MANDATORY, default=No default value]

The result.

2.140. FIX

Full Name:	herschel.ia.numeric.toolbox.basic.Fix
Alias:	FIX
Type:	Java Class - 
Import:	from herschel.ia.numeric.toolbox.basic import Fix

Description

Gives the largest long less than or equal to x.

Returns a long array. Only float or double arrays are allowed as input argument.

Example

Example 1: Apply FIX on a Float1d

```
x=Float1d([1.0,2.1,3.5,4.9])
print FIX(x) # [1,2,3,4]
#To change the returned array type into an Int1d array:
i = Int1d(FIX(x))
```

API Summary

Jython Syntax

```
<y>=FIX(<x>)
```

Properties

[any array of any rank **x** \[INPUT, MANDATORY, default=no default value\]](#)

[long array **y** \[OUTPUT, MANDATORY, default=no default value\]](#)

API details

Properties

any array of any rank **x [INPUT, MANDATORY, default=no default value]**

The input array can be an array of rank 1. Only float or double arrays are allowed.

long array **y [OUTPUT, MANDATORY, default=no default value]**

Returns a long array. In order to create an array of a different type, you have to create the new array type using the returned array as an argument to the constructor (see examples).

See also

- [CEIL](#)
- [FLOOR](#)
- [ROUND](#)

2.141. Flag

Full Name:	herschel.ia.dataset.image.Flag
Alias:	Flag
Type:	Java Class - 
Import:	from herschel.ia.dataset.image import Flag
Category:	Image

Description

A class to describe Flags for images (SimpleImage, SimpleCube, SimpleStack, SlicedImage, Sliced-Cube and SlicedStack).

This class represents different kinds of flags in n-dimensions. The Flag is defined as an ArrayDataset. The different flag types are defined in the metadata of the ArrayDataset, a Shortnd describes the flags.

Example

Example 1: A basic example on how to create a Flag for an SimpleImage with dimensions 310 x 200.

```
flag = Flag(200, 310)
# Only the -"UNVALID" flag type is available after constructing the Flag.
# Adding an extra -"GLITCH" flag type.
flag.addFlagType("GLITCH", -"Flag for signals affected by a cosmic ray hit")
# Adding information about the flagged pixels.
# validFlagData is a Bool2d, with the same dimensions of the flag (310 x 200)
flag.setFlag("UNVALID", validFlagData)
flag.setFlag("GLITCH", glitchFlagData)
# Getting information on Flagged pixels
print flag.getFlag("GLITCH")
```

API Summary

Constructors	
Flag()	The standard constructor for a Flag
Flag(int k)	Constructor for a 1 dimensional flag
Flag(int k, int l)	Constructor for a 2 dimensional flag
Flag(int k, int l, int m)	Constructor for a 3 dimensional flag
Flag(int k, int l, int m, int n)	A constructor for a four-dimensional flag of given dimensions.
Flag(int k, int l, int m, int n, int o)	Constructor for a 5 dimensional flag
Flag(Flag flag)	The copy constructor

Methods
<p>Flag copy()</p> <p>The copy method returns a new Flag which is a copy of the original flag</p>
<p>addFlagType(String flagType, String description)</p> <p>Defines a new flag type.</p>
<p>StringId getFlagTypes()</p> <p>Give a list with the defined flag types.</p>
<p>boolean hasFlag(String name)</p> <p>Checks if the flag exists.</p>
<p>AbstractArrayData getFlag(String flagType)</p> <p>Gives the flag for this specific flag type.</p>
<p>AbstractArrayData getFlag()</p> <p>Gives the flag.</p>
<p>AbstractArrayData getFlagAsShortnd()</p> <p>Gives the flag.</p>
<p>setFlag(String flagType, AbstractArrayData flag)</p> <p>Sets the flag for this specific flag type.</p>
<p>setFlag(String flagType, int index, boolean flag)</p> <p>Sets the flag of a certain pixel.</p>
<p>setFlag(String flagType, int row, int column, boolean flag)</p> <p>Sets the flag of a certain pixel.</p>
<p>setFlag(String flagType, int index1, int index2, int index3, boolean flag)</p> <p>Sets the flag of a certain pixel.</p>
<p>setFlag(String flagType, int index1, int index2, int index3, int index4, boolean flag)</p> <p>Sets the flag of a certain pixel.</p>
<p>setFlag(String flagType, int index1, int index2, int index3, int index4, int index5, boolean flag)</p> <p>Sets the flag of a certain pixel.</p>

API details

Constructors

<p>Flag()</p> <p>The standard constructor for a Flag</p> <p>A constructor for a flag. After construction, 1 flagtype is available : UNVALID</p>
<p>Flag(int k)</p> <p>Constructor for a 1 dimensional flag</p> <p>A constructor for a one-dimensional flag of given dimension. A Flag of k elements in 1 dimension is made. No pixels are masked out. After construction, 1 flagtype is available : UNVALID</p> <p>Argument</p>

Flag(int k)

```
int k [INPUT, MANDATORY]
```

Example

Typical example on how to create a one-dimensional Flag.

```
flag = Flag(100)
```

Flag(int k, int l)

Constructor for a 2 dimensional flag

A constructor for a two-dimensional flag of given dimensions. A Flag of (k, l) elements in 2 dimensions is made. No pixels are masked out. After construction, 1 flagtype is available : UNVALID

Arguments

```
int k [INPUT, MANDATORY]
```

```
int l [INPUT, MANDATORY]
```

Example

Typical example on how to create a 2-dimensional Flag.

```
flag = Flag(100, 50)
```

Flag(int k, int l, int m)

Constructor for a 3 dimensional flag

A constructor for a three-dimensional flag of given dimensions. A Flag of (k, l, m) elements in 3 dimensions is made. No pixels are masked out. After construction, 1 flagtype is available : UNVALID

Arguments

```
int k [INPUT, MANDATORY]
```

```
int l [INPUT, MANDATORY]
```

```
int m [INPUT, MANDATORY]
```

Example

Typical example on how to create a 3-dimensional Flag.

```
flag = Flag(100, 50, 75)
```

Flag(int k, int l, int m, int n)

A constructor for a four-dimensional flag of given dimensions.

A Flag of (k, l, m, n) elements in 4 dimensions is made. No pixels are masked out. After construction, 1 flagtype is available : UNVALID

Arguments

```
int k [INPUT, MANDATORY]
```

```
int l [INPUT, MANDATORY]
```

```
int m [INPUT, MANDATORY]
```

```
int n [INPUT, MANDATORY]
```

Example

Typical example on how to create a 4-dimensional Flag.

Flag(int k, int l, int m, int n)

```
flag = Flag(100, 50, 75, 125)
```

Flag(int k, int l, int m, int n, int o)

Constructor for a 5 dimensional flag

A constructor for a five-dimensional flag of given dimensions. A Flag of (k, l, m, n, o) elements in 5 dimensions is made. No pixels are masked out. After construction, 1 flagtype is available : UNVALID

Arguments

int **k** [INPUT, MANDATORY]

int **l** [INPUT, MANDATORY]

int **m** [INPUT, MANDATORY]

int **n** [INPUT, MANDATORY]

int **o** [INPUT, MANDATORY]

Example

Typical example on how to create a 5-dimensional Flag.

```
flag = Flag(100, 50, 75, 125, 25)
```

Flag(Flag flag)

The copy constructor

The copy constructor makes an exact copy of the flag.

Argument

[Flag](#) **flag** [INPUT, MANDATORY]

Methods

[Flag](#) copy()

The copy method returns a new Flag which is a copy of the original flag

The copy method returns a new Flag which is a copy of the original flag

Return

[Flag](#)

Dataset A new Flag which is a copy of the current Flag.

addFlagType([String](#) flagType, [String](#) description)

Defines a new flag type.

Define a new flag type : a temporary flag type that can be used at the user's discretion in the processing. This method guarantees that the identifier used does not interfere with existing identifiers. There can be no more than 16 different flag types.

Arguments

[String](#) **flagType** [INPUT, MANDATORY]

[String](#) **description** [INPUT, MANDATORY]

StringId getFlagTypes()

Give a list with the defined flag types.

Give a list of the defined flag types. A StringId with all defined flag types is returned.

Return

StringId

The list of the mask type identifiers

boolean hasFlag(String name)

Checks if the flag exists.

Checks if the flag type exists. Returns true if the flag has the given flag type.

Argument

String name [INPUT, MANDATORY]

Return

boolean

The list of the mask type identifiers Returns true if the flag has the given flag type.

AbstractArrayData getFlag(String flagType)

Gives the flag for this specific flag type.

Gives the flag for this specific flag type as an AbstractArrayData. Depending on the dimensions of the flag, a Boolnd is returned with the flag for a given flag type.

Argument

String flagType [INPUT, MANDATORY]

Return

AbstractArrayData

A Boolnd (depending on the dimension of the flag), describing the flag of the given flagType.

AbstractArrayData getFlag()

Gives the flag.

Gives the flag as an AbstractArrayData. Depending on the dimensions of the flag, a Boolnd is returned with the flag.

Return

AbstractArrayData

A Boolnd (depending on the dimension of the flag), describing the flag

AbstractArrayData getFlagAsShortnd()

Gives the flag.

Gives the flag as an AbstractArrayData. Depending on the dimensions of the flag, a Shortnd is returned with the flag.

AbstractArrayData getFlagAsShortnd()

Return

AbstractArrayData

A Shortnd (depending on the dimension of the flag), describing the flag

setFlag([String](#) flagType, AbstractArrayData flag)

Sets the flag for this specific flag type.

Sets the flag for this specific flag type as an AbstractArrayData. A boolnd can be given for a flag type. If the boolnd does not have the same dimension as the dimension of the flag, an IllegalArgumentException is thrown.

Arguments

[String](#) **flagType** [INPUT, MANDATORY]

AbstractArrayData **flag** [INPUT, MANDATORY]

setFlag([String](#) flagType, int index, boolean flag)

Sets the flag of a certain pixel.

Set a certain flag of a certain flagType to a certain value

Arguments

[String](#) **flagType** [INPUT, MANDATORY]

int **index** [INPUT, MANDATORY]

boolean **flag** [INPUT, MANDATORY]

setFlag([String](#) flagType, int row, int column, boolean flag)

Sets the flag of a certain pixel.

Set a certain flag of a certain flagType to a certain value

Arguments

[String](#) **flagType** [INPUT, MANDATORY]

int **row** [INPUT, MANDATORY]

int **column** [INPUT, MANDATORY]

boolean **flag** [INPUT, MANDATORY]

setFlag([String](#) flagType, int index1, int index2, int index3, boolean flag)

Sets the flag of a certain pixel.

Set a certain flag of a certain flagType to a certain value

Arguments

[String](#) **flagType** [INPUT, MANDATORY]

int **index1** [INPUT, MANDATORY]

int **index2** [INPUT, MANDATORY]

int **index3** [INPUT, MANDATORY]

boolean **flag** [INPUT, MANDATORY]

```
setFlag(String flagType, int index1, int index2, int index3, int index4, boolean flag)
```

Sets the flag of a certain pixel.

Set a certain flag of a certain flagType to a certain value

Arguments

```
String flagType [INPUT, MANDATORY]  
int index1 [INPUT, MANDATORY]  
int index2 [INPUT, MANDATORY]  
int index3 [INPUT, MANDATORY]  
int index4 [INPUT, MANDATORY]  
boolean flag [INPUT, MANDATORY]
```

```
setFlag(String flagType, int index1, int index2, int index3, int index4, int index5, boolean flag)
```

Sets the flag of a certain pixel.

Set a certain flag of a certain flagType to a certain value


Arguments

```
String flagType [INPUT, MANDATORY]  
int index1 [INPUT, MANDATORY]  
int index2 [INPUT, MANDATORY]  
int index3 [INPUT, MANDATORY]  
int index4 [INPUT, MANDATORY]  
int index5 [INPUT, MANDATORY]  
boolean flag [INPUT, MANDATORY]
```

See also

- [SimpleImage](#)
- [SimpleCube](#)

2.142. FlagPixels

Full Name:	herschel.ia.toolbox.spectrum.FlagPixels
Type:	Java Task - 
Import:	from herschel.ia.toolbox.spectrum import FlagPixels

Description

Task for flagging pixels of spectra included in a spectrum container.

The flagging task always modifies the input data.

You have different alternatives to specify a 'mask' with the pixel indices to be flagged.

- **General mask:** For each segment within a specific point spectrum specify an entry in a dictionary, with keys given by tuples (i,j) (where i stands for the point spectrum index and j for the segment index) and values specifying the list of pixel to be flagged.
- **Uniform masks:** For all the point spectra (possibly restricted by a suitable selection model - see the 'selection' parameter below) the same mask apply. This is given again by a dictionary where the keys specify the segment index (an integer) and the values specify the list of pixel indices to be flagged. Alternatively, you can directly specify a java.util.Map with Integer keys and herschel.ia.dataset.Selection as values. The values specified as Selection's contain the indices of the pixels to be flagged. Alternatively, instead of a Selection you can also specify an Int1d containing the indices or a Bool1d with 'true' at the positions the flag should be set.
- **Uniform masks, uniform for several segments:** Specify a list of pixel indices to be flagged. This mask applies to all the segments and point spectra - possibly restricted by a suitable 'selection' and 'segments' indices.

As input spectra you can specify not only a single spectrum container but also a list of spectrum containers. Alternatively, you can specify a list of tuples with the first argument in the tuple specifying the spectrum container and the second argument the mask with the pixels to be flagged.

Example

Example 1: from Jide:

```
# Set flag 2 in point spectrum 2 and segments 1 at the pixel indices 1,2,667
and
# in point spectrum 2 and segment 3 at the pixel indices 30,690.
flagPixels(ds=spectra, mask={(2,1):[1,20,667], (2,3):[30,690]}, flag=2)
# Set flag 2 in segments 1 and 2 at the indices 1,2,667 and 30,690
respectively.
flagPixels(ds=spectra, mask={1:[1,20,667], 2:[30,690]}, flag=2)
# flag power(2,30) is set at the same positions.
flagPixels(ds=spectra, mask={1:[1,20,667], 2:[30,690]})
# Set flag 2 in segments 1 and 2 at the indices 1,2,667 and 30,690
respectively,
# but only in the point spectra with indices [0,1,2,3]
# For further ways of how to specify selections see e.g. the AverageSpectrum-
task
flagPixels(ds=spectra, selection=[0,1,2,3], mask={1:[1,20,667], 2:[30,690]},
flag=2)
# Set flag 2 at the pixels 1,20,667 in the segments 1,2,3 in point spectra
with the bdtype-attribute equal to 6031.
flagPixels(ds=spectra, mask=[1,20,667], segments=[1,2,3], selection={"bdtype":
[6031]}, flag=2)
# For spectral and spectra2 two SpectrumContainers:
# Set flag 2 at the pixels 1,20,667 in the segments 1,2,3 for spectral and
```

Example 1: from Jide:

```
# for spectra2 in point spectrum 2 and segments 1 at the pixel indices
1,2,667 and
# in point spectrum 2 and segment 3 at the pixel indices 30,690.
flagPixels(ds=[(spectral,[1,20,667]),(spectra2,{(2,1):[1,20,667], (2,3):
[30,690]})], segments=[1,2,3], flag=2)
# With a single container you can just specify one tuple:
flagPixels(ds=(spectra,{(2,1):[1,20,667], (2,3):[30,690]}), flag=2)
```

API Summary

Properties
Object ds [INOUT, MANDATORY, default=no default value.]
Object mask [INPUT, OPTIONAL, default=None.]
Object selection [INPUT, OPTIONAL, default=None.]
int flag [INPUT, OPTIONAL, default=2 up to the power of 30 = 1073741824.]
Boolean setFluxToNaN [INPUT, OPTIONAL, default=False.]

API details


Properties

Object ds [INOUT, MANDATORY, default=no default value.]
The input container(s) in which pixels should be flagged. See examples below.
Object mask [INPUT, OPTIONAL, default=None.]
Specification of the pixels to flag. See the description above and the examples below.
Object selection [INPUT, OPTIONAL, default=None.]
Specification of what point spectra the flagging mask should be applied to. Different ways to specify these selections are possible:
<ul style="list-style-type: none"> • Specify a list of indices (in jython) of the point spectra for which the mask should be applied. • Use a dictionary (in jython) to specify a selection model (lookup for attributes, filter attributes to be included in ranges or intervals). • Pass a string representation of a jython list or a jython dictionary. The string will be parsed and the processing delegated to the two cases above. • Pass any java instance that implements the SelectionModel interface (for the advanced user).
int flag [INPUT, OPTIONAL, default=2 up to the power of 30 = 1073741824.]
The flag value to set for the specified pixel.
Boolean setFluxToNaN [INPUT, OPTIONAL, default=False.]
Boolean to indicate that the flux value of the pixels to flag should be set to NaN. This will typically remove the pixel from plots.

History

- 2009-04-29 - meli: initial
- 2009-05-31 - meli: first extensions

2.143. FlagSaturatedPixelsCubeTask

Full Name:	herschel.ia.toolbox.cube.FlagSaturatedPixelsCubeTask
Type:	Java Task - 
Import:	from herschel.ia.toolbox.cube import FlagSaturatedPixelsCubeTask

Description

A Task to flag saturated pixels in a cube.

API Summary

Properties
Cube cube [INPUT, MANDATORY, default=No default value]
Double value [INPUT, MANDATORY, default=No default value]
Cube result [OUTPUT, MANDATORY, default=No default value]

API details

Properties

Cube cube [INPUT, MANDATORY, default=No default value]
The cube.
Double value [INPUT, MANDATORY, default=No default value]
The value of saturation.
Cube result [OUTPUT, MANDATORY, default=No default value]
The resulting cube.

2.144. FlagSaturatedPixelsTask

Full Name:	herschel.ia.toolbox.image.FlagSaturatedPixelsTask
Type:	Java Task - 
Import:	from herschel.ia.toolbox.image import FlagSaturatedPixelsTask

Description

A Task to flag saturated pixels.

A Task to flag saturated pixels in an image.

API Summary


Properties
Image image [INPUT, MANDATORY, default=No default value]
Double value [INPUT, MANDATORY, default=No default value]
Image flaggedImage [OUTPUT, MANDATORY, default=No default value]

API details

Properties

Image image [INPUT, MANDATORY, default=No default value]
The image.
Double value [INPUT, MANDATORY, default=No default value]
The value of saturation.
Image flaggedImage [OUTPUT, MANDATORY, default=No default value]
The resulting image.

2.145. Float1d


Full Name:	herschel.ia.numeric.Float1d
Type:	Java Class - 
Import:	from herschel.ia.numeric import Float1d

Description

A rectangular numeric float array of rank 1.

Numeric arrays are part of the Numeric library. An explanation on how to use them, you can find in chapter **Numeric** -> **Arrays**. (Yes, it should be a link)

2.146. Float2d


Full Name:	herschel.ia.numeric.Float2d
Type:	Java Class - 
Import:	from herschel.ia.numeric import Float2d

Description

A rectangular numeric float array of rank 2.

Numeric arrays are part of the Numeric library. An explanation on how to use them, you can find in chapter **Numeric** -> **Arrays**. (Yes, it should be a link)

2.147. Float3d


Full Name:	herschel.ia.numeric.Float3d
Type:	Java Class - 
Import:	from herschel.ia.numeric import Float3d

Description

A rectangular numeric float array of rank 3.

Numeric arrays are part of the Numeric library. An explanation on how to use them, you can find in chapter **Numeric** -> **Arrays**. (Yes, it should be a link)

2.148. Float4d


Full Name:	herschel.ia.numeric.Float4d
Type:	Java Class - 
Import:	from herschel.ia.numeric import Float4d

Description

A rectangular numeric float array of rank 4.

Numeric arrays are part of the Numeric library. An explanation on how to use them, you can find in chapter **Numeric** -> **Arrays**. (Yes, it should be a link)

2.149. Float5d


Full Name:	herschel.ia.numeric.Float5d
Type:	Java Class - 
Import:	from herschel.ia.numeric import Float5d

Description

A rectangular numeric float array of rank 5.

Numeric arrays are part of the Numeric library. An explanation on how to use them, you can find in chapter **Numeric** -> **Arrays**. (Yes, it should be a link)

2.150. FLOOR

Full Name:	herschel.ia.numeric.toolbox.basic.Floor
Alias:	FLOOR
Type:	Java Class - 
Import:	from herschel.ia.numeric.toolbox.basic import Floor

Description

Gives the largest integer less than or equal to x .

Returns a float array for float input array and double array for any other numeric array type.

Example

Example 1: Apply FLOOR on a Float1d

```
x=Float1d([0.1,0.5,0.9])
print FLOOR(x) # [0.0,0.0,0.0]
```

API Summary

Jython Syntax

```
<y>=FLOOR(<x>)
```

Properties

[any array of any rank \$x\$ \[INPUT, MANDATORY, default=no default value\]](#)

[float or double array \$y\$ \[OUTPUT, MANDATORY, default=no default value\]](#)

API details

Properties

any array of any rank x [INPUT, MANDATORY, default=no default value]

The input array can be an array of rank 1


float or double array y [OUTPUT, MANDATORY, default=no default value]

Returns a float array for float input array and double array for any other numeric array type.

See also

- [CEIL](#)
- [ROUND](#)

2.151. FoldSpectrum

Full Name:	herschel.ia.toolbox.spectrum.FoldSpectrum
Type:	Java Task - 
Import:	from herschel.ia.toolbox.spectrum import FoldSpectrum

Description

Task for folding frequency-switched spectra.

The folded spectra are constructed by averaging the original spectra with a shifted and inverted copy. This algorithm corresponds to the most simple scheme described in article ["Recovering line profiles from frequency-switched spectra"](#), H.Liszt, Astron. Astrophys. Suppl. Ser. 124, 183-188 (1997)}. By default, the task modifies the input spectra. Note that the operation is performed on a per segment basis. For that reason you may consider stitching the segments before folding.

API Summary

Properties
SpectrumContainer ds [INPUT, MANDATORY, default=no default value.]
SpectrumContainer result [OUTPUT, MANDATORY, default=no default value.]
Object freqThrow [INPUT, MANDATORY, default='LoThrow']
String unit [INPUT, OPTIONAL, default=no default value.]
Boolean shift [INPUT, OPTIONAL, default=False.]
Boolean overwrite [INPUT, OPTIONAL, default=True.]

API details

Properties

SpectrumContainer ds [INPUT, MANDATORY, default=no default value.]
Input container with the spectra to be folded.
SpectrumContainer result [OUTPUT, MANDATORY, default=no default value.]
Output container with the folded spectra.
Object freqThrow [INPUT, MANDATORY, default='LoThrow']
Either you can pass here directly the frequency throw to be used when folding the spectra. Or, alternatively, specify the name of the attribute (e.g. column name) where the frequency throw can be found. By default, 'LoThrow' (or 'loThrow') is used as value if an attribute with this name or a meta data item with with key is available. If a double value is provided or the throw taken from the meta data, the same double value is applied to all the spectra in the container. If the throw is specified as an attribute name (or in the default) it is typically dependent on the point spectrum.

String `unit` [INPUT, OPTIONAL, default=no default value.]

Specify the unit the frequency throw is expressed in. If no unit is specified it is assumed that the unit of the frequency throw is assumed to be the same as the one found on the wave scale. Typical values are "GHz" or "MHz".


Boolean `shift` [INPUT, OPTIONAL, default=False.]

If true, the spectrum is shifted in frequency scale by half the throw distance so that the resulting spectrum is centred between the original and the "switched" spectrum.

Boolean `overwrite` [INPUT, OPTIONAL, default=True.]

Specify whether the input data container can be reused - the values found therein are overwritten. If set to false, a new container is created and the spectral segments are reduced in shape by the frequency throw distance.

2.152. FullQuery

Full Name:	herschel.ia.pal.query.FullQuery
Type:	Java Class - 
Import:	from herschel.ia.pal.query import FullQuery

Description

A data mining query formulates a query the full interface of a

Product. As such, one can query any aspect of a Product. Typically this type of query is quite slow and should be used in combination with query refinements.

Example


Example 1: Example of a data mining query

```
q=FullQuery(MyProduct.class,"p", -"ANY(p['array'].data<2)")
```

See also

- [???](#)

2.153. GAMMALN

Full Name:	herschel.ia.numeric.toolbox.basic.GammaLn
Alias:	GAMMALN
Type:	Java Class - 
Import:	from herschel.ia.numeric.toolbox.basic import GammaLn

Description

Computes the natural log of the Gamma function.

Example

Example 1: Apply GAMMALN on a Double1d

```
x = Double1d([1.0,2.0,3.0,4.0,5.0,6.0])
gn = GAMMALN(x)
print gn # [0.0,-4.440892098500626E-16,0.6931471805599443,
          1.791759469228055,3.1780538303479453,4.787491742782044]
```

API Summary

Jython Syntax

```
<y>=GAMMALN(<x>)
```

Properties

[an double array or a number **x** \[INPUT, MANDATORY, default=.\]](#)

[returns an array **y** \[OUTPUT, \(or a number if the input is a number\), default=no default value\]](#)

API details

Properties

an double array or a number **x [INPUT, MANDATORY, default=.]**


returns an array **y [OUTPUT, (or a number if the input is a number), default=no default value]**

where each element is the natural logarithm of the Gamma function of the corresponding element of the input array.

See also

- [GammaP](#)
- [GammaQ](#)

2.154. GammaP

Full Name:	herschel.ia.numeric.toolbox.basic.GammaP
Type:	Java Class - 
Import:	from herschel.ia.numeric.toolbox.basic import GammaP

Description

Computes the incomplete Gamma function $P(a,x)$.

Example

Example 1: Apply GammaP on a Double1d

```
a = 0.5
x = Double1d([0.0316228,0.0707107,5.0,1.04881,2.44949,25.4951])
gp = GammaP(a)(x)
print gp # [0.19856221732013887,0.2931279825202761,0.9984345977771615,
          0.8524713739638958,0.9731274396553844,0.999999999990717]
]
```

API Summary

Jython Syntax

```
<y>=GammaP(a)(<x>)
```

Properties

[a double value a](#) [INPUT, MANDATORY, default=.]

[a double array or number x](#) [INPUT, MANDATORY, default=.]

[returns an double or float array \(or a number y](#) [OUTPUT, MANDATORY, default=no default value]

API details

Properties

a double value a [INPUT, MANDATORY, default=.]

a double array or number x [INPUT, MANDATORY, default=.]


returns an double or float array (or a number y [OUTPUT, MANDATORY, default=no default value]

if the input is a number) where each element is the incomplete Gamma function of the corresponding element of the input array.

See also

- [GammaQ](#)
- [GAMMALN](#)

2.155. GammaQ

Full Name:	herschel.ia.numeric.toolbox.basic.GammaQ
Type:	Java Class - 
Import:	from herschel.ia.numeric.toolbox.basic import GammaQ

Description

Computes the complement of incomplete Gamma function $Q(a,x)$.

Example

Example 1: Apply GammaQ on a Double1d

```
a = 0.5
x = Double1d([0.5,0.0316228,0.0707107,5.0,1.04881,2.44949,25.4951])
gq = GammaQ(a)(x)
print gq # [0.31731050863888255,0.8014377826798611,0.7068720174797238,
0.001565402222838555,0.14752862603610417,0.026872560344615565,9.282826956361127E-13]
```

API Summary

Jython Syntax

```
<y>=GammaQ(a)(<x>)
```

Properties

[a double value a](#) [INPUT, MANDATORY, default=.]

[a double array or number x](#) [INPUT, MANDATORY, default=.]

[returns an array \(or a number if the input is a number\) y](#) [OUTPUT, MANDATORY, default=no default value]

API details

Properties

a double value a [INPUT, MANDATORY, default=.]

a double array or number x [INPUT, MANDATORY, default=.]


returns an array (or a number if the input is a number) y [OUTPUT, MANDATORY, default=no default value]

where each element is the complement of the incomplete Gamma function of the corresponding element of the input array.

See also

- [GammaP](#)
- [GAMMALN](#)

2.156. Gauss2DModel

Full Name:	herschel.ia.numeric.toolbox.fit.Gauss2DModel
Type:	Java Class - 
Import:	from herschel.ia.numeric.toolbox.fit import Gauss2DModel

Description

Rotationally symmetric two dimensional Gaussian Model.

The model has 4 parameters yielding a "round" 2d-Gaussian.

$$f(x,y;p) = p_0 * \exp(-0.5 * ((x - p_1) / p_3)^2) * \exp(-0.5 * ((y - p_2) / p_3)^2)$$

p_0 = amplitude p_1 = x-shift p_2 = y-shift p_3 = sigma

The parameters are initialized at $\{1.0/\text{Math.sqrt}(\text{Math.PI}), 0.0, 0.0, 1.0\}$, normalised to 1.0

See [example](#)

Example


Example 1: Gauss2DModel

```
gauss = Gauss2DModel()
print gauss
print gauss.getNumberOfParameters()
print gauss.getDimensions()
# ... fitter etc. see LevenbergMarquardtFitter
<p>
```

See also

- [an asymmetric, rotated Gaussian2D model](#)
- [Gauss2DRotModel](#).

2.157. Gauss2DRotModel

Full Name:	herschel.ia.numeric.toolbox.fit.Gauss2DRotModel
Type:	Java Class - 
Import:	from herschel.ia.numeric.toolbox.fit import Gauss2DRotModel

Description

Rotated asymmetric 2 dimensional Gaussian Model.

The model has 6 parameters yielding a arbitrarily oriented elliptical 2d-Gaussian.

$$f(x,y;p) = p_0 * \exp(-0.5 * ((aa * cc + bb * ss) * xx + (aa * ss + bb * cc) * yy + 2 * x * y * c * s * (bb - aa)))$$

where

- $x = x - p_1$
- $y = y - p_2$
- $a = 1 / p_3 = 1 / \#_x$
- $b = 1 / p_4 = 1 / \#_y$
- $c = \cos(p_5) = \text{cosine of rotational angle}$
- $s = \sin(p_5) = \text{sine of rotational angle}$
- $aa = a * a$. Etc.

The parameters are resp. amplitude, x-shift, y-shift, x-width, y-width and rotational angle. They are initialized at {1.0/PI, 0.0, 0.0, 1.0, 2.0, 0.0}.


Do **not** initialize both widths at the same value as the model then degenerates: the rotational angle does not have a direction anymore.

See [example](#)

Example

Example 1: Gauss2DRotModel
<pre>gauss = Gauss2DRotModel() print gauss print gauss.getNumberOfParameters() # ... fitter etc. see LevenbergMarquardtFitter</pre>

2.158. GaussFitTask

Full Name:	herschel.ia.toolbox.fit.GaussFitTask
Alias:	GaussFitTask
Type:	Java Task - 
Import:	from herschel.ia.toolbox.fit import GaussFitTask
Category:	generic task

Description

generic module: GaussFitTask

Task shell around the package `herschel.ia.numeric.toolbox.fit`. Both a command line and a GUI interface are provided. With the GUI interface there is more of the Fitter package accessible than on this tasks command line. However not everything possible within the fitter package is accessible via this task, GUI or otherwise.

Most text fields in the GUI accept either numerics or JIDE variables or JIDE expressions.

Deprecated Use SourceFitTask in stead.

In SourceFitTask 4 output parameters have disappeared: `stdev`, `yfit`, `yband` and `evidence`.

At the same time they reappeared as get-methods of the task: `getStdev()`, `getYfit()`, `getYband()` and `getEvidence()`.

Because of the jython-stub convention the calls within HIPE (JIDE) stay the same. Parameters can be addressed in HIPE as `taskname.item` while get-methods can be addressed in the same way: `taskname.item`. Because of this fortuitous coincidence HIPE user will not be affected by this change.

Java developers do need to change their code.

Example

Example 1: GaussFitTask

```
from herschel.hifi.generic.task import GaussFitTask
# Assume that `tt` and/or `data` are DoubleIeld's
# usage on command line:
ft = GaussFitTask()( x=tt, y=data -)
ft.fitter = -"LevenbergMarquardtFitter"
ft() # performs the execute method
print ft.result # parameters of the fit
print ft.stdev # standard deviations
print ft.fitter # name of the fitter
# Etcetera, etc.
# use the GUI.
ft = GaussFitTask()
ft.gui = 1 # start the GUI
# from the GUI select the data, model, fitter, properties etc and run.
print ft.result # parameters of the fit
print ft.stdev # standard deviations
print ft.fitter # name of the fitter
# Etcetera as before.
```

API Summary

Jython Syntax

see example below

Properties
NumericData x [INPUT, OPTIONAL, default=null]
DoubleArray y [INPUT, MANDATORY, default=no]
Boolean indexview [INPUT, OPTIONAL, default=true]
DoubleArray weights [INPUT, OPTIONAL, default=null]
AbstractModel model [INPUT, OPTIONAL, default=null]
String modelname [INPUT, OPTIONAL, default=no]
ArrayIdData modelarg [INPUT, OPTIONAL, default=null]
Fitter fitter [INPUT, OPTIONAL, default=null]
String fittername [INPUT, true, default=no]
String autoinit [INPUT, OPTIONAL, default=""]
Integer smooth [INPUT, OPTIONAL, default=1]
DoubleId result [OUTPUT, OPTIONAL, default=n/a]
DoubleId parameters [OUTPUT, OPTIONAL, default=n/a]
DoubleId stdev [OUTPUT, OPTIONAL, default=n/a]
Double chisq [OUTPUT, OPTIONAL, default=n/a]
DoubleArray yfit [OUTPUT, OPTIONAL, default=n/a]
DoubleArray yband [OUTPUT, OPTIONAL, default=n/a]
DoubleId prior [INPUT, OPTIONAL, default=null]
Double evidence [OUTPUT, OPTIONAL, default=n/a]
Double scale [OUTPUT, OPTIONAL, default=n/a]
Boolean auto [INPUT, OPTIONAL, default=false]
Double fixed [INPUT, OPTIONAL, default=1.0]
Double mixed [INPUT, OPTIONAL, default=0.0]
Double tolerance [INPUT, OPTIONAL, default=0.01]
Integer iterations [INPUT, OPTIONAL, default=10000]
Double temperature [INPUT, OPTIONAL, default=0.0]
Double cooling [INPUT, OPTIONAL, default=0.95]
Integer tempsteps [INPUT, OPTIONAL, default=100]
DoubleId initialpars [INPUT, OPTIONAL, default=from model]
DoubleId highlimits [INPUT, OPTIONAL, default=null]
DoubleId lowlimits [INPUT, OPTIONAL, default=null]
IntId keepfixed [INPUT, OPTIONAL, default=null]
DoubleId fixedvalues [INPUT, OPTIONAL, default=null]
Boolean gui [INPUT, OPTIONAL, default=false]

Limitations

Not everything which is possible using the package directly is accessible via this task.

Miscellaneous

In case of problems look at the [trouble shooting](#) section.

API details

Properties

NumericData x [INPUT, OPTIONAL, default=null]
The independent variable(s) of the fit problem. If not provided, it will use indices ranging from 0 to len(y). If y is more-dimensional the proper 2-d indices are provided.
DoubleArray y [INPUT, MANDATORY, default=no]
The data to be fitted. It can be more-dimensional. E.g. a 2-dimensional map.
Boolean indexview [INPUT, OPTIONAL, default=true]
Applicable parameters appear in the same order as in the y DoubleArray This parameter is only active when y has 2 dimensions or more.
DoubleArray weights [INPUT, OPTIONAL, default=null]
The weights to be used in the fit. Can be more-dimensional. When provided it should be the same size as parameter y.
AbstractModel model [INPUT, OPTIONAL, default=null]
The model to be fitted. Default: GaussModel (for 1d data) or Gauss2DModel (for 2d data)
String modelname [INPUT, OPTIONAL, default=no]
The model to be fitted. Default: GaussModel (for 1d data) or Gauss2DModel (for 2d data)
Array1dData modelarg [INPUT, OPTIONAL, default=null]
Arguments, if any, needed in the Constructor of the model.
Fitter fitter [INPUT, OPTIONAL, default=null]
Fitter to be used. Default: LevenbergMarquardtFitter
String fittername [INPUT, true, default=no]
Fitter to be used. Default: LevenbergMarquardtFitter
String autoinit [INPUT, OPTIONAL, default=""]
Automated search for initial params. options: "high", "low", "center"
Integer smooth [INPUT, OPTIONAL, default=1]
Smooth data with BoxCarFilter before auto-search. smooth > 1.
Double1d result [OUTPUT, OPTIONAL, default=n/a]
The parameters of the fit.
Double1d parameters [OUTPUT, OPTIONAL, default=n/a]
The parameters of the fit. (same as result)
Double1d stdev [OUTPUT, OPTIONAL, default=n/a]
The standard deviations pertaining to the parameters.

Double <code>chisq</code> [OUTPUT, OPTIONAL, default=n/a]
Chi-squared of the fit.
DoubleArray <code>yfit</code> [OUTPUT, OPTIONAL, default=n/a]
An array of the same size as the data, containing the fitted valued
DoubleArray <code>yband</code> [OUTPUT, OPTIONAL, default=n/a]
An array of the same size as the data, containing a 1-# MonteCarlo error at each point.
DoubleId <code>prior</code> [INPUT, OPTIONAL, default=null]
Prior ranges for the parameters, needed when the evidence is requested.
Double <code>evidence</code> [OUTPUT, OPTIONAL, default=n/a]
Relative evidence the model carries w.r.t. the data. It needs values for prior, as many as the number of parameters and one more if auto-scaling is set.
Double <code>scale</code> [OUTPUT, OPTIONAL, default=n/a]
Noise scale of the data (when auto or mixed is selected)
Boolean <code>auto</code> [INPUT, OPTIONAL, default=false]
Select automatic noise scaling.
Double <code>fixed</code> [INPUT, OPTIONAL, default=1.0]
Fixed noise scale.
Double <code>mixed</code> [INPUT, OPTIONAL, default=0.0]
Automatic noise scaling with a minimum.
Double <code>tolerance</code> [INPUT, OPTIONAL, default=0.01]
Stopping criterion for iterative fitting.
Integer <code>iterations</code> [INPUT, OPTIONAL, default=10000]
Maximum number of iterations.
Double <code>temperature</code> [INPUT, OPTIONAL, default=0.0]
Starting temperature in annealing amoeba fitting.
Double <code>cooling</code> [INPUT, OPTIONAL, default=0.95]
Cooling step in annealing amoeba fitting.
Integer <code>tempsteps</code> [INPUT, OPTIONAL, default=100]
Number of exploratory steps at each temperature.
DoubleId <code>initialpars</code> [INPUT, OPTIONAL, default=from model]
Initial parameters in iterative fitters.
DoubleId <code>highlimits</code> [INPUT, OPTIONAL, default=null]
Upper limits of the parameters (only in AmoebaFitter)

<code>DoubleId lowlimits [INPUT, OPTIONAL, default=null]</code>

Lower limits of the parameters (only in AmoebaFitter)

<code>IntId keepfixed [INPUT, OPTIONAL, default=null]</code>
--

Keep these parameters fixed. (Parameter numbering starts at 0)
--

<code>DoubleId fixedvalues [INPUT, OPTIONAL, default=null]</code>

Values at which the parameters should be kept.
--


<code>Boolean gui [INPUT, OPTIONAL, default=false]</code>

Allows to handle this task using a gui.

History

- 15-10-2006 DK

2.159. GaussianFilter

Full Name:	herschel.ia.numeric.toolbox.filter.GaussianFilter
Alias:	GaussianFilter
Type:	Java Class - 
Import:	from herschel.ia.numeric.toolbox.filter import GaussianFilter

Description

Creates a Gaussian filter, that can be applied to numeric arrays of rank 1.

Example

Example 1: Apply GaussianFilter on a Int1d
<pre>x=Int1d([1,3,2,4,6,5]) f=GaussianFilter(0.2) print f(x) # [0.10650697891920073,1.1065069789192006, 2.6804790632423976, 2.319520936757602, 3.9999999999999996, 5.680479063242397]</pre>

API Summary

Jython Syntax
<pre><f>=GaussianFilter(<sigma> [, <center>=true false] [, <edge>=Convolution.ZEROES CIRCULAR REPEAT]) <x>=<f>(<x>)</pre>
Properties
real sigma [INPUT, MANDATORY, default=no default value]
boolean center [INPUT, NOT MANDATORY, default=false]
Convolution.ZEROES CIRCULAR REPEAT center [INPUT, NOT MANDATORY, default=Convolution.ZEROES]

API details

Properties

real sigma [INPUT, MANDATORY, default=no default value]
It must be a real.
boolean center [INPUT, NOT_MANDATORY, default=false]
Set center to true or false.
Convolution.ZEROES CIRCULAR REPEAT center [INPUT, NOT_MANDATORY, default=Convolution.ZEROES]
Set edge to true or false
<ul style="list-style-type: none"> • <code>edge=Convolution.ZEROES</code>: Set result to zero at edges.


```
Convolution.ZEROES|CIRCULAR|REPEAT center [INPUT, NOT_MANDATORY,  
default=Convolution.ZEROES]
```

- edge=Convolution.CIRCULAR: Wrap around at edges (circular convolution).
- edge=Convolution.REPEAT: Repeat the edge values of input array.

See also

- [BoxCarFilter](#)
- [Convolution](#)

2.160. GaussianSmoothingTask

Full Name:	herschel.ia.toolbox.image.GaussianSmoothingTask
Type:	Java Task - 
Import:	from herschel.ia.toolbox.image import GaussianSmoothingTask

Description

A Task to smooth an image using a convolution with a gaussian.

A Task to smooth an image by convolving it with a gaussian function.

API Summary


Properties
Image image [INPUT, MANDATORY, default=No default value]
Double sigma [INPUT, MANDATORY, default=Default value : Double(3.0)]
Image smoothed [OUTPUT, MANDATORY, default=No default value]

API details

Properties

Image image [INPUT, MANDATORY, default=No default value]
The image before smoothing.
Double sigma [INPUT, MANDATORY, default=Default value : Double(3.0)]
The standard deviation of the gaussian.
Image smoothed [OUTPUT, MANDATORY, default=No default value]
The image after smoothing.

2.161. GaussModel

Full Name:	herschel.ia.numeric.toolbox.fit.GaussModel
Type:	Java Class - 
Import:	from herschel.ia.numeric.toolbox.fit import GaussModel

Description

Gaussian Model.

$$f(x;p) = p_0 * \exp(-0.5 * ((x - p_1) / p_2)^2)$$

p_0 = amplitude p_1 = x-shift p_2 = sigma

The parameters are initialized at {1.0, 0.0, 1.0}.


See [example](#)

Example

Example 1: GaussModel

```
gauss = GaussModel()
print gauss
print gauss.getNumberOfParameters()
print gauss( DoubleId.range(11)-5 -)          # gauss between [-5,5]
# ... fitter etc. see LevenbergMarquardtFitter
```

2.162. getObservation

Full Name:	herschel.ia.toolbox.util.jython.getobs
Alias:	getObservation
Type:	Jython Task - 
Import:	from herschel.ia.toolbox.util.jython import getObservation
Category:	Input-Output

Description

Load an observation into your session.

This jython function will load the requested observation into your HIPE session. The pool and location of the data can be specified as optional arguments. By default, the function will look into a number of predefined locations, including the usual `~/ .hcss/lstore`. To configure this you can use the following configuration properties:

`hcss.knownPoolLocations` contains a list of directories with pools, these directories are searched in the order they appear in the list. If a directory name ends with an asterisk, all the sub-dirs of that directory are also searched for pools.

`hcss.dataPoolPrefix` the prefix of the pattern name of data pools, the final pattern has the form `<dataPoolPrefix><OD>_*` (where OD is the Operational Day)

`hcss.auxPoolPattern` pattern that matches the names of auxiliary pools

Note that you need to restart Hipe to change the value of these properties

Example

Example 1: Loading an observation in your HIPE session

```
obs = getObservation(1342180222L)
obs = getObservation(1342180231L, useHsa=True)
obs = getObservation(1342183656L, useHsa=True, instrument=PACS)
obs = getObservation(1342179542L, od=53)
obs = getObservation(1342180231L, poolLocation="/data/pools")
```

API Summary

Jython Syntax

```
obs = getObservation(obsid [, od=<OD Number>] [, useHsa=False|True] [, verbose=False|True] [, poolLocation=<directory>] [, poolName=<pool name>] [instrument=PACS|HIFI|SPIRE] [, creationDate=<date-time>] [, latest=True|False]
```

Properties

<code>long obsid</code> [INPUT, MANDATORY, default=no default]
<code>integer od</code> [INPUT, OPTIONAL, default=see description]
<code>boolean useHsa</code> [INPUT, OPTIONAL, default=False]
<code>boolean verbose</code> [INPUT, OPTIONAL, default=False]
<code>String poolName</code> [INPUT, OPTIONAL, default=see description]

Properties
<code>String poolLocation [INPUT, OPTIONAL, default=see description]</code>
<code>String instrument [INPUT, OPTIONAL, default=None]</code>
<code>boolean latest [INPUT, OPTIONAL, default=True]</code>

API details


Properties

<code>long obsid [INPUT, MANDATORY, default=no default]</code>
A long number representing the observation identifier. Append a 'L' character to prevent problems in conversion from int to long.
<code>integer od [INPUT, OPTIONAL, default=see description]</code>
An integer representing the operational day on which this observation was executed. This number is used for identifying MPE pools when access from within the 'MPE Grid'.
<code>boolean useHsa [INPUT, OPTIONAL, default=False]</code>
A flag to use the HSA as the source of all your products. Use this to load the observation context and all its associated products from the HSA. You will need to set your username/password correctly for this to work, preferably in a read protected property file, e.g. ~/hcss/user.props.
<code>boolean verbose [INPUT, OPTIONAL, default=False]</code>
A flag for more verbose output. Use this flag if you want to see which pools from which locations are used in the product storage that is queried for your observation.
<code>String poolName [INPUT, OPTIONAL, default=see description]</code>
The name of the pool to be used to load the requested observation
<code>String poolLocation [INPUT, OPTIONAL, default=see description]</code>
The directory where the pool with the requested observation is located
<code>String instrument [INPUT, OPTIONAL, default=None]</code>
The instrument for which this observation is valid, this is useful when asking for a parallel mode observation and you are only allowed access to one of the instruments data
<code>boolean latest [INPUT, OPTIONAL, default=True]</code>
When more then one observation context is found that satisfies the query criteria, the latest created Product will be returned

See also

- [???](#)

2.163. HAMMING

Full Name:	herschel.ia.numeric.toolbox.xform.Hamming
Alias:	HAMMING
Type:	Java Class - 
Import:	from herschel.ia.numeric.toolbox.xform import Hamming

Description

Hamming function for multiplying with input arrays.

This function performs an element-by-element multiplication of a input array with a Hamming function having the same number of elements. The Hamming function is defined by the following equation:

```
Hamming[i] = 0.54 + 0.46 * cos(2#( i-# -) -/ N)
where:
Hamming[i] is the value of the Hamming function at array index i, where 0#i\
#(N-1)
N is the length of the array for which the Hamming function is calculated.
# is the phase shift, defined as # = N/2.0
```

Input:

The input array should be a `Double1d` or `Float1d` and have length `N#3`.

Output:

The output array is a floating point array of length `N` that contains the element-by-element multiplication of the input array with the Hamming function.

Symmetry:

This equation produces a Hamming function with a unique point at index `i=0`. For even length arrays, the maximum amplitude is located at `N/2.0`. For odd length arrays, there are two amplitude maxima located at `ceiling(N/2.0)` and `floor(N/2.0)`.

Syntax:

To apply an element-by-element multiplication of the array `x` by the Hamming function, use any of the following options while coding in Jython:

1. `p = HAMMING(x) #Area normalized`
2. `p = HAMMING.AREA(x) #Area normalized, alternative syntax`
3. `q = HAMMING.AMPLITUDE(x) #Amplitude normalized`
4. `r = HAMMING.ENERGY(x) #Energy normalized`

Example

Example 1: Apply Hamming function to an input array x in java:

```
<pre>
Double1d x = new Double1d(100,1.0); -// create array of 100 1's
Double1d p = (Double1d)x.apply(Hamming.AREA); -// Area normalized
Double1d p2 = (Double1d)x.apply(Hamming.PROCEDURE); -//Area normalized,
alternative syntax
```

Example 1: Apply Hamming function to an input array x in java:

```

Double1d q = (Double1d)x.apply(Hamming.AMPLITUDE);  -//Amplitude normalized
Double1d r = (Double1d)x.apply(Hamming.ENERGY);    -//Energy normalized
</pre>
In Jython:


```

x = Double1d(100,1.0)
p = HAMMING(x) #Area normalized
p2 = HAMMING.AREA(x) #Area normalized, alternative syntax
q = HAMMING.AMPLITUDE(x) #Amplitude normalized
r = HAMMING.ENERGY(x) #Energy normalized
</pre>

```


```

API Summary

Jython Syntax

```
<y>=HAMMING.[<normalization> = AREA|AMPLITUDE|ENERGY] (<x>)
```

Properties

[Double1d or Float1d x \[INPUT, MANDATORY, default=No default value\]](#)

[Double1d or Float1d y \[OUTPUT, OPTIONAL, default=No default value\]](#)

[AMPLITUDE|ENERGY|AREA|PROCEDURE normalization \[INPUT, OPTIONAL, default=No default value\]](#)

Limitations

This function on operates on Double1d and Float1d arrays. It does not work for Complex1d arrays.

API details

Properties

Double1d or Float1d x [INPUT, MANDATORY, default=No default value]

The input array to be multiplied by the Hamming function.

Double1d or Float1d y [OUTPUT, OPTIONAL, default=No default value]

Outputs an array containing the element-by-element multiplication of the input array with the Hamming function.

AMPLITUDE|ENERGY|AREA|PROCEDURE normalization [INPUT, OPTIONAL, default=No default value]

The type of normalization of the Hamming function.

See also


- [HAMMING](#)

History

- 2007-Nov-23 (ZW) Added two new types of normalization.

- Changed the private constructor to take in one String parameter
- in order to have 3 types of normalization of the window functions.
- Changed some javadoc.
- 2008-Mar-07 (PK) Added static PROCEDURE member, modified n2 division and refactored.
- Used double to calculate n2 peak of the Hamming function thus affecting the symmetry of function.
- 2008-Mar-13 (PK) Changed to use a unique static member for each normalization.

2.164. HANNING

Full Name:	herschel.ia.numeric.toolbox.xform.Hanning
Alias:	HANNING
Type:	Java Class - 
Import:	from herschel.ia.numeric.toolbox.xform import Hanning

Description

Hanning function for multiplying with input arrays.

This function performs an element-by-element multiplication of a input array with a Hanning function having the same number of elements. The Hanning function is defined by the following equation:

```
Hanning[i] = 0.50 + 0.50 * cos(2#( i-# -) -/ N)
where:
Hanning[i] is the value of the Hanning function at array index i, where 0#i\
#(N-1)
N is the length of the array for which the Hanning function is calculated.
# is the phase shift, defined as # = N/2.0
```

Input:

The input array should be a `Double1d` or `Float1d` and have length `N#3`.

Output:

The output array is a floating point array of length `N` that contains the element-by-element multiplication of the input array with the Hanning function.

Symmetry:

This equation produces a Hanning function with a unique point at index `i=0`. For even length arrays, the maximum amplitude is located at `N/2.0`. For odd length arrays, there are two amplitude maxima located at `ceiling(N/2.0)` and `floor(N/2.0)`.

Syntax:

To apply an element-by-element multiplication of the array `x` by the Hanning function, use any of the following options while coding in Jython:

1. `p = HANNING(x) #Area normalized`
2. `p = HANNING.AREA(x) #Area normalized, alternative syntax`
3. `q = HANNING.AMPLITUDE(x) #Amplitude normalized`
4. `r = HANNING.ENERGY(x) #Energy normalized`

Example

Example 1: Apply Hanning function to an input array x in java:

```
<pre>
Double1d x = new Double1d(100,1.0); -// create array of 100 1's
Double1d p = (Double1d)x.apply(Hanning.AREA); -// Area normalized
Double1d p2 = (Double1d)x.apply(Hanning.PROCEDURE); -//Area normalized,
alternative syntax
```

Example 1: Apply Hanning function to an input array x in java:

```

Double1d q = (Double1d)x.apply(Hanning.AMPLITUDE);  -//Amplitude normalized
Double1d r = (Double1d)x.apply(Hanning.ENERGY);    -//Energy normalized
</pre>
In Jython:


```

x = Double1d(100,1.0)
p = HANNING(x) #Area normalized
p2 = HANNING.AREA(x) #Area normalized, alternative syntax
q = HANNING.AMPLITUDE(x) #Amplitude normalized
r = HANNING.ENERGY(x) #Energy normalized
</pre>

```


```

API Summary

Jython Syntax

```
<y>=HANNING.[<normalization> = AREA|AMPLITUDE|ENERGY] (<x>)
```

Properties

[Double1d or Float1d x \[INPUT, MANDATORY, default=No default value\]](#)

[Double1d or Float1d y \[OUTPUT, OPTIONAL, default=No default value\]](#)

[AMPLITUDE|ENERGY|AREA|PROCEDURE normalization \[INPUT, OPTIONAL, default=No default value\]](#)

Limitations

This function on operates on Double1d and Float1d arrays. It does not work for Complex1d arrays.

API details

Properties

Double1d or Float1d x [INPUT, MANDATORY, default=No default value]

The input array to be multiplied by the Hanning function.

Double1d or Float1d y [OUTPUT, OPTIONAL, default=No default value]

Outputs an array containing the element-by-element multiplication of the input array with the Hanning function.

AMPLITUDE|ENERGY|AREA|PROCEDURE normalization [INPUT, OPTIONAL, default=No default value]

The type of normalization of the Hanning function.

See also


- [HANNING](#)

History

- 2007-Nov-23 (ZW) Added two new types of normalization.

- Changed the private constructor to take in one String parameter
- in order to have 3 types of normalization of the window functions.
- Changed some javadoc.
- 2008-Mar-07 (PK) Added static PROCEDURE member, modified n2 division and refactored.
- Used double to calculate n2 peak of the Hanning function thus affecting the symmetry of function.
- 2008-Mar-13 (PK) Changed to use a unique static member for each normalization.

2.165. HarmonicModel

Full Name:	herschel.ia.numeric.toolbox.fit.HarmonicModel
Type:	Java Class - 
Import:	from herschel.ia.numeric.toolbox.fit import HarmonicModel

Description

Harmonic oscillator Model.


$$f(x;p) = \sum_j (p_k * \cos(2*\pi*j*x) + p_{k+1} * \sin(2*\pi*j*x)) \quad j = 1, N; k = 0, 2N, 2$$

All parameters are initialized at 1.0. It is a linear model.

Example

Example 1: HarmonicModel	
<pre>harm = HarmonicModel(3 -) # period = 1 print harm.getNumberOfParameters() # 6 harm = HarmonicModel(4, 2.7 -) # period = 2.7</pre>	

2.166. HduHeaders

Full Name:	herschel.ia.io.fits.HduHeaders
Alias:	HduHeaders
Type:	Java Class - 
Import:	from herschel.ia.io.fits import HduHeaders
Category:	class

Description

A class for working with FITS HDUs. HDU data is loaded only when requested.

Restrictions:

- Only HCSS FITS files (version 4) can be modified and saved.
- Avoid working with several datasets at the same time if you are planning to modify them.
- Only FITS files can be modified (fits from an InputStream cannot be updated).
- If the HDU is a compositeDataset, the children are returned when asking for the HDU dataset.
- If you modify a CompositeDataset, only its Metadata can be updated. If you want to add or remove a CompositeDataset child, you must work with it directly (add/remove CompositeDataset children directly).
- If you remove a CompositeDataset, its children are removed also.
- When modifying/updating/removing HDUs, a temporary file is created.

Examples

Example 1: default usage:

```
fa = FitsArchive();
#fa.reader = FitsArchive.STANDARD_READER #for reading non HCSS FITS files.
fh = fa.getFitsHduHeaders(path)
print fh
#get last dataset
ds = fh.headers[fh.numHeaders-1].dataset
print ds
#or
h = fh.headers[fh.numHeaders-1]
ds = h.dataset
print ds
```

Example 2: modify a metadata/fits header:

```
fa = FitsArchive();
fh = fa.getFitsHduHeaders(path)
print fh
#get last dataset
h = fh.headers[fh.numHeaders-1]
ds = h.dataset
print ds
ds.description = -'new description'
h.updateDataset(ds)
#or
fh.updateDataset(fh.numHeaders-1, ds)
```


Example 3: add a new dataset/HDU:

```
fa = FitsArchive();
fh = fa.getFitsHduHeaders(path)
ds = ArrayDataset(Int1d([1,2,3]))
fh.addDataset('new_dataset_id',ds)
```

Example 4: remove a HDU:

```
fa = FitsArchive();
fh = fa.getFitsHduHeaders(path)
#remove last header
fh.removeHeader(fh.numHeaders-1)
```

2.167. help

Full Name:	herschel.ia.toolbox.util.HelpTask
Alias:	help
Type:	Java Task - 
Import:	from herschel.ia.toolbox.util import HelpTask
Category:	task

Description

The help task for interactive help.

Users can access the help system by calling help from the command line. Help opens a pop up window displaying the specific help topic for the specified item or the main entry of documentation if no entry is specified or specific help can be found.

The current implementation displays the specific help only for tasks, Plot and Image.

Examples

Example 1: overview help

```
help()
```

Example 2: help on help

```
help(help)
```

Example 3: help on plot (1)

```
p = PlotXY
help(p)
```

Example 4: help on plot (2)

```
help("plotxy")
```

Example 5: help on display (1)

```
d = Display()
help(d)
```

Example 6: help on display (2)

```
help("display")
```

API Summary

Jython Syntax

```
help()  
help(item)
```

Property

Object item [INPUT, NO, default=help]

Limitations

The current implementation displays the specific help only for tasks, Plot and Image.

Miscellaneous

No miscellaneous

API details

Property

Object item [INPUT, NO, default=help]

The item to look for in the help. Currently supported: Task(s), Plot, Image (see examples)
--


See also

- [references](#)

History

- 2004-07-13 - NdC: first release.

2.168. Histogram

Full Name:	herschel.ia.numeric.toolbox.basic.Histogram
Alias:	Histogram
Type:	Java Class - 
Import:	from herschel.ia.numeric.toolbox.basic import Histogram

Description

Create a histogram on a Numeric data type, using a user-specified bin size.

Examples

Example 1: Apply Histogram on a Double1d

```
from herschel.ia.numeric import *
from herschel.ia.numeric.toolbox.random import RandomGauss
from herschel.ia.numeric.toolbox.basic import Histogram
from herschel.ia.numeric.toolbox.basic import BinCentres
hist=Histogram (1)
x=Double1d( [1,2,3,4] -)
print hist(x) [1,1,1,1]
```

Example 2: Plot histogram over bin centres

```
d = Double1d(200000,10)
d[99000:110000] = 15
d[99900:101000] = 20.0
rnd = RandomGauss()
for ix in range(200000):
    d[ix] += rnd.calc(1.0)
p = PlotXY (d)
binsize = STDDEV (d) * 2.35
print binsize
hist=Histogram(binsize)
bins=BinCentres(binsize)
p=PlotXY(bins(d),hist(d))
# or you could try: p = PlotXY (Histogram(binsize), BinCentres (binSize))
style=p.getStyle()
style.setChartType(Style.HISTOGRAM)
```

API Summary

Jython Syntax

```
<histogram>=Histogram(binSize)
<H>=histogram(<x>)
```

Properties

```
type binSize [INPUT, MANDATORY, default=no default value]
any array type x [INPUT, MANDATORY, default=no default value]
```

API details

Properties

<code>type binSize [INPUT, MANDATORY, default=no default value]</code>
--


Size of the histogram bins. See BinCentres for a description of the array of x-values for the centers of a histogram
--

<code>any array type x [INPUT, MANDATORY, default=no default value]</code>
--

See also

- [BinCentres](#)

2.169. Histogram

Full Name:	herschel.ia.toolbox.image.gui.Histogram
Type:	Java Class - 
Import:	from herschel.ia.toolbox.image.gui import Histogram

Description

An implementation of making a histogram. You can choose the kind of area

of which to make a histogram, give the cut levels and the number of bins, used for the histogram. When you push the plot-button, the histogram will be shown and the eventual region will be marked on the image.

API Summary

Constructors	
Histogram(ImageAnalysisToolbox toolbox)	The standard constructor for Histogram. This constructor creates a window
Histogram(boolean useAsComponent, ImageAnalysisToolbox toolbox)	Constructor for Histogram. When the new Histogram is
Methods	
PlotXY getPlot()	Returns the PlotXY shown in this Histogram.
JPanel getPanel()	Returns the panel of this Histogram.
int getUsedArea()	Returns the used type of area for this Histogram.
int getUsedRegion()	Returns the used type of region for this Histogram.
int getNumberOfEdges()	Returns the number of edges, if the region of interest is a
int getUsedCutLevels()	Returns the used type of cut levels for this Histogram.
double[] getCutlevels()	Returns the used cut levels (min, max) for this Histogram.
int getNumberOfBins()	Returns the used number of bins for this Histogram.
int getBinWidth()	Returns the width of the bins for this Histogram.
int getNbOfClicks()	Returns the number of valid clicks (i.e. in the image) you made.
boolean checkInput()	Returns true if the given input is correct; false otherwise. The

Methods
boolean checkArea() Checks whether the input area is correct. The error message is
boolean checkParameters() Checks whether the input parameters are correct. The error message
boolean checkCutLevels() Checks whether the input cut levels are correct. The error message
boolean checkNumberOfBins() Checks whether the input number of bins is correct. The error
ArrayList getImageFigures() Returns all ImageFigures used for this AreaHistogram (the eventual

API details

Constructors

Histogram(ImageAnalysisToolbox toolbox) The standard constructor for Histogram. This constructor creates a window in which you can make a histogram. In that window you give the input and retrieve the output. Argument ImageAnalysisToolbox toolbox [INPUT, MANDATORY]
Histogram(boolean useAsComponent, ImageAnalysisToolbox toolbox) Constructor for Histogram. When the new Histogram is component-based, a window for plotting the histogram is opened; otherwise nothing will be shown. Arguments boolean useAsComponent [INPUT, MANDATORY] ImageAnalysisToolbox toolbox [INPUT, MANDATORY]

Methods

PlotXY getPlot() Returns the PlotXY shown in this Histogram. Return PlotXY Returns the PlotXY shown in this Histogram.
JPanel getPanel() Returns the panel of this Histogram. Return JPanel Returns the panel of this Histogram.

```
int getUsedArea( )
```

Returns the used type of area for this Histogram.

Return

int

Returns the used type of area for this Histogram.

```
int getUsedRegion( )
```

Returns the used type of region for this Histogram.

Return

int

Returns the used type of region for this Histogram.

```
int getNumberOfEdges( )
```

Returns the number of edges, if the region of interest is a polygon; otherwise -1 is returned.

Return

int

Returns the number of edges, if the region of interest is a polygon; otherwise -1 is returned.

```
int getUsedCutLevels( )
```

Returns the used type of cut levels for this Histogram.

Return

int

Returns the used type of cut levels for this Histogram.

```
double[ ] getCutlevels( )
```

Returns the used cut levels (min, max) for this Histogram.

Return

double[]

Returns the used cut levels (min, max) for this Histogram.

```
int getNumberOfBins( )
```

Returns the used number of bins for this Histogram.

Return

int

Returns the used number of bins for this Histogram.

```
int getBinWidth( )
```

Returns the width of the bins for this Histogram.

Return

int getBinWidth()**int**

Returns the width of the bins for this Histogram.

int getNbOfClicks()

Returns the number of valid clicks (i.e. in the image) you made.

Return**int**

Returns the number of valid clicks (i.e. in the image) you made.

boolean checkInput()

Returns true if the given input is correct; false otherwise. The eventual appropriate error message is also constructed here.

Return**boolean**

Returns true if the given input is correct; false otherwise.

boolean checkArea()

Checks whether the input area is correct. The error message is adapted, when errors occur.

Return**boolean**

Returns true if the input area is correct; false otherwise.

boolean checkParameters()

Checks whether the input parameters are correct. The error message is adapted, when errors occur.

Return**boolean**

Returns true if the input parameters are correct; false otherwise.

boolean checkCutLevels()

Checks whether the input cut levels are correct. The error message is adapted, when errors occur.

Return**boolean**

Returns true if the input cut levels are correct; false otherwise.

boolean checkNumberOfBins()

Checks whether the input number of bins is correct. The error message is adapted, when errors occur.

Return

boolean

Returns true if the input number of bins is correct; false otherwise.

[ArrayList](#) getImageFigures()


Returns all ImageFigures used for this AreaHistogram (the eventual region of interest).

Return

[ArrayList](#)

Returns all ImageFigures used for this AreaHistogram (the eventual region of interest).

2.170. HistogramPanel

Full Name:	herschel.ia.toolbox.image.gui.HistogramPanel
Type:	Java Class - 
Import:	from herschel.ia.toolbox.image.gui import HistogramPanel

Description

A panel for the HistogramTask.

A panel to serve as GUI for the HistogramTask

API Summary

Constructor
HistogramPanel() The construction of a new HistogramPanel.
Methods
JPanel getImagePanel() Returns a display of the image.
JPanel getParameterPanel() Returns the parameter panel.
JPanel getButtonPanel() Returns the button panel.
Action getRunAction() Returns the run action for this HistogramPanel.
Action getResetAction() Returns the reset action for this HistogramPanel.
setSiteEventHandler(SiteEventHandler handler) Setting the site event handler.
setTask(TaskApi task) Associating the task.
setVariableSelection(VariableSelection selection) Setting the variable selection.
Display getDisplay() Returns the display for this HistogramPanel.
Image getImage() Returns the associated image.
TaskApi getTask() Returns the associated image.
Map getMap() Returns the associated map.
SiteEventHandler getHandler()

Methods
Returns the associated site event handler.
Double getInputLowCut() Returns the input low cut level.
getInputHighCut() Returns the input high cut level.
getInputNbOfBins() Returns the input number of bins.
PlotXY getHistogram() Returns the associated histogram.
setPoint(MouseEvent me) Storing the given mouse event in pixel coordinates.
setPoint(Double point) Storing the given screen coordinates in pixel coordinates.
updateFigure() Updates the associated figure.
ArrayList getClickedPoints() Returns the clicked points in pixel coordinates.
int getNbOfClickedPoints() Returns the number of times there was clicked before on the image.
increaseClicks() Increases the number of clicks made before on the image.
drawFigure() Draws the figure on the associated image.
trigger() Triggers the execution of the associated task.
updateHistogram() Updates the associated histogram.

API details

Constructor

<code>HistogramPanel()</code>
The construction of a new HistogramPanel.
The construction of a new HistogramPanel.

Methods

JPanel getImagePanel()
Returns a display of the image.
Returns a panel that displays the image for this HistogramPanel.
Return

[JPanel](#) getImagePanel()**[JPanel](#)**

Returns a panel that displays the image for this HistogramPanel.

[JPanel](#) getParameterPanel()

Returns the parameter panel.

Returns the parameter panel for this HistogramPanel.

Return**[JPanel](#)**

Returns the parameter panel for this HistogramPanel.

[JPanel](#) getButtonPanel()

Returns the button panel.

Returns the button panel for this HistogramPanel.

Return**[JPanel](#)**

Returns the button panel for this HistogramPanel.

[Action](#) getRunAction()

Returns the run action for this HistogramPanel.

Returns the run action for this HistogramPanel.

Return**[Action](#)**

Returns the run action for this HistogramPanel.

[Action](#) getResetAction()

Returns the reset action for this HistogramPanel.

Returns the reset action for this HistogramPanel.

Return**[Action](#)**

Returns the reset action for this HistogramPanel.

setSiteEventHandler(SiteEventHandler handler)

Setting the site event handler.

Sets the site event handler for this HistogramPanel to the given site event handler.

Argument

SiteEventHandler **handler** [INPUT, MANDATORY]

setTask(TaskApi task)

Associating the task.

Associates the given task with this HistogramPanel.

Argument

TaskApi **task** [INPUT, MANDATORY]

setVariableSelection(VariableSelection selection)

Setting the variable selection.

Sets the variable selection for this HistogramPanel to the given variable selection.

Argument

VariableSelection **selection** [INPUT, MANDATORY]

Display **getDisplay()**

Returns the display for this HistogramPanel.

Returns the display for this HistogramPanel.

Return

[Display](#)

Returns the display for this HistogramPanel.

Image **getImage()**

Returns the associated image.

Returns the image associated with this HistogramPanel.

Return

Image

Returns the image associated with this HistogramPanel.

TaskApi **getTask()**

Returns the associated image.

Returns the task associated with this HistogramPanel.

Return

TaskApi

Returns the task associated with this HistogramPanel.

Map **getMap()**

Returns the associated map.

Returns the map associated with this HistogramPanel.

Return

[Map](#)

Map <code>getMap()</code>
Returns the map associated with this HistogramPanel.
SiteEventHandler <code>getHandler()</code>
Returns the associated site event handler.
Returns the site event handler associated with this HistogramPanel.
Return
SiteEventHandler
Returns the site event handler associated with this HistogramPanel.
Double <code>getInputLowCut()</code>
Returns the input low cut level.
Returns the input low cut level for this HistogramPanel.
Return
Double
Returns the input low cut level for this HistogramPanel.
getInputHighCut()
Returns the input high cut level.
Returns the input high cut level for this HistogramPanel.
getInputNbOfBins()
Returns the input number of bins.
Returns the input number of bins for this HistogramPanel.
PlotXY <code>getHistogram()</code>
Returns the associated histogram.
Returns the histogram associated with this HistogramPanel.
Return
PlotXY
Returns the histogram associated with this HistogramPanel.
setPoint (MouseEvent <code>me</code>)
Storing the given mouse event in pixel coordinates.
Sets the position of the given mouse event at the appropriate place (i.e. at the index that corresponds with the number of times there was clicked before) in the list of clicked points for this HistogramPanel in pixel coordinates.
Argument
MouseEvent <code>me</code> [INPUT, MANDATORY]

setPoint(Double point)

Storing the given screen coordinates in pixel coordinates.

Sets the given screen coordinates at the appropriate place (i.e. at the index that corresponds with the number of times there was clicked before) in the list of clicked points for this HistogramPanel in pixel coordinates.

Argument

Double point [INPUT, MANDATORY]

updateFigure()

Updates the associated figure.

Updates the figure associated with this HistogramPanel.

ArrayList getClickedPoints()

Returns the clicked points in pixel coordinates.

Returns the clicked points for this HistogramPanel in pixel coordinates.

Return

ArrayList

Returns the clicked points for this HistogramPanel in pixel coordinates.

int getNbOfClickedPoints()

Returns the number of times there was clicked before on the image.

Returns the number of times there was clicked before on the image for this HistogramPanel.

Return

int

Returns the number of times there was clicked before on the image for this HistogramPanel.

increaseClicks()

Increases the number of clicks made before on the image.

Increases the number of clicks made before on the image for this HistogramPanel.

drawFigure()

Draws the figure on the associated image.

Draws the figure on the image associated with this HistogramPanel.

trigger()

Triggers the execution of the associated task.


Triggers the execution of the task associated with this HistogramPanel.

updateHistogram()

Updates the associated histogram.

updateHistogram()
Updates the histogram associated with this HistogramPanel.

2.171. HistogramTask

Full Name:	herschel.ia.toolbox.image.HistogramTask
Type:	Java Task - 
Import:	from herschel.ia.toolbox.image import HistogramTask

Description

An abstract Task to make histograms.

An abstract Task for making a histogram of an image as a whole or of a certain region of interest, which is bounded by a circle, an ellipse, a rectangle or a polygon. Its subclasses/subtasks are ImageHistogramTask, CircleHistogramTask, EllipseHistogramTask, RectangleHistogramTask and PolygonHistogramTask.

API Summary


Properties
Image image [INPUT, MANDATORY, default=No default value]
Double lowCut [INPUT, OPTIONAL, default=Default value : NaN]
Double highCut [INPUT, OPTIONAL, default=Default value : NaN]
Double bins [INPUT, MANDATORY, default=Default value : new Integer(2000)]

API details

Properties

Image image [INPUT, MANDATORY, default=No default value]
The image.
Double lowCut [INPUT, OPTIONAL, default=Default value : NaN]
The lower cut level.
Double highCut [INPUT, OPTIONAL, default=Default value : NaN]
The upper cut level.
Double bins [INPUT, MANDATORY, default=Default value : new Integer(2000)]
The number of bins.

2.172. HttpClientFactory

Full Name:	herschel.ia.pal.pool.http.HttpClientFactory
Type:	Java Class - 
Import:	from herschel.ia.pal.pool.http import HttpClientFactory
Category:	PAL

Description


Factory class to create HttpClientPool instances.

Example

Example 1: Create a HttpClientPool

```
params = java.util.HashMap()  
params.put("id", -"foo")  
params.put("url", -"http://localhost:8080/hcss/pal")  
storage=ProductStorage();  
storage.register(HttpClientFactory().createPool("http", params))
```

2.173. IFFT

Full Name:	herschel.ia.numeric.toolbox.xform.IFFT
Alias:	IFFT
Type:	Unknown (XML-based documentation) - 
Import:	from herschel.ia.numeric.toolbox.xform import IFFT

Description

Gives the inverse of the Fast Fourier Transform.

Example

Example 1: Apply IFF
<pre> from java.lang.Math import PI # Frequency modulated signal: parameters ts = 1E-6 # Sampling period (sec) fc = 200000 # Carrier frequency (Hz) fm = 2000 # Modulation frequency (Hz) beta = -.0003 # Modulation index (Hz) n = 5000 # Number of samples # Create signal in complex form t = DoubleId.range(n) * ts signal = SIN(2 * PI * fc * t * (1 + beta * COS(2 * PI * fm * t))) z_signal=ComplexId(signal) # Get spectrum spectrum = ABS(FFT(z_signal)) # Repeat with apodizing z_signal=ComplexId(HAMMING(signal)) spectrum2 = ABS(FFT(z_signal)) </pre>

API Summary

Jython Syntax
<y>=IFFT(<x>)
Properties
ComplexId x [INPUT , MANDATORY , default=no default value]
Array of float, double, complex. y [INPUT , MANDATORY , default=no default value]

Limitations

Does not work for ComplexId.

API details


Properties

ComplexId x [INPUT , MANDATORY , default=no default value]
ComplexId arrays only.
Array of float, double, complex. y [INPUT , MANDATORY , default=no default value]
Returns a float array for float input array, complex array for input complex array, and double array for any other numeric array type. The result is in radians.

See also

- [FET](#)

2.174. ImageAbsTask

Full Name:	herschel.ia.toolbox.image.ImageAbsTask
Type:	Java Task - 
Import:	from herschel.ia.toolbox.image import ImageAbsTask

Description

A Task that takes the absolute intensity values of an image.

A Task that replaces the intensity values in an image with their absolute value.

API Summary


Properties
Image image [INPUT, MANDATORY, default=No default value]
Image absolute [OUTPUT, MANDATORY, default=No default value]

API details

Properties

Image image [INPUT, MANDATORY, default=No default value]
The input image.
Image absolute [OUTPUT, MANDATORY, default=No default value]
The output image, being the image with the absolute values of the input image.

2.175. ImageAddTask

Full Name:	herschel.ia.toolbox.image.ImageAddTask
Type:	Java Task - 
Import:	from herschel.ia.toolbox.image import ImageAddTask

Description

A Task to add images.

A Task that allows to add two images pixel-to-pixel or co-add based on the Wcs coordinates, or to add a scalar to all intensity values in an image.

API Summary

Properties
Image image1 [INPUT, MANDATORY, default=No default value]
Image image2 [INPUT, OPTIONAL, default=No default value]
Integer ref [INPUT, OPTIONAL, default=No default value]
Number scalar [INPUT, OPTIONAL, default=No default value]
Image sum [OUTPUT, OPTIONAL, default=No default value]

API details

Properties

Image image1 [INPUT, MANDATORY, default=No default value]
The first image term/addend.
Image image2 [INPUT, OPTIONAL, default=No default value]
The second image term/addend.
Integer ref [INPUT, OPTIONAL, default=No default value]
The reference frame for the calculation of the sum.
Number scalar [INPUT, OPTIONAL, default=No default value]
The scalar term/addend.
Image sum [OUTPUT, OPTIONAL, default=No default value]
The sum.

2.176. ImageAddTaskSignatureComponent

Full Name:	herschel.ia.toolbox.image.gui.ImageAddTaskSignatureComponent
Type:	Java Class - 
Import:	from herschel.ia.toolbox.image.gui import ImageAddTaskSignatureComponent

Description

The task dialog for the ImageAddTask.

A task dialog to serve as GUI for the ImageAddTask.

API Summary

Constructor
ImageAddTaskSignatureComponent() The construction of a new ImageAddTaskSignatureComponent.
Methods
JLabel getFirstArgument() Returns the label for the first addend.
JLabel getSecondArgument() Returns the label for the second addend.

API details

Constructor

ImageAddTaskSignatureComponent()
The construction of a new ImageAddTaskSignatureComponent.
The construction of a new ImageAddTaskSignatureComponent.

Methods

JLabel getFirstArgument() Returns the label for the first addend. Returns the label for the first addend. Return JLabel Returns the label for the first addend.
JLabel getSecondArgument() Returns the label for the second addend. Returns the label for the second addend.


[JLabel](#) `getSecondArgument()`

Return

[JLabel](#)

Returns the label for the second addend.

2.177. ImageAnalysis

Full Name:	herschel.ia.toolbox.image.gui.ImageAnalysis
Type:	Java Class - 
Import:	from herschel.ia.toolbox.image.gui import ImageAnalysis

Description

A class to deal with ImageAnalysis (in the current version :

ProfilePlotting, AperturePhotometry, AreaHistograma, and ContourPlotting).

API Summary

Constructor
<p>ImageAnalysis(ImageAnalysisToolbox toolbox, String title)</p> <p>Standard constructor for ImageAnalysis. A new window with the</p>
Methods
<p>JFrame getFrame()</p> <p>Method that returns the JFrame of this ImageAnalysis.</p>
<p>ImageAnalysisToolbox getToolbox()</p> <p>Returns the ImageAnalysisToolbox, associated with this</p>
<p>Container getComponent()</p> <p>Returns the Content Pane of the JFrame of this</p>
<p>ArrayList getImageFigures()</p> <p>Returns all ImageFigures used for this ImageAnalysis.</p>

API details

Constructor

<code>ImageAnalysis(ImageAnalysisToolbox toolbox, String title)</code>
<p>Standard constructor for ImageAnalysis. A new window with the</p> <p>given title and associated with the given ImageAnalysisToolbox is created.</p>
<p>Arguments</p> <p>ImageAnalysisToolbox <code>toolbox</code> [INPUT, MANDATORY]</p> <p>String <code>title</code> [INPUT, MANDATORY]</p>

Methods

<code>JFrame getFrame()</code>
<p>Method that returns the JFrame of this ImageAnalysis.</p>
<p>Return</p> <p>JFrame</p>

[JFrame](#) getFrame()

Returns the JFrame of this ImageAnalysis.

[ImageAnalysisToolbox](#) getToolbox()

Returns the ImageAnalysisToolbox, associated with this

ImageAnalysis.

Return

[ImageAnalysisToolbox](#)

Returns the ImageAnalysisToolbox, associated with this ImageAnalysis.

[Container](#) getComponent()

Returns the Content Pane of the JFrame of this

ImageAnalysis.

Return

[Container](#)

Returns the Content Pane of the JFrame of this ImageAnalysis.

[ArrayList](#) getImageFigures()


Returns all ImageFigures used for this ImageAnalysis.

Return

[ArrayList](#)

Returns all ImageFigures used for this ImageAnalysis.

2.178. ImageAnalysisToolbox

Full Name:	herschel.ia.toolbox.image.gui.ImageAnalysisToolbox
Alias:	ImageAnalysisToolbox
Type:	Java Class - 
Import:	from herschel.ia.toolbox.image.gui import ImageAnalysisToolbox

Description

A class to perform image analysis.

In the current version you can do the following : 1) You can draw a 2D profile plot of a straight line, starting at a point you clicked on with the mouse, and ending at the current mouse position (in the image) or at the 2nd point you clicked on with the mouse. 2) You can perform aperture photometry on the image, either with an annular or a rectangular sky aperture, and with a chosen sky estimation algorithm. 3) You can make a histogram, either of the whole image, or of a selected region (bounded by a circle, an ellipse, a rectangle or a polygon). You can choose the cut levels and the number of bins for the histogram. 4) You can perform contour plotting on the image. You can give the contour values for which you wish the contours to be generated manually or let them be calculated (then you should specify the number of contour levels, the extreme contour values and the distribution of the contour levels). You can also increase/decrease the minimum length for contours to be drawn. In all cases the image analysis is performed on the shown layer and shown in a JTabbedPane on the right hand side.

Example

Example 1: A basic example on how to open a toolbox for image analysis on an

```
ImageDataset imds - :
iat = ImageAnalysisToolbox(imds)
or
iat = ImageAnalysisToolbox() iat.setImage(imds)
```

API Summary

Constructors
ImageAnalysisToolbox() The standard constructor for ImageAnalysisToolbox. This
ImageAnalysisToolbox(Image image) Constructor for ImageAnalysisToolbox. This constructor shows an
ImageAnalysisToolbox(Array2dData array2d) Constructor for ImageAnalysisToolbox. This constructor shows an
ImageAnalysisToolbox(Array3dData array3d) Constructor for ImageAnalysisToolbox. This constructor shows an
Methods
setLayer() Sets the image of this ImageAnalysisToolbox, using a file chooser.
addLayer() Sets the image of this ImageAnalysisToolbox, using a file chooser.
exit() Exits this ImageanalysisToolbox (closes (the window of) this

Methods	
setImage(Image image)	Sets the image you wish to analyze with this ImageAnalysisToolbox,
setImage(Array2dData array2d)	Sets the image you wish to analyse with this ImageAnalysisToolbox,
setImage(Array3dData array3d)	Sets the image you wish to analyze with this ImageAnalysisToolbox,
addImage(Image image)	Adds the given image to the image, analyzed with
addImage(Array2dData array2d)	Adds the given image to the images, analyzed with
addImage(Array3dData array3d)	Adds the given image to the images, analyzed with
closeActiveTab()	Closes the active tab, if there is one.
closeAllTabs()	Closes all tabs, if there are any.
setEnabled(boolean enabled)	Disables/enables all tabs on the tabbed pane and the menus for
boolean isEnabled()	Returns whether the tabbed pane and menus for closing tabs and
JFrame getFrame()	Method that returns the frame of this ImageAnalysisToolbox.
JPanel getPanel()	Method that returns the panel of this ImageAnalysisToolbox.
JTabbedPane getTabbedPane()	Method that returns the tabbed pane of this ImageAnalysisToolbox.
Image getImage()	Returns the displayed image.
Display getDisplay()	Returns the Display of the image you are analyzing with
int getSelectedLayer()	Returns the index of the shown layer, or the index of the selected
int getSelectedTab()	Returns the index of the tab, selected for the shown layer.
JTabbedPane getTabOnPane()	Returns the tab (tabbed pane) on the tabbed pane, that is
boolean isInImage(MouseEvent me)	Checks whether the given MouseEvent has occurred within the image
boolean isInImage(double pixX, double pixY)	Checks whether the point with given pixel coordinates (pixX, pixY)
boolean hasWCS()	

Methods
Returns true if sky coordinates are available for the image,
plotProfile2D()
Draws straight line on the image, starting at the point in the you
aperturePhotometry()
Performs aperture photometry on the image you are analyzing with
histogram()
Makes an area histogram. You can choose the kind of area you want
ArrayList getAllFigures()
Returns all ImageFigures used for this ImageAnalysisToolbox.
addFigures(ArrayList figures)
Adds the ImageFigures belonging to the selected tab up front to
updateImage()
Updates the image, when another tab is made active.
removeAllAnnotations()
Makes all annotations invisible.
createSpaceOnTabbedPane()
Creates a free space at the end of the ArrayList of ArrayLists of
createSpaceInTab()
Creates a free space at index 0 in the ArrayList of CanvasFigures,

API details

Constructors

ImageAnalysisToolbox()
The standard constructor for ImageAnalysisToolbox. This constructor shows an empty ImageAnalysisToolbox window in 2 parts, with only a title, buttons for maximizing, minimizing and closing the window, a menu bar (File, Image, Help) and a color bar and a status bar at the bottom.
ImageAnalysisToolbox(Image image)
Constructor for ImageAnalysisToolbox. This constructor shows an ImageAnalysisToolbox window in 2 parts, with a title, buttons for maximizing, minimizing and closing the window, a menu bar (file, Image, Help) and a color bar and a status bar at the bottom. Also the image, you wish to analyze is displayed.
Argument
Image image [INPUT, MANDATORY]
ImageAnalysisToolbox(Array2dData array2d)
Constructor for ImageAnalysisToolbox. This constructor shows an empty ImageAnalysisToolbox window in 2 parts, with a title, buttons for maximizing, minimizing and closing the window, and a menubar (file, Image, Help). Also the image you wish to analyse, is displayed.

ImageAnalysisToolbox(Array2dData array2d)
--

Argument

Array2dData array2d [INPUT, MANDATORY]

ImageAnalysisToolbox(Array3dData array3d)
--

Constructor for ImageAnalysisToolbox. This constructor shows an

empty ImageAnalysisToolbox window in 2 parts, with a title, buttons for maximizing, minimizing and closing the window, and a menu ar (file, Image, Help). Also the image you wish to analyze, is displayed.

Argument

Array3dData array3d [INPUT, MANDATORY]

Methods

setLayer()

Sets the image of this ImageAnalysisToolbox, using a file chooser.
--

addLayer()

Sets the image of this ImageAnalysisToolbox, using a file chooser.
--

exit()

Exits this ImageanalysisToolbox (closes (the window of) this ImageAnalysisToolbox).

setImage(Image image)

Sets the image you wish to analyze with this ImageAnalysisToolbox, to the given image.
--

Argument

Image image [INPUT, MANDATORY]

setImage(Array2dData array2d)

Sets the image you wish to analyze with this ImageAnalysisToolbox, to the given ImageDataset.

Argument

Array2dData array2d [INPUT, MANDATORY]

setImage(Array3dData array3d)

Sets the image you wish to analyze with this ImageAnalysisToolbox, to the given ImageDataset.

Argument

Array3dData array3d [INPUT, MANDATORY]

addImage(Image image)

Adds the given image to the image, analyzed with
--

addImage(Image image)
this ImageAnalysisToolbox.
Argument
Image image [INPUT, MANDATORY]
addImage(Array2dData array2d)
Adds the given image to the images, analyzed with
this ImageAnalysisToolbox.
Argument
Array2dData array2d [INPUT, MANDATORY]
addImage(Array3dData array3d)
Adds the given image to the images, analyzed with
this ImageAnalysisToolbox.
Argument
Array3dData array3d [INPUT, MANDATORY]
closeActiveTab()
Closes the active tab, if there is one.
closeAllTabs()
Closes all tabs, if there are any.
setEnabled(boolean enabled)
Disables/enables all tabs on the tabbed pane and the menus for
closing tabs and performing image analysis, of this ImageAnalysisToolbox.
Argument
boolean enabled [INPUT, MANDATORY]
boolean isEnabled()
Returns whether the tabbed pane and menus for closing tabs and
performing image analysis, of this ImageAnalysisToolbox are enabled/disabled.
Return
boolean
Returns true is the tabbed pane and menus for closing tabs and performing image analysis, of this ImageAnalysisToolbox are enabled; false otherwise.
JFrame getFrame()
Method that returns the frame of this ImageAnalysisToolbox.
Return
JFrame
The frame of this ImageAnalysisToolbox.

JPanel `getPanel()`

Method that returns the panel of this ImageAnalysisToolbox.

Return

[JPanel](#)

The panel of this ImageAnalysisToolbox.

JTabbedPane `getTabbedPane()`

Method that returns the tabbed pane of this ImageAnalysisToolbox.

Return

[JTabbedPane](#)

The tabbed pane of this ImageAnalysisToolbox.

Image `getImage()`

Returns the displayed image.

Return

Image

Returns the displayed image.

Display `getDisplay()`

Returns the Display of the image you are analyzing with this ImageAnalysisToolbox.

Return

[Display](#)

Returns the display of the image you are analyzing with this ImageAnalysisToolbox.

int `getSelectedLayer()`

Returns the index of the shown layer, or the index of the selected tab on the tabbed pane.

Return

int

Returns the index of the shown layer, or the index of the selected tab on the tabbed pane.

int `getSelectedTab()`

Returns the index of the tab, selected for the shown layer.

Return

int

The index of the tab, selected for the shown layer.

JTabbedPane `getTabOnPane()`

Returns the tab (tabbed pane) on the tabbed pane, that is

JTabbedPane getTabOnPane()
<p>selected.</p> <p>Return</p> <p>JTabbedPane</p> <p>Returns the tab (tabbed pane) on the tabbed pane, that is selected.</p>
boolean isInImage(MouseEvent me)
<p>Checks whether the given MouseEvent has occurred within the image we are analyzing with this ImageAnalysisToolbox.</p> <p>Argument</p> <p>MouseEvent me [INPUT, MANDATORY]</p> <p>Return</p> <p>boolean</p> <p>Returns true if the given MouseEvent has occurred within the image that is being analyzed with this ImageAnalysisToolbox; false otherwise.</p>
boolean isInImage(double pixX, double pixY)
<p>Checks whether the point with given pixel coordinates (pixX, pixY) lies in the image we are analyzing with this ImageAnalysisToolbox.</p> <p>Arguments</p> <p>double pixX [INPUT, MANDATORY]</p> <p>double pixY [INPUT, MANDATORY]</p> <p>Return</p> <p>boolean</p> <p>Returns true if the point with given pixel coordinates (pixX, pixY) lies in the image we are analyzing with this ImageAnalysisToolbox; false otherwise.</p>
boolean hasWCS()
<p>Returns true if sky coordinates are available for the image, analyzed with this ImageAnalysisToolbox; false otherwise.</p> <p>Return</p> <p>boolean</p> <p>Returns true if sky coordinates are available for the image, analyzed with this ImageAnalysisToolbox; false otherwise.</p>
plotProfile2D()
<p>Draws straight line on the image, starting at the point in the you clicked on with the mouse the 1st time, and ending at the current mouse position (in the image), or at the 2nd point (in the image) you clicked on, and plots the intensity along that straight line. When you click or move outside of the image, no straight line is drawn and no plot is shown.</p>

aperturePhotometry()

Performs aperture photometry on the image you are analyzing with

this ImageAnalysisToolbox, either with an annular sky aperture or a rectangular sky aperture, and with a chosen sky estimation algorithm.

histogram()

Makes an area histogram. You can choose the kind of area you want

to make the histogram of (the whole image, or a region bounded by a circle, an ellipse, a rectangle or a polygon), given the cut levels and the number of bins for the histogram.

[ArrayList](#) getAllFigures()

Returns all ImageFigures used for this ImageAnalysisToolbox.

Return

[ArrayList](#)

Returns all ImageFigures used for this ImageAnalysisToolbox.

addFigures([ArrayList](#) figures)

Adds the ImageFigures belonging to the selected tab up front to

the list of ImageFigures, used for this ImageAnalysisToolbox.

Argument

[ArrayList](#) figures [INPUT, MANDATORY]

updateImage()

Updates the image, when another tab is made active.

removeAllAnnotations()

Makes all annotations invisible.

createSpaceOnTabbedPane()

Creates a free space at the end of the ArrayList of ArrayLists of

ArrayLists of ImageFigures, that belong to the image analysis on the shown layer.

createSpaceInTab()

Creates a free space at index 0 in the ArrayList of CanvasFigures,


belonging to a new image analysis on the shown layer.

See also

- [???](#)
- [???](#)
- [???](#)

- [???](#)
- [???](#)
- [???](#)

2.179. ImageArithmeticsTask

Full Name:	herschel.ia.toolbox.image.ImageArithmeticsTask
Type:	Java Task - 
Import:	from herschel.ia.toolbox.image import ImageArithmeticsTask

Description

An abstract Task for image arithmetics.

An abstract Task for image arithmetics. The subclasses/subtasks are ImageAddTask (adding), ImageSubtractTask (subtracting), ImageMultiplyTask (multiply), ImageDivideTask (dividing) and ImageModuloTask (modulo).

API Summary


Properties
Image image1 [INPUT, MANDATORY, default=No default value]
Image image2 [INPUT, OPTIONAL, default=No default value]
Integer ref [INPUT, OPTIONAL, default=Default value : 0]
Number scalar [INPUT, OPTIONAL, default=No default value]

API details

Properties

Image image1 [INPUT, MANDATORY, default=No default value]
The input first image.
Image image2 [INPUT, OPTIONAL, default=No default value]
The input second image.
Integer ref [INPUT, OPTIONAL, default=Default value : 0]
The input reference frame for the calculation.
Number scalar [INPUT, OPTIONAL, default=No default value]
The input scalar.

2.180. ImageArithmeticsTaskSignatureComponent

Full Name:	herschel.ia.toolbox.image.gui.ImageArithmeticsTaskSignatureComponent
Type:	Java Class - 
Import:	from herschel.ia.toolbox.image.gui import ImageArithmeticsTaskSignatureComponent

Description

The task dialog for the ImageArithmeticsTask.

A task dialog to serve as GUI for the ImageArithmeticsTask.

API Summary

Constructor
ImageArithmeticsTaskSignatureComponent() The construction of a new ImageArithmeticsTaskSignatureComponent.
Methods
check() Validates the current modifications.
clear() Clears the current modifications.
getParameters() Returns a map of parameters and modifiers storing the values selected by the user.
setSignature(SignatureApi signature) Setting the signature.
setVariableSelection(VariableSelection selection) Setting the variable selection.
JLabel newLabel(String title, String tooltip) Returns a label with the given title and tooltip.
JComponent getComponent() Returns the component.
JLabel getFirstArgument() Returns the label for the first argument.
JLabel getSecondArgument() Returns the label for the second argument.

API details

Constructor

<code>ImageArithmeticsTaskSignatureComponent()</code>
The construction of a new ImageArithmeticsTaskSignatureComponent.

ImageArithmeticsTaskSignatureComponent()

The construction of a new ImageArithmeticsTaskSignatureComponent.

Methods**check()**

Validates the current modifications.

Validates the current modifications for this ImageArithmeticsTaskSignatureComponent.

clear()

Clears the current modifications.

Clears the current modifications for this ImageArithmeticsTaskSignatureComponent.

getParameters()

Returns a map of parameters and modifiers storing the values selected by the user.

Returns a map of parameters and modifiers storing the values selected by the user for this ImageArithmeticsTaskSignatureComponent.

setSignature(SignatureApi signature)

Setting the signature.

Sets the given signature as the signature for this ImageArithmeticsTaskSignatureComponent.

Argument

SignatureApi **signature** [INPUT, MANDATORY]

setVariableSelection(VariableSelection selection)

Setting the variable selection.

Sets the given selection as the variable selection for this ImageArithmeticsTaskSignatureComponent.

Argument

VariableSelection **selection** [INPUT, MANDATORY]

JLabel newLabel(String title, String tooltip)

Returns a label with the given title and tooltip.

Returns a label with the given title and tooltip.

Arguments

[String](#) **title** [INPUT, MANDATORY]

[String](#) **tooltip** [INPUT, MANDATORY]

Return

[JLabel](#)

Returns a label with the given title and tooltip.

[JComponent](#) getComponent()

Returns the component.

Returns the component for this ImageArithmeticsTaskSignatureComponent.

Return

[JComponent](#)

Returns the component for this ImageArithmeticsTaskSignatureComponent.

[JLabel](#) getFirstArgument()

Returns the label for the first argument.

Returns the label for the first argument for this ImageArithmeticsTaskSignatureComponent.

Return

[JLabel](#)

Returns the label for the first argument for this ImageArithmeticsTaskSignatureComponent.

[JLabel](#) getSecondArgument()

Returns the label for the second argument.


Returns the label for the second argument for this ImageArithmeticsTaskSignatureComponent.

Return

[JLabel](#)

Returns the label for the second argument for this ImageArithmeticsTaskSignatureComponent.

2.181. ImageAxis

Full Name:	herschel.ia.gui.image.ImageAxis
Type:	Java Class - 
Import:	from herschel.ia.gui.image import ImageAxis
Category:	Display

Description

Axes for image display.

This class can change the look of the axes of an image display. A grid can also be enabled.

API Summary

Methods
<code>disable()</code> Disables the axis
<code>enable()</code> Enables the axis
<code>setLabel(String newLabel)</code> Set the label of the axis
<code>String getLabel()</code> Returns the label
<code>setBackground(Color newBackground)</code> Sets the Background.
<code>Position getOrientation()</code> Returns the orientation of the axis
<code>setOrientation(Position orientation)</code> Sets the orientation of the axis
<code>setAxisStroke(int stroke)</code> Sets the axis stroke
<code>int getAxisStroke()</code> Returns the stroke of the axis.
<code>setTicksColor(Color color)</code> Sets the color of the ticks of the axis
<code>Color getTicksColor()</code> Returns the color of the ticks of the axis.
<code>setAxisColor(Color color)</code> Sets the color of the axis
<code>Color getAxisColor()</code> Returns the color of the axis.
<code>setLabelFont(Font labelFont)</code> Sets the font for the label.
<code>setLabelFont(int newSize)</code>

Methods
Sets the size of the font for the label.
Font getLabelFont() Returns the font for the label
Color getLabelFontColor() Returns the color of the label
setLabelFontColor(Color labelFontcolor) Sets the color for the label
showColorTable(boolean showColorTable) Shows or hides the color table for this axis.
getColorTable() Returns whether or not to show the color table for this axis.
setWorldCoordinates(boolean wcs) Show or hide the world coordinates
getWorldCoordinates() returns whether of not to show the world Coordinates.
setTickLabelFont(int newSize) Sets the size of the font for the ticks.
setTickLabelFont(Font tickLabelFont) Sets the font of the ticks
Font getTickLabelFont() Returns the font of the ticks.
Color getTickLabelFontColor() Returns the color of the font of the ticks
setTickLabelFontColor(Color color) Sets the color of the ticks font
setInnerTickLength(int length) Sets the length of the inner ticks
int getInnerTickLength() Returns the inner tick length
setOuterTickLength(int length) Sets the length of the outer ticks
int getOuterTickLength() Returns the outer tick length
setMainTicks(int ticks) Sets the number of main ticks
int getMainTicks() Returns the number of main ticks.
setMinorTicks(int ticks) Sets the number of minor ticks
int getMinorTicks() Returns the number of minor ticks.

Methods
setTickDistance(int pixels) Sets the tick distance
int getTickDistance() Returns the number of pixels between two ticks on the axis.
showGridLines(boolean gl) Enables or disables the grid
boolean getShowGridLines() Returns the status of the grid
setGridColor(Color gridColor) Sets the color of the grid
Color getGridColor() Returns the color of the grid
setDecimalDegrees(boolean b) Show the world coordinates in decimal degrees or not
getDecimalDegrees() returns whether of not to show the world Coordinates in decimal degrees.

API details

Methods

disable() Disables the axis Disables the axis
enable() Enables the axis Enables the axis
setLabel(String newLabel) Set the label of the axis The label of the axis is set. If no label is needed, an empty string should be given as label Argument String newLabel [INPUT, MANDATORY]
String getLabel() Returns the label Returns the label of the axis. Return String

String `getLabel()`

The label of the axis.

setBackground(Color newBackground)

Sets the Background.

Sets the Background.

Argument

Color `newBackground` [INPUT, MANDATORY]

Position `getOrientation()`

Returns the orientation of the axis

Returns the orientation of the axis

Return

Position

The orientation (as a Position)

setOrientation(Position orientation)

Sets the orientation of the axis

Changes the orientation of the axis.

Argument

Position `orientation` [INPUT, MANDATORY]

setAxisStroke(int stroke)

Sets the axis stroke

Sets the stroke (line width) of the axis (in pixels)

Argument

int `stroke` [INPUT, MANDATORY]

int `getAxisStroke()`

Returns the stroke of the axis.

Returns the stroke (line width) of the axis.

Return

int

The stroke of the axis.

setTicksColor(Color color)

Sets the color of the ticks of the axis

Sets the color of the ticks of the axis.

Argument

```
setTicksColor(Color color)
```

```
    Color color [INPUT, MANDATORY]
```

```
Color getTicksColor()
```

Returns the color of the ticks of the axis.

Returns the color of the ticks of the axis

Return

[Color](#)

Color The color of the ticks of the axis.

```
setAxisColor(Color color)
```

Sets the color of the axis

Sets the color of the axis.

Argument

```
    Color color [INPUT, MANDATORY]
```

```
Color getAxisColor()
```

Returns the color of the axis.

Returns the color of the axis

Return

[Color](#)

Color The color of the axis.

```
setLabelFont(Font labelFont)
```

Sets the font for the label.

Sets the font for the label.

Argument

```
    Font labelFont [INPUT, MANDATORY]
```

```
setLabelFont(int newSize)
```

Sets the size of the font for the label.

Sets the size of the font for the label.

Argument

```
    int newSize [INPUT, MANDATORY]
```

```
Font getLabelFont()
```

Returns the font for the label

Returns the font for the label of the axis.

Return

[Font](#)

Font <code>getLabelFont()</code>
The font for the label.
Color <code>getLabelFontColor()</code>
Returns the color of the label
Returns the color of the label.
Return
Color
the color of the label.
setLabelFontColor(Color labelFontcolor)
Sets the color for the label
Sets the color for the label
Argument
Color labelFontcolor [INPUT, MANDATORY]
showColorTable(boolean showColorTable)
Shows or hides the color table for this axis.
Shows (true) or hides (false) the color table for this axis.
Argument
boolean showColorTable [INPUT, MANDATORY]
getColorTable()
Returns whether or not to show the color table for this axis.
Returns whether or not to show the color table for this axis.
setWorldCoordinates(boolean wcs)
Show or hide the world coordinates
Shows the world coordinates (True) or the pixel coordinates (False)
Argument
boolean wcs [INPUT, MANDATORY]
getWorldCoordinates()
returns whether of not to show the world Coordinates.
returns whether of not to show the world Coordinates.
setTickLabelFont(int newSize)
Sets the size of the font for the ticks.
Sets the size of the font for the ticks.
Argument

```
setTickLabelFont(int newSize)
```

```
int newSize [INPUT, MANDATORY]
```

```
setTickLabelFont(Font tickLabelFont)
```

Sets the font of the ticks

Sets the font of the ticks

Argument

```
Font tickLabelFont [INPUT, MANDATORY]
```

```
Font getTickLabelFont()
```

Returns the font of the ticks.

Returns the font of the ticks.

Return

```
Font
```

The font of the ticks,

```
Color getTickLabelFontColor()
```

Returns the color of the font of the ticks

Returns the color of the font of the ticks

Return

```
Color
```

The color of the ticks font.

```
setTickLabelFontColor(Color color)
```

Sets the color of the ticks font

Sets the color of the ticks font

Argument

```
Color color [INPUT, MANDATORY]
```

```
setInnerTickLength(int length)
```

Sets the length of the inner ticks

Sets the distance that the ticks enter in the image

Argument

```
int length [INPUT, MANDATORY]
```

```
int getInnerTickLength()
```

Returns the inner tick length

Returns the distance that the ticks enter in the image.

Return

int getInnerTickLength()**int**

The distance that the ticks enter in the image.

setOuterTickLength(int length)

Sets the length of the outer ticks

Sets the distance that the ticks go out of the image

Argument**int length** [INPUT, MANDATORY]**int getOuterTickLength()**

Returns the outer tick length

Returns the distance that the ticks go out of the image.

Return**int**

The distance that the ticks go out of the image.

setMainTicks(int ticks)

Sets the number of main ticks

Sets the number of main ticks. Main ticks are labeled and are slightly longer than the other (minor) ticks. If the number of ticks is less than two, automatic selection of the number of ticks is enabled.

Argument**int ticks** [INPUT, MANDATORY]**int getMainTicks()**

Returns the number of main ticks.

Returns the number of main (labeled) ticks.

Return**int**

The number of main ticks.

setMinorTicks(int ticks)

Sets the number of minor ticks

Sets the number of minor ticks between 2 main ticks. Minor ticks are not labeled. Negative values will take the standard value of 4 minor ticks.

Argument**int ticks** [INPUT, MANDATORY]**int getMinorTicks()**

Returns the number of minor ticks.

int getMinorTicks()

Returns the number of minor (non-labeled) ticks, between two main ticks.

Return

int

The number of minor ticks between two main ticks.

setTickDistance(int pixels)

Sets the tick distance

Sets the distance (in pixels) between two (minor) ticks. Negative (or 0) values will enable automatic selection of the distance.

Argument

int pixels [INPUT, MANDATORY]

int getTickDistance()

Returns the number of pixels between two ticks on the axis.

Returns the number of pixels between two ticks on the axis. A negative value means the automatic selection of the tick distance is enabled.

Return

int

int The distance (in pixels) between two ticks.

showGridLines(boolean gl)

Enables or disables the grid

Shows or hides the grid for this axis. The grid can only be displayed if the magnification is at least 4!

Argument

boolean gl [INPUT, MANDATORY]

boolean getShowGridLines()

Returns the status of the grid

Returns true if the grid is shown.

Return

boolean

True if the grid is shown

setGridColor(Color gridColor)

Sets the color of the grid

Sets the color of the grid for this axis.

Argument

setGridColor([Color](#) gridColor)

[Color](#) gridColor [INPUT, MANDATORY]

[Color](#) getGridColor()

Returns the color of the grid

Returns the color of the grid

Return

[Color](#)

The color of the grid.

setDecimalDegrees(boolean b)

Show the world coordinates in decimal degrees or not

Shows the world coordinates in decimal degrees (True) or not

Argument


boolean **b** [INPUT, MANDATORY]

getDecimalDegrees()

returns whether of not to show the world Coordinates in decimal degrees.

returns whether of not to show the world Coordinates in decimal degrees.

2.182. ImageCeilTask

Full Name:	herschel.ia.toolbox.image.ImageCeilTask
Type:	Java Task - 
Import:	from herschel.ia.toolbox.image import ImageCeilTask

Description

A Task to ceil intensity values.

A Task that takes the ceiled intensity values of an image.

API Summary


Properties
Image image [INPUT, MANDATORY, default=No default value]
Image ceiled [OUTPUT, MANDATORY, default=No default value]

API details

Properties

Image image [INPUT, MANDATORY, default=No default value]
The input image.
Image ceiled [OUTPUT, MANDATORY, default=No default value]
The output ceiled image.

2.183. ImageContour

Full Name:	herschel.ia.dataset.image.ImageContour
Type:	Java Class - 
Import:	from herschel.ia.dataset.image import ImageContour

Description

A class to deal with ImageContours.

A class to deal with ImageContours.

API Summary

Constructors
ImageContour() This constructor creates a new ImageContour.
ImageContour(Image image) This constructor creates a new ImageContour, associated with the given image.
ImageContour(Image image, int nbOfContourValues) This constructor creates a new ImageContour, associated with the given image and with the given number of contour levels.
ImageContour(Image image, int nbOfContourLevels, double[] extremeContourValues, String distribution) This constructor creates a new ImageContour
Methods
addContourLevel(ContourLevel contourLevel, double contourValue) Adds the given ContourLevel for the given contour value to this ImageContour.
addContourLevel(ContourLevel contourLevel, String key) Adds the given ContourLevel to this ImageContour with the given key.
setWcs(Wcs wcs) Sets the Wcs for this ImageContour.
Wcs getWcs() Returns the Wcs for this ImageContour.
boolean hasWcs() Checks whether a Wcs is attached to this ImageContour.
boolean hasValidWcs() Checks whether a valid Wcs is attached to this ImageContour.

API details

Constructors

ImageContour()
This constructor creates a new ImageContour.

ImageContour()

This constructor creates a new ImageContour.

ImageContour(Image image)

This constructor creates a new ImageContour, associated with the given image.

This constructor creates a new ImageContour, associated with the given image.

Argument

Image **image** [INPUT, MANDATORY]

ImageContour(Image image, int nbOfContourValues)

This constructor creates a new ImageContour, associated with the given image and with the given number of contour levels.

This constructor creates a new ImageContour, associated with the given image and with the given number of contour levels.

Arguments

Image **image** [INPUT, MANDATORY]

int **nbOfContourValues** [INPUT, MANDATORY]

ImageContour(Image image, int nbOfContourLevels, double[] extremeContourValues, String distribution)

This constructor creates a new ImageContour

This constructor creates a new ImageContour, associated with the given image and with the given number of contour levels, the given extreme contour values and the given distribution of the contour levels.

Arguments

Image **image** [INPUT, MANDATORY]

int **nbOfContourLevels** [INPUT, MANDATORY]

double[] **extremeContourValues** [INPUT, MANDATORY]

[String](#) **distribution** [INPUT, MANDATORY]

Methods

addContourLevel([ContourLevel](#) contourLevel, double contourValue)

Adds the given ContourLevel for the given contour value to this ImageContour.

Adds the given ContourLevel for the given contour value to this ImageContour.

Arguments

[ContourLevel](#) **contourLevel** [INPUT, MANDATORY]

double **contourValue** [INPUT, MANDATORY]

addContourLevel([ContourLevel](#) contourLevel, [String](#) key)

Adds the given ContourLevel to this ImageContour with the given key.

Adds the given ContourLevel to this ImageContour with the given key.

Arguments

addContourLevel([ContourLevel](#) contourLevel, [String](#) key)

[ContourLevel](#) contourLevel [INPUT, MANDATORY]

[String](#) key [INPUT, MANDATORY]

setWcs([Wcs](#) wcs)

Sets the Wcs for this ImageContour.

Sets the Wcs for this ImageContour.

Argument

[Wcs](#) wcs [INPUT, MANDATORY]

[Wcs](#) getWcs()

Returns the Wcs for this ImageContour.

Returns the Wcs for this ImageContour.

Return

[Wcs](#)

Returns the Wcs for this ImageContour.

boolean hasWcs()

Checks whether a Wcs is attached to this ImageContour.

Returns true if a Wcs is attached to this ImageContour; false otherwise.

Return

boolean

Returns true if a Wcs is attached to this ImageContour; false otherwise.

boolean hasValidWcs()

Checks whether a valid Wcs is attached to this ImageContour.


Returns true if a valid Wcs is attached to this ImageContour; false otherwise.

Return

boolean

Returns true if a valid Wcs is attached to this ImageContour; false otherwise.

2.184. ImageConvolutionTask

Full Name:	herschel.ia.toolbox.image.ImageConvolutionTask
Type:	Java Task - 
Import:	from herschel.ia.toolbox.image import ImageConvolutionTask

Description

A Task to convolve an image with a convolution kernel.

A Task to convolve an image with a convolution kernel.

API Summary


Properties
Image image [INPUT, MANDATORY, default=No default value]
Image kernel [INPUT, MANDATORY, default=No default value.]
boolean center [INPUT, MANDATORY, default=Default value : true]
int edge [INPUT, MANDATORY, default=Default value : Convolution.REPEAT]
Image convolution [OUTPUT, MANDATORY, default=No default value]

API details

Properties

Image image [INPUT, MANDATORY, default=No default value]
The input image.
Image kernel [INPUT, MANDATORY, default=No default value.]
The input convolution kernel.
boolean center [INPUT, MANDATORY, default=Default value : true]
Whether to center the kernel to align the result with the data.
int edge [INPUT, MANDATORY, default=Default value : Convolution.REPEAT]
The behavior at the edges.
Image convolution [OUTPUT, MANDATORY, default=No default value]
The convolved image.

2.185. ImageConvolutionTaskSignatureComponent

Full Name:	herschel.ia.toolbox.image.gui.ImageConvolutionTaskSignatureComponent
Type:	Java Class - 
Import:	from herschel.ia.toolbox.image.gui import ImageConvolutionTaskSignatureComponent

Description

The task dialog for the ImageConvolutionTask.

A task dialog to serve as GUI for the ImageConvolutionTask.

API Summary

Constructor
ImageConvolutionTaskSignatureComponent() The construction of a new ImageConvolutionTaskSignatureComponent.
Methods
check() Validates the current modifications.
clear() Clears the current modifications.
Map getParameters() Returns a map of parameters and modifiers.
setSignature(SignatureApi signature) Setting the signature.
setVariableSelection(VariableSelection selection) Setting the variable selection.

API details

Constructor


<code>ImageConvolutionTaskSignatureComponent()</code>
The construction of a new ImageConvolutionTaskSignatureComponent.
The construction of a new ImageConvolutionTaskSignatureComponent.

Methods

<code>check()</code>
Validates the current modifications.
Validates the current modifications for this ImageConvolutionTaskSignatureComponent.

clear()
<p>Clears the current modifications.</p> <p>Clears the current modifications for this ImageConvolutionTaskSignatureComponent.</p>
Map getParameters()
<p>Returns a map of parameters and modifiers.</p> <p>Returns a map of parameters and modifiers storing the value selected by the user for this ImageConvolutionTaskSignatureComponent.</p> <p>Return</p> <p>Map</p> <p>Returns a map of parameters and modifiers storing the value selected by the user for this ImageConvolutionTaskSignatureComponent.</p>
setSignature(SignatureApi signature)
<p>Setting the signature.</p> <p>Sets the given signature as the signature for this ImageConvolutionTaskSignatureComponent.</p> <p>Argument</p> <p>SignatureApi signature [INPUT, MANDATORY]</p>
setVariableSelection(VariableSelection selection)
<p>Setting the variable selection.</p> <p>Sets the given selection as the variable selection for this ImageConvolutionTaskSignatureComponent.</p> <p>Argument</p> <p>VariableSelection selection [INPUT, MANDATORY]</p>

2.186. ImageDivideTask

Full Name:	herschel.ia.toolbox.image.ImageDivideTask
Type:	Java Task - 
Import:	from herschel.ia.toolbox.image import ImageDivideTask

Description

A Task to divide images.

A Task that allows to divide two images pixel-to-pixel or to divide based on the Wcs coordinates, or to divide all intensity values in an image by a scalar.

API Summary


Properties
Image image1 [INPUT, MANDATORY, default=No default value]
Integer ref [INPUT, OPTIONAL, default=No default value]
Number scalar [INPUT, OPTIONAL, default=No default value]
Image quotient [OUTPUT, MANDATORY, default=No default value]

API details

Properties

Image image1 [INPUT, MANDATORY, default=No default value]
The image dividend.
Integer ref [INPUT, OPTIONAL, default=No default value]
The reference frame for the calculation of the quotient.
Number scalar [INPUT, OPTIONAL, default=No default value]
The scalar divisor.
Image quotient [OUTPUT, MANDATORY, default=No default value]
The quotient.

2.187. ImageDivideTaskSignatureComponent

Full Name:	herschel.ia.toolbox.image.gui.ImageDivideTaskSignatureComponent
Type:	Java Class - 
Import:	from herschel.ia.toolbox.image.gui import ImageDivideTaskSignatureComponent

Description

The task dialog for the ImageDivideTask.

A task dialog to serve as GUI for the ImageDivideTask.

API Summary

Constructor
ImageDivideTaskSignatureComponent() The construction of a new ImageDivideTaskSignatureComponent.
Methods
JLabel getFirstArgument() Returns the label for the dividend.
JLabel getSecondArgument() Returns the label for the divisor.

API details

Constructor

ImageDivideTaskSignatureComponent()
The construction of a new ImageDivideTaskSignatureComponent.
The construction of a new ImageDivideTaskSignatureComponent.

Methods

JLabel getFirstArgument() Returns the label for the dividend. Returns the label for the dividend. Return JLabel Returns the label for the dividend.
JLabel getSecondArgument() Returns the label for the divisor. Returns the label for the divisor.


[JLabel](#) getSecondArgument()

Return

[JLabel](#)

Returns the label for the divisor.

2.188. ImageExp10Task

Full Name:	herschel.ia.toolbox.image.ImageExp10Task
Type:	Java Task - 
Import:	from herschel.ia.toolbox.image import ImageExp10Task

Description

exp10, OUTPUT, Image, MANDATORY, No default value

The output image, being the exp10 scaled image.

API Summary


Property
Image image [INPUT, MANDATORY, default=No default value]

API details

Property

Image image [INPUT, MANDATORY, default=No default value]
The input image.

2.189. ImageExpNTask

Full Name:	herschel.ia.toolbox.image.ImageExpNTask
Type:	Java Task - 
Import:	from herschel.ia.toolbox.image import ImageExpNTask

Description

A Task that allows to change the intensity values of an image according to an expN scaling.

API Summary


Properties
Image image [INPUT, MANDATORY, default=No default value]
Double n [INPUT, MANDATORY, default=Default value : 2.0]
Image expN [OUTPUT, MANDATORY, default=No default value]

API details

Properties

Image image [INPUT, MANDATORY, default=No default value]
The input image.
Double n [INPUT, MANDATORY, default=Default value : 2.0]
The input exponent.
Image expN [OUTPUT, MANDATORY, default=No default value]
The output image, being the expN scaled image.

2.190. ImageExpTask

Full Name:	herschel.ia.toolbox.image.ImageExpTask
Type:	Java Task - 
Import:	from herschel.ia.toolbox.image import ImageExpTask

Description

A Task that allows to change the intensity values of an image according to an exp scaling.

API Summary


Properties
Image image [INPUT, MANDATORY, default=No default value]
Image exp [OUTPUT, MANDATORY, default=No default value]

API details

Properties

Image image [INPUT, MANDATORY, default=No default value]
The input image.
Image exp [OUTPUT, MANDATORY, default=No default value]
The ouput image, being the exp scaled image.

2.191. ImageFloorTask

Full Name:	herschel.ia.toolbox.image.ImageFloorTask
Type:	Java Task - 
Import:	from herschel.ia.toolbox.image import ImageFloorTask

Description

A Task to floor intensity values.

A Task that takes the floored intensity values of an image.

API Summary

Properties
Image image [INPUT, MANDATORY, default=No default value]
Image floored [OUTPUT, MANDATORY, default=No default value]

API details

Properties

Image image [INPUT, MANDATORY, default=No default value]
The input image.
Image floored [OUTPUT, MANDATORY, default=No default value]
The output floored image.

2.192. ImageHistogramExplorer

Full Name:	herschel.ia.gui.image.ImageHistogramExplorer
Type:	Java Class - 
Import:	from herschel.ia.gui.image import ImageHistogramExplorer

Description

An explorer for ImageHistogramProducts.

An explorer for ImageHistogramProducts.

API Summary

Constructors
ImageHistogramExplorer() The construction of a new ImageHistogramExplorer.
ImageHistogramExplorer(Object object) The construction of a new ImageHistogramExplorer associated with the given object.
Methods
String getName() Returns the name.
String getDescription() Returns the description.
boolean canHandle(Class className) Checks whether this ImageHistogramExplorer can handle objects of the given class.
setObject(Object object) Sets the object.
ImageHistogramProduct getObject() Returns the object.
Class getVariableType() Returns the expected variable type.
JComponent getComponent() Returns the component that is responsible for displaying the data object.
JScrollPane getParameterTable() Returns the parameter table.
JComponent getHistogram() Returns the histogram.
addDataObjectListener(DataObjectListener listener) Adds the given listener.
removeDataObjectListener(DataObjectListener listener) Removes the given listener.

API details

Constructors

ImageHistogramExplorer()
The construction of a new ImageHistogramExplorer. The construction of a new ImageHistogramExplorer.
ImageHistogramExplorer(Object object)
The construction of a new ImageHistogramExplorer associated with the given object. The construction of a new ImageHistogramExplorer associated with the given object. Argument Object object [INPUT, MANDATORY]

Methods

String getName()
Returns the name. Returns the name for this ImageHistogramExplorer. Return String Returns the name for this ImageHistogramExplorer.
String getDescription()
Returns the description. Returns the description for this ImageHistogramExplorer. Return String Returns the description for this ImageHistogramExplorer.
boolean canHandle(Class className)
Checks whether this ImageHistogramExplorer can handle objects of the given class. Returns true if this ImageHistogramExplorer can handle objects of the given class; false otherwise. Argument Class className [INPUT, MANDATORY] Return boolean Returns true if this ImageHistogramExplorer can handle objects of the given class; false otherwise.

setObject(Object object)
Sets the object. Sets the object for this ImageHistogramExplorer to the given object. Argument Object object [INPUT, MANDATORY]
ImageHistogramProduct getObject()
Returns the object. Returns the object for this ImageHistogramExplorer. Return ImageHistogramProduct Returns the object for this ImageHistogramExplorer.
Class getVariableType()
Returns the expected variable type. Returns the expected variable type for this ImageHistogramExplorer. Return Class Returns the expected variable type for this ImageHistogramExplorer.
JComponent getComponent()
Returns the component that is responsible for displaying the data object. Returns the component that is responsible for displaying the data object for this ImageHistogramExplorer. Return JComponent Returns the component that is responsible for displaying the data object for this ImageHistogramExplorer.
JScrollPane getParameterTable()
Returns the parameter table. Returns the parameter table for this ImageHistogramExplorer. Return JScrollPane Returns the parameter table for this ImageHistogramExplorer.
JComponent getHistogram()
Returns the histogram.

JComponent `getHistogram()`

Returns the histogram for this ImageHistogramExplorer.

Return**JComponent**

Returns the histogram component for this ImageHistogramExplorer.

addDataObjectListener(DataObjectListener listener)

Adds the given listener.

Adds the given listener to this ImageHistogramExplorer to receive data object events from it.

Argument

DataObjectListener **listener** [INPUT, MANDATORY]

removeDataObjectListener(DataObjectListener listener)


Removes the given listener.

Removes the given listener for this ImageHistogramExplorer, so that it no longer receives data object events by this explorer.

Argument

DataObjectListener **listener** [INPUT, MANDATORY]

2.193. ImageHistogramProduct

Full Name:	herschel.ia.dataset.image.ImageHistogramProduct
Type:	Java Class - 
Import:	from herschel.ia.dataset.image import ImageHistogramProduct

Description

A class to deal with the results of an image histogram.

API Summary

Constructor
ImageHistogramProduct() The constructor of a new ImageHistogramProduct.
Methods
setCutLevels(double lowCut, double highCut) Sets the cut levels for this ImageHistogramProduct
setNbOfBins(int bins) Sets the number of bins for this
setHistogram(DoubleId values, DoubleId frequencies) Sets the histogram for this ImageHistogramProduct to a
setUnit(Unit unit) Sets the unit for this ImageHistogramProduct to the given unit.
double getLowCut() Returns the minimum cut level for this
double getHighCut() Returns the maximum cut level for this
int getNbOfBins() Returns the number of bins for this
TableDataset getHistogram() Returns the histogram for this ImageHistogramProduct.
DoubleId getValues() Returns the values for the histogram for this
DoubleId getFrequencies() Returns the frequencies for the histogram for this
String getUnit() Returns the unit for this ImageHistogramProduct.

API details

Constructor

ImageHistogramProduct()
The constructor of a new ImageHistogramProduct.

Methods

setCutLevels(double lowCut, double highCut)

Sets the cut levels for this ImageHistogramProduct to the given cut levels.

Arguments

double **lowCut** [INPUT, MANDATORY]
double **highCut** [INPUT, MANDATORY]

setNbOfBins(int bins)

Sets the number of bins for this ImageHistogramProduct to the given number of bins.

Argument

int **bins** [INPUT, MANDATORY]

setHistogram(DoubleId values, DoubleId frequencies)

Sets the histogram for this ImageHistogramProduct to a histogram with the given values and frequencies.

Arguments

[DoubleId](#) **values** [INPUT, MANDATORY]
[DoubleId](#) **frequencies** [INPUT, MANDATORY]

setUnit(Unit unit)

Sets the unit for this ImageHistogramProduct to the given unit.

Argument

Unit **unit** [INPUT, MANDATORY]

double getLowCut()

Returns the minimum cut level for this ImageHistogramProduct.

Return

double

Returns the minimum cut level for this ImageHistogramProduct.

double getHighCut()

Returns the maximum cut level for this ImageHistogramProduct.


Return

double

Returns the maximum cut level for this ImageHistogramProduct.

int <code>getNbOfBins()</code>
Returns the number of bins for this ImageHistogramProduct.
Return int Returns the number of bins for this ImageHistogramProduct.
TableDataset <code>getHistogram()</code>
Returns the histogram for this ImageHistogramProduct.
Return TableDataset Returns the histogram for this ImageHistogramProduct.
DoubleId <code>getValues()</code>
Returns the values for the histogram for this ImageHistogramProduct.
Return DoubleId Returns the values for the histogram for this
DoubleId <code>getFrequencies()</code>
Returns the frequencies for the histogram for this ImageHistogramProduct.
Return DoubleId Returns the frequencies for the histogram for this ImageHistogramProduct.
String <code>getUnit()</code>
Returns the unit for this ImageHistogramProduct.
Return String Returns the unit for this ImageHistogramProduct.

2.194. ImageHistogramTask

Full Name:	herschel.ia.toolbox.image.ImageHistogramTask
Type:	Java Task - 
Import:	from herschel.ia.toolbox.image import ImageHistogramTask

Description

A Task to make a histogram of an image.

A Task to make a histogram of an image as a whole.

API Summary

Properties
Image image [INPUT, MANDATORY, default=No default value]
Double lowCut [INPUT, OPTIONAL, default=Default value : NaN]
Double highCut [INPUT, OPTIONAL, default=Default value : NaN]
Integer bins [INPUT, MANDATORY, default=Default value : new Integer(2000)]
ImageHistogramProduct histogram [OUTPUT, MANDATORY, default=Default value : new ImageHistogramProduct()]

API details

Properties

Image image [INPUT, MANDATORY, default=No default value]
The image.
Double lowCut [INPUT, OPTIONAL, default=Default value : NaN]
The lower cut level.
Double highCut [INPUT, OPTIONAL, default=Default value : NaN]
The upper cut level.
Integer bins [INPUT, MANDATORY, default=Default value : new Integer(2000)]
The number of bins.
ImageHistogramProduct histogram [OUTPUT, MANDATORY, default=Default value : new ImageHistogramProduct()]
The histogram.

2.195. ImageHistogramTaskSignatureComponent

Full Name:	herschel.ia.toolbox.image.gui.ImageHistogramTaskSignatureComponent
Type:	Java Class - 
Import:	from herschel.ia.toolbox.image.gui import ImageHistogramTaskSignatureComponent

Description

The task dialog for the ImageHistogramTask.

A Task dialog to serve as GUI for the ImageHistogramTask.

API Summary

Constructor
ImageHistogramTaskSignatureComponent() The construction of a new ImageHistogramTaskSignatureComponent.
Methods
check() Validates the current modifications.
clear() Clears the current modifications.
Map getParameters() Returns a map of parameters and modifiers storing the value selected by the user.
setSignature(SignatureApi signature) Setting the signature.
setVariableSelection(VariableSelection selection) Setting the variable selection.
JLabel newLabel(String title, String tooltip) Returns a label with the given title and tooltip.
JComponent getComponent() Returns the component for this ImageHistogramTaskSignatureComponent.

API details

Constructor

<code>ImageHistogramTaskSignatureComponent()</code>
The construction of a new ImageHistogramTaskSignatureComponent.
The construction of a new ImageHistogramTaskSignatureComponent.

Methods

check()
Validates the current modifications.
Validates the current modifications for this ImageHistogramTaskSignatureComponent.
clear()
Clears the current modifications.
Clears the current modifications for this ImageHistogramTaskSignatureComponent.
Map getParameters()
Returns a map of parameters and modifiers storing the value selected by the user.
Returns a map of parameters and modifiers storing the value selected by the user for this ImageHistogramTaskSignatureComponent.
Return
Map
Returns a map of parameters and modifiers storing the value selected by the user for this ImageHistogramTaskSignatureComponent.
setSignature(SignatureApi signature)
Setting the signature.
Sets the given signature as the signature for this ImageHistogramTaskSignatureComponent.
Argument
SignatureApi signature [INPUT, MANDATORY]
setVariableSelection(VariableSelection selection)
Setting the variable selection.
Sets the given selection as the variable selection for this ImageHistogramTaskSignatureComponent.
Argument
VariableSelection selection [INPUT, MANDATORY]
JLabel newLabel(String title, String tooltip)
Returns a label with the given title and tooltip.
Returns a label with the given title and tooltip.
Arguments
String title [INPUT, MANDATORY]
String tooltip [INPUT, MANDATORY]
Return
JLabel

[JLabel](#) newLabel([String](#) title, [String](#) tooltip)

Returns a label with the given title and tooltip.

[JComponent](#) getComponent()

Returns the component for this ImageHistogramTaskSignatureComponent.


Returns the component for this ImageHistogramTaskSignatureComponent.

Return

[JComponent](#)

Returns the component for this ImageHistogramTaskSignatureComponent.

2.196. ImageLog10Task

Full Name:	herschel.ia.toolbox.image.ImageLog10Task
Type:	Java Task - 
Import:	from herschel.ia.toolbox.image import ImageLog10Task

Description

A Task that allows to change the intensity values of an image according to a log10 scaling.

A Task that allows to change the intensity values of an image according to a log10 scaling.

API Summary


Properties
Image image [INPUT, MANDATORY, default=No default value]
Image log10 [OUTPUT, MANDATORY, default=No default value]

API details

Properties

Image image [INPUT, MANDATORY, default=No default value]
The input image.
Image log10 [OUTPUT, MANDATORY, default=No default value]
The ouput image, being the log10 scaled image.

2.197. ImageLogNTask

Full Name:	herschel.ia.toolbox.image.ImageLogNTask
Type:	Java Task - 
Import:	from herschel.ia.toolbox.image import ImageLogNTask

Description

A Task that allows to change the intensity values of an image according to a logN scaling.

A Task that allows to change the intensity values of an image according to a logN scaling.

API Summary


Properties
Image.class image [INPUT, MANDATORY, default=No default value]
Double.class n [INPUT, MANDATORY, default=No default value]
Image.class logN [OUTPUT, MANDATORY, default=No default value]

API details

Properties

Image.class image [INPUT, MANDATORY, default=No default value]
The input image.
Double.class n [INPUT, MANDATORY, default=No default value]
The input n.
Image.class logN [OUTPUT, MANDATORY, default=No default value]
The output logN scaled image.

2.198. ImageLogTask

Full Name:	herschel.ia.toolbox.image.ImageLogTask
Type:	Java Task - 
Import:	from herschel.ia.toolbox.image import ImageLogTask

Description

A Task that allows to cahnge the intensity values of an image according to a log scaling.

API Summary


Properties
Image image [INPUT, MANDATORY, default=No default value]
Image log [OUTPUT, MANDATORY, default=No default value]

API details

Properties

Image image [INPUT, MANDATORY, default=No default value]
The input image.
Image log [OUTPUT, MANDATORY, default=No default value]
The ouput image, being the log scaled image.

2.199. ImageModuloTask

Full Name:	herschel.ia.toolbox.image.ImageModuloTask
Type:	Java Task - 
Import:	from herschel.ia.toolbox.image import ImageModuloTask

Description

A Task for modulo calculation with images.

A Task that allows to take the modulus of an image, either with another image or with a scalar.

API Summary

Properties
Image image1 [INPUT, MANDATORY, default=No default value]
Image image2 [INPUT, OPTIONAL, default=No default value]
Integer ref [INPUT, OPTIONAL, default=No default value]
Number scalar [INPUT, OPTIONAL, default=No default value]
Image remainder [OUTPUT, MANDATORY, default=No default value]

API details

Properties

Image image1 [INPUT, MANDATORY, default=No default value]
The image dividend.
Image image2 [INPUT, OPTIONAL, default=No default value]
The image divisor.
Integer ref [INPUT, OPTIONAL, default=No default value]
The reference frame for the calculation of the modulus.
Number scalar [INPUT, OPTIONAL, default=No default value]
The scalar divisor.
Image remainder [OUTPUT, MANDATORY, default=No default value]
The remainder after division.

2.200. ImageModuloTaskSignatureComponent

Full Name:	herschel.ia.toolbox.image.gui.ImageModuloTaskSignatureComponent
Type:	Java Class - 
Import:	from herschel.ia.toolbox.image.gui import ImageModuloTaskSignatureComponent

Description

The task dialog for the ImageModuloTask.

A task dialog to serve as GUI for the ImageModuloTask.

API Summary

Constructor
ImageModuloTaskSignatureComponent() The construction of a new ImageModuloTaskSignatureComponent.
Methods
JLabel getFirstArgument() Returns the label for the dividend.
JLabel getSecondArgument() Returns the label for the divisor.

API details

Constructor

<code>ImageModuloTaskSignatureComponent()</code>
The construction of a new ImageModuloTaskSignatureComponent.
The construction of a new ImageModuloTaskSignatureComponent.

Methods

<code>JLabel getFirstArgument()</code>
Returns the label for the dividend.
Returns the label for the dividend.
Return
JLabel Returns the label for the dividend.
<code>JLabel getSecondArgument()</code>
Returns the label for the divisor.

[JLabel](#) `getSecondArgument()`


Returns the label for the divisor.

Return

[JLabel](#)

Returns the label for the divisor.

2.201. ImageMultiplyTask

Full Name:	herschel.ia.toolbox.image.ImageMultiplyTask
Type:	Java Task - 
Import:	from herschel.ia.toolbox.image import ImageMultiplyTask

Description

A Task multiply images.

A Task that allows to multiply two images pixel-to-pixel or to multiply based on the Wcs coordinates, or to multiply all intensity values in an image with a scalar.

API Summary


Properties
Image image1 [INPUT, MANDATORY, default=No default value]
Image image2 [INPUT, OPTIONAL, default=No default value]
Integer ref [INPUT, OPTIONAL, default=No default value]
Number scalar [INPUT, OPTIONAL, default=No default value]
Image product [OUTPUT, MANDATORY, default=No default value]

API details

Properties

Image image1 [INPUT, MANDATORY, default=No default value]
The image multiplier.
Image image2 [INPUT, OPTIONAL, default=No default value]
The image multiplicand.
Integer ref [INPUT, OPTIONAL, default=No default value]
The reference frame for the calculation of the product.
Number scalar [INPUT, OPTIONAL, default=No default value]
The scalar multiplicand.
Image product [OUTPUT, MANDATORY, default=No default value]
The product.

2.202. ImageMultiplyTaskSignatureComponent

Full Name:	herschel.ia.toolbox.image.gui.ImageMultiplyTaskSignatureComponent
Type:	Java Class - 
Import:	from herschel.ia.toolbox.image.gui import ImageMultiplyTaskSignatureComponent

Description

The task dialog for the ImageMultiplyTask.

A task dialog to serve as GUI for the ImageMultiplyTask.

API Summary

Constructor
ImageMultiplyTaskSignatureComponent() The construction of a new ImageMultiplyTaskSignatureComponent.
Methods
JLabel getFirstArgument() Returns the label for the multiplier.
JLabel getSecondArgument() Returns the label for the multiplicand.

API details

Constructor

<code>ImageMultiplyTaskSignatureComponent()</code>
The construction of a new ImageMultiplyTaskSignatureComponent.
The construction of a new ImageMultiplyTaskSignatureComponent.

Methods

<code>JLabel getFirstArgument()</code>
Returns the label for the multiplier.
Returns the label for the multiplier.
Return
JLabel
Returns the label for the multiplier.
<code>JLabel getSecondArgument()</code>
Returns the label for the multiplicand.

[JLabel](#) `getSecondArgument()`


Returns the label for the multiplicand.

Return

[JLabel](#)

Returns the label for the multiplicand.

2.203. ImagePowerTask

Full Name:	herschel.ia.toolbox.image.ImagePowerTask
Type:	Java Task - 
Import:	from herschel.ia.toolbox.image import ImagePowerTask

Description

A Task that changes images according to a power scaling.

A Task that changes the intensity values of an image according to a power scaling.

API Summary


Properties
Image image [INPUT, MANDATORY, default=No default value]
Double n [INPUT, MANDATORY, default=Default value : 2.0]
Image powered [OUTPUT, MANDATORY, default=No default value]

API details

Properties

Image image [INPUT, MANDATORY, default=No default value]
The input image.
Double n [INPUT, MANDATORY, default=Default value : 2.0]
The input power.
Image powered [OUTPUT, MANDATORY, default=No default value]
The output image.

2.204. ImageRoundTask

Full Name:	herschel.ia.toolbox.image.ImageRoundTask
Type:	Java Task - 
Import:	from herschel.ia.toolbox.image import ImageRoundTask

Description

A Task to round intensity values.

A Task that takes the rounded intensity values of an image.

API Summary

Properties
Image image [INPUT, MANDATORY, default=No default value]
Image rounded [OUTPUT, MANDATORY, default=No default value]

API details

Properties

Image image [INPUT, MANDATORY, default=No default value]
The input image.
Image rounded [OUTPUT, MANDATORY, default=No default value]
The output rounded image.

2.205. ImageSaverTask

Full Name:	herschel.ia.toolbox.image.ImageSaverTask
Type:	Java Task - 
Import:	from herschel.ia.toolbox.image import ImageSaverTask
Category:	task/image

Description

A Task to save images.

A task which translates and creates a grey colors JPEG file from an image, not taking into account annotations or cut levels.

Example

Example 1: Saving a grey color image to a file

```
ImageSaverTask()(image = im, filename = -"test.jpg")
```

API Summary

Jython Syntax

```
ImageSaverTask()(image, filename)
```

Properties

[Image **image** \[INPUT, MANDATORY, default=No default value\]](#)

[String **filename** \[INPUT, MANDATORY, default=No default value\]](#)

API details

Properties


Image image [INPUT, MANDATORY, default=No default value]

The image to save.

String filename [INPUT, MANDATORY, default=No default value]

The filename for the save image.

2.206. ImageSqrtTask

Full Name:	herschel.ia.toolbox.image.ImageSqrtTask
Type:	Java Task - 
Import:	from herschel.ia.toolbox.image import ImageSqrtTask

Description

A Task that changes images according to a sqrt scaling.

A Task that changes the intensity values of an image according to a sqrt scaling.

API Summary


Properties
Image image [INPUT, MANDATORY, default=No default value]
Image sqrt [OUTPUT, MANDATORY, default=No default value]

API details

Properties

Image image [INPUT, MANDATORY, default=No default value]
The input image.
Image sqrt [OUTPUT, MANDATORY, default=No default value]
The output image, being the image changed according to a sqrt scaling.

2.207. ImageSquareTask

Full Name:	herschel.ia.toolbox.image.ImageSquareTask
Type:	Java Task - 
Import:	from herschel.ia.toolbox.image import ImageSquareTask

Description

A Task that changes images according to a square scaling.

A Task that allows to change the intensity values of an image according to a square scaling.

API Summary


Properties
Image image [INPUT, MANDATORY, default=No default value]
Image square [OUTPUT, MANDATORY, default=No default value]

API details

Properties

Image image [INPUT, MANDATORY, default=No default value]
The input image.
Image square [OUTPUT, MANDATORY, default=No default value]
The output image, being the squared image.

2.208. ImageSubtractTask

Full Name:	herschel.ia.toolbox.image.ImageSubtractTask
Type:	Java Task - 
Import:	from herschel.ia.toolbox.image import ImageSubtractTask

Description

A Task to subtract images.

A Task that allows to subtract two images pixel-to-pixel or based on the Wcs coordinates, or subtract a scalar from all intensity values of an image.

API Summary


Properties
Image image1 [INPUT, MANDATORY, default=No default value]
Image image2 [INPUT, OPTIONAL, default=No default value]
Integer ref [INPUT, OPTIONAL, default=No default value]
Number scalar [INPUT, OPTIONAL, default=No default value]
Image difference [OUTPUT, OPTIONAL, default=No default value]

API details

Properties

Image image1 [INPUT, MANDATORY, default=No default value]
The image minuend.
Image image2 [INPUT, OPTIONAL, default=No default value]
The image subtrahend.
Integer ref [INPUT, OPTIONAL, default=No default value]
The reference frame for the calculation of the difference.
Number scalar [INPUT, OPTIONAL, default=No default value]
The scalar subtrahend.
Image difference [OUTPUT, OPTIONAL, default=No default value]
The difference.

2.209. ImageSubtractTaskSignatureComponent

Full Name:	herschel.ia.toolbox.image.gui.ImageSubtractTaskSignatureComponent
Type:	Java Class - 
Import:	from herschel.ia.toolbox.image.gui import ImageSubtractTaskSignatureComponent

Description

The task dialog for the ImageSubtractTask.

A task dialog for the ImageSubtractTask.

API Summary

Constructor
ImageSubtractTaskSignatureComponent() The construction of a new ImageSubtractTaskSignatureComponent.
Methods
JLabel getFirstArgument() Returns the label for the minuend.
JLabel getSecondArgument() Returns the label for the subtrahend.

API details

Constructor

<code>ImageSubtractTaskSignatureComponent()</code>
The construction of a new ImageSubtractTaskSignatureComponent.
The construction of a new ImageSubtractTaskSignatureComponent.

Methods

JLabel getFirstArgument() Returns the label for the minuend. Returns the label for the minuend. Return JLabel Returns the label for the minuend.
JLabel getSecondArgument() Returns the label for the subtrahend.

[JLabel](#) `getSecondArgument()`


Returns the label for the subtrahend.

Return

[JLabel](#)

Returns the label for the subtrahend.

2.210. importCube

Full Name:	herschel.ia.toolbox.cube.ImportCubeTask
Alias:	importCube
Type:	Java Task - 
Import:	from herschel.ia.toolbox.cube import ImportCubeTask
Category:	task/cube

Description

The ImportCube task for Cubes.

A task which imports a Cube from a fits file

Example

Example 1: Import a fits file

```
importCube(simplecube = cube, filename = -"myFile.fits")
```

API Summary

Jython Syntax

```
importCube(cube, filename)
```

Properties

Cube **simplecube** [IO, MANDATORY, default=No default value]

string **filename** [INPUT, MANDATORY, default=no default value]

API details

Properties


Cube **simplecube** [IO, MANDATORY, default=No default value]

The input Cube to load

string **filename** [INPUT, MANDATORY, default=no default value]

The file to import

2.211. importImage

Full Name:	herschel.ia.toolbox.image.ImportImageTask
Alias:	importImage
Type:	Java Task - 
Import:	from herschel.ia.toolbox.image import ImportImageTask
Category:	task/image

Description

A Task to import images into an Image object.

A task which imports an Image from a file (bmp, gif, jpg, png, pnm, tiff or fits format). It is also possible to import a FITS file.

Example

Example 1: Import a jpg file

```
importImage(image = im, filename = -"myFile.jpg")
```

API Summary

Jython Syntax

```
importImage(image, filename)
```

Properties

[Image image](#) [INOUT, MANDATORY, default=No default value]

[string filename](#) [INPUT, MANDATORY, default=no default value]

API details

Properties

Image image [INOUT, MANDATORY, default=No default value]

The input Image to load.

string filename [INPUT, MANDATORY, default=no default value]

The file to import.

2.212. importRgbImage

Full Name:	herschel.ia.toolbox.image.ImportRgbImageTask
Alias:	importRgbImage
Type:	Java Task - 
Import:	from herschel.ia.toolbox.image import ImportRgbImageTask
Category:	task/image

Description

A Task to import color images into an RgbImage object.

A task which imports an Image from a file (bmp, gif, jpg, png, pnm, tiff or fits format).

Example

Example 1: Import a jpg file

```
importRgbImage(image = im, filename = "myFile.jpg")
```

API Summary

Jython Syntax

```
importRgbImage(image, filename)
```

Properties

[Image image](#) [INOUT, MANDATORY, default=No default value]

[string filename](#) [INPUT, MANDATORY, default=no default value]

API details

Properties


Image image [INOUT, MANDATORY, default=No default value]

The input Image to load.

string filename [INPUT, MANDATORY, default=no default value]

The file to import.

2.213. importSpectralCube

Full Name:	herschel.ia.toolbox.cube.ImportSpectralCubeTask
Alias:	importSpectralCube
Type:	Java Task - 
Import:	from herschel.ia.toolbox.cube import ImportSpectralCubeTask
Category:	task/cube

Description

The ImportSpectralCube task for SpectralSimpleCubes.

A task which imports a SpectralSimpleCube from a fits file

Example

Example 1: Import a fits file

```
importSpectralCube(spectralcube = cube, filename = -"myFile.fits")
```

API Summary

Jython Syntax

```
importSpectralCube(spectralcube, filename)
```

Properties

Cube `spectralcube` [IO, MANDATORY, default=No default value]

string `filename` [INPUT, MANDATORY, default=no default value]

API details

Properties

Cube `spectralcube` [IO, MANDATORY, default=No default value]

The input SpectralCube to load

string `filename` [INPUT, MANDATORY, default=no default value]

The file to import

2.214. importUfDirToPal

Full Name:	herschel.ia.toolbox.util.ImportUfDirToPalTask
Alias:	importUfDirToPal
Type:	Java Task - 
Import:	from herschel.ia.toolbox.util import ImportUfDirToPalTask
Category:	task

Description

Imports an observation context from a user friendly directory structure into a pool.

The importUfDirToPal task is used to ingest observations from a user friendly directory structure into a HIPE pool, so that they can be used with HIPE. The whole observation described by the XML file will be imported and its ObservationContext returned in a new variable.

Example

Example 1: Simple example

```
mypool = PoolManager.getPool("mib")
oc = importUfDirToPal(pool=myspool, dirin="sourcedir", xml="111111111-
herschel.ia.obs.ObservationContext-0.xml")
```

API Summary

Jython Syntax

```
<obsContext> = importUfDirToPal(<pool>, <dirin>, <xml>)
```

Properties

[ProductPool](#) **pool** [INPUT, MANDATORY, default=No default value]

[String](#) **dirin** [INPUT, MANDATORY, default=No default value]

[String](#) **xml** [INPUT, MANDATORY, default=No default value]

[ObservationContext](#) **obsContext** [OUTPUT, MANDATORY, default=no default value]

Limitations

The whole observation described by the XML file will be imported.

API details

Properties

ProductPool **pool** [INPUT, MANDATORY, default=No default value]

The pool to export products from.

String **dirin** [INPUT, MANDATORY, default=No default value]

The directory where the user friendly structure of products resides.

<code>String xml [INPUT, MANDATORY, default=No default value]</code>
--

The xml file that describes the observation to import.
--

<code>ObservationContext obsContext [OUTPUT, MANDATORY, default=no default value]</code>
--

The observation (context) imported into the pool.


See also

- [???](#)

History

- 16-01-2009 Created
- 2009-03-17 - Added: output
- 2009-09-30 - Modifiers: are created through getCustomModifiers (SCR HCSS-8253)
- 2010-07-01 - Output: changed from p_ref to oc (SCR HCSS-10700)

2.215. info

Full Name:	herschel.ia.inspector.InfoTask
Alias:	info
Type:	Java Task - 
Import:	from herschel.ia.inspector import InfoTask
Category:	task/general

Description

A task for inspecting the content of a variable in the jython session.

Users can inspect a variable by calling info from the command line. The task opens a pop up window displaying the structure of the specified item.

Examples

Example 1: info of myvariable

```
info("myvariable")
```

Example 2: info of myvariable setting a refresh rate of 2 seconds (2000 milliseconds)

```
info("myvariable", 2000)
```

API Summary

Jython Syntax

```
info("item")<br>
info("item", 5)
```

Properties

[String](#) **item** [INPUT, YES, default=No default value]

[Object](#) **refresh** [INPUT, NO, default=5000]

Limitations

No limitation

Miscellaneous

No miscellaneous

API details

Properties

[String](#) **item** [INPUT, YES, default=No default value]

The item to display the structure

<code>Object refresh [INPUT, NO, default=5000]</code>

The refresh rate (milliseconds)


See also

- [references](#)

History

- 2004-12-05 - NdC: first release.

2.216. InputImageNSignatureComponent

Full Name:	herschel.ia.toolbox.image.gui.InputImageNSignatureComponent
Type:	Java Class - 
Import:	from herschel.ia.toolbox.image.gui import InputImageNSignatureComponent

Description

The task dialog if only an image and n should be given as input.

A task dialog to serve as GUI if only an image and n should be given as input.

API Summary

Constructor
InputImageNSignatureComponent() The construction of a new InputImageNSignatureComponent.
Methods
check() Validates the current modifications.
clear() Clears the current modifications.
Map getParameters() Returns a map of parameters and modifiers storing the value selected by the user.
setSignature(SignatureApi signature) Setting the signature.
setVariableSelection(VariableSelection selection) Setting the variable selection.
JLabel newLabel(String title, String tooltip) Returns a label with the given title and tooltip.
JComponent getComponent() Returns the component.

API details

Constructor

InputImageNSignatureComponent()
The construction of a new InputImageNSignatureComponent.
The construction of a new InputImageNSignatureComponent.

Methods

check()
Validates the current modifications.

check()
Validates the current modifications for this InputImageNSignatureComponent.
clear()
Clears the current modifications.
Clears the current modifications for this InputImageNSignatureComponent
Map <code>getParameters()</code>
Returns a map of parameters and modifiers storing the value selected by the user.
Returns a map of parameters and modifiers storing the value selected by the user for this InputImageNSignatureComponent.
Return
Map
Returns a map of parameters and modifiers storing the value selected by the user for this InputImageNSignatureComponent.
setSignature(SignatureApi signature)
Setting the signature.
Sets the given signature as the signature for this InputImageNSignatureComponent.
Argument
SignatureApi signature [INPUT, MANDATORY]
setVariableSelection(VariableSelection selection)
Setting the variable selection.
Sets the given selection as the variable selection for this InputImageNSignatureComponent.
Argument
VariableSelection selection [INPUT, MANDATORY]
JLabel <code>newLabel(String title, String tooltip)</code>
Returns a label with the given title and tooltip.
Returns a label with the given title and tooltip.
Arguments
String title [INPUT, MANDATORY]
String tooltip [INPUT, MANDATORY]
Return
JLabel
Returns a label with the given title and tooltip.
JComponent <code>getComponent()</code>
Returns the component.

[JComponent](#) `getComponent()`


Returns the component for this InputImageNSignatureComponent.

Return

[JComponent](#)

Returns the component for this InputImageNSignatureComponent.

2.217. InputImageSignatureComponent

Full Name:	herschel.ia.toolbox.image.gui.InputImageSignatureComponent
Type:	Java Class - 
Import:	from herschel.ia.toolbox.image.gui import InputImageSignatureComponent

Description

The task dialog if only an image should be given as input.

A task dialog to serve as GUI if only an image should be given as input.

API Summary

Constructor
InputImageSignatureComponent() The construction of a new InputImageSignatureComponent.
Methods
check() Validates the current modifications.
clear() Clears the current modifications.
Map getParameters() Returns a map of parameters and modifiers storing the value selected by the user.
setSignature(SignatureApi signature) Setting the signature.
setVariableSelection(VariableSelection selection) Setting the variable selection.
JLabel newLabel(String title, String tooltip) Returns a label with the given title and tooltip.
JComponent getComponent() Returns the component.

API details

Constructor

InputImageSignatureComponent()
The construction of a new InputImageSignatureComponent.
The construction of a new InputImageSignatureComponent.

Methods

check()
Validates the current modifications.

check()
Validates the current modifications for this InputImageSignatureComponent.
clear()
Clears the current modifications.
Clears the current modifications for this InputImageSignatureComponent.
Map <code>getParameters()</code>
Returns a map of parameters and modifiers storing the value selected by the user.
Returns a map of parameters and modifiers storing the value selected by the user for this InputImageSignatureComponent.
Return
Map
Returns a map of parameters and modifiers storing the value selected by the user for this InputImageSignatureComponent.
setSignature(SignatureApi signature)
Setting the signature.
Sets the given signature as the signature for this InputImageSignatureComponent.
Argument
SignatureApi signature [INPUT, MANDATORY]
setVariableSelection(VariableSelection selection)
Setting the variable selection.
Sets the given selection as the variable selection for this InputImageSignatureComponent.
Argument
VariableSelection selection [INPUT, MANDATORY]
JLabel <code>newLabel(String title, String tooltip)</code>
Returns a label with the given title and tooltip.
Returns a label with the given title and tooltip.
Arguments
String title [INPUT, MANDATORY]
String tooltip [INPUT, MANDATORY]
Return
JLabel
Returns a label with the given title and tooltip.
JComponent <code>getComponent()</code>
Returns the component.

[JComponent](#) `getComponent()`


Returns the component for this InputImageSignatureComponent.

Return

[JComponent](#)

Returns a label with the given title and tooltip.

2.218. Int1d


Full Name:	herschel.ia.numeric.Int1d
Type:	Java Class - 
Import:	from herschel.ia.numeric import Int1d

Description

A rectangular numeric int array of rank 1.

Numeric arrays are part of the Numeric library. An explanation on how to use them, you can find in chapter **Numeric** -> **Arrays**. (Yes, it should be a link)

2.219. Int2d


Full Name:	herschel.ia.numeric.Int2d
Type:	Java Class - 
Import:	from herschel.ia.numeric import Int2d

Description

A rectangular numeric int array of rank 2.

Numeric arrays are part of the Numeric library. An explanation on how to use them, you can find in chapter **Numeric** -> **Arrays**. (Yes, it should be a link)

2.220. Int3d


Full Name:	herschel.ia.numeric.Int3d
Type:	Java Class - 
Import:	from herschel.ia.numeric import Int3d

Description

A rectangular numeric int array of rank 3.

Numeric arrays are part of the Numeric library. An explanation on how to use them, you can find in chapter **Numeric** -> **Arrays**. (Yes, it should be a link)

2.221. Int4d


Full Name:	herschel.ia.numeric.Int4d
Type:	Java Class - 
Import:	from herschel.ia.numeric import Int4d

Description

A rectangular numeric int array of rank 4.

Numeric arrays are part of the Numeric library. An explanation on how to use them, you can find in chapter **Numeric** -> **Arrays**. (Yes, it should be a link)

2.222. Int5d


Full Name:	herschel.ia.numeric.Int5d
Type:	Java Class - 
Import:	from herschel.ia.numeric import Int5d

Description

A rectangular numeric int array of rank 5.

Numeric arrays are part of the Numeric library. An explanation on how to use them, you can find in chapter **Numeric** -> **Arrays**. (Yes, it should be a link)

2.223. IntegratedMapDisplay

Full Name:	herschel.ia.gui.cube.IntegratedMapDisplay
Type:	Java Class - 
Import:	from herschel.ia.gui.cube import IntegratedMapDisplay

Description

A class to deal with IntegratedMap.

API Summary

Methods
setBegin(Double begin) Sets the begin of the straight line, associated with this
setEnd(Double end) Sets the end of the straight line, associated with this
ImageFigure getLine() Returns the straight line (ImageFigure), of which we wish to plot
Double getBegin() Returns the begin of the drawn line in PixelCoordinates.
Double getEnd() Returns the end of the drawn line in PixelCoordinates.
ArrayList getIntegratedMaps() Returns the integrated images in a list
String getSkyCoordinates() Returns the String version of the sky coordinates

API details

Methods

setBegin(Double begin) Sets the begin of the straight line, associated with this Velocity position map to the given point (in UserCoordinates). Argument <code>Double begin</code> [INPUT, MANDATORY]
setEnd(Double end) Sets the end of the straight line, associated with this Velocity position map to the given point (in UserCoordinates). Argument <code>Double end</code> [INPUT, MANDATORY]

ImageFigure `getLine()`

Returns the straight line (ImageFigure), of which we wish to plot the intensity in this IntegratedMapDisplay.

Return

[ImageFigure](#)

The straight line (ImageFigure), of which we wish to plot the intensity in this IntegratedMapDisplay.

Double `getBegin()`

Returns the begin of the drawn line in PixelCoordinates.

Return

[Double](#)

Returns the begin of the drawn line in PixelCoordinates.

Double `getEnd()`

Returns the end of the drawn line in PixelCoordinates.

Return

[Double](#)

Returns the end of the drawn line in PixelCoordinates.

ArrayList `getIntegratedMaps()`

Returns the integrated images in a list

Return

[ArrayList](#)

Returns the integrated images in a list

String `getSkyCoordinates()`

Returns the String version of the sky coordinates


(WorldCoordinates) of the begin and end of the drawn straight line.

Return

[String](#)

Returns the String version of the sky coordinates (WorldCoordinates) of the begin and end of the drawn straight line.

2.224. IntegrateMapFromCubeTask

Full Name:	herschel.ia.toolbox.cube.IntegrateMapFromCubeTask
Type:	Java Task - 
Import:	from herschel.ia.toolbox.cube import IntegrateMapFromCubeTask

Description

Task which integrate one or more "map" or images from a Cube.

In the CubeSpectrumAnalysisToolbox (via the integratedmapdisplay GUI), this task integrate one or more set of contiguous images defined by ranges. The output of this task is a set of simpleImages

API Summary

Properties
<code>Cube cube</code> [INPUT, MANDATORY, default=no default value]
<code>integer bbStack</code> [INPUT, default value 1, default=no default value]
<code>Double2d startArray</code> [INPUT, MANDATORY, default=no default value]
<code>Double2d endArray</code> [INPUT, MANDATORY, default=no default value]
<code>SimpleImages[] imageArray</code> [OUTPUT, MANDATORY, default=no default value]

API details

Properties

<code>Cube cube</code> [INPUT, MANDATORY, default=no default value]
The Cube containing the spectra to extract
<code>integer bbStack</code> [INPUT, default value 1, default=no default value]
give the number of image to inegrate
<code>Double2d startArray</code> [INPUT, MANDATORY, default=no default value]
The Array of spectral values for the start of each integration the dimension of this array is the number of integration to execute
<code>Double2d endArray</code> [INPUT, MANDATORY, default=no default value]
The Array of spectral values for the end of each integration the dimension of this array is the number of integration to execute
<code>SimpleImages[] imageArray</code> [OUTPUT, MANDATORY, default=no default value]
an array of simpleimages containing the result of each integration

See also


- [???](#)

- [???](#)

History

- 2009-05-20 - AG: creation

2.225. Integrator

Full Name:	herschel.ia.numeric.toolbox.integr.Integrator
Alias:	Integrator
Type:	Java Class - 
Import:	from herschel.ia.numeric.toolbox.integr import Integrator

Description

Interface for all integrators.

Available integrators are: - GaussHermiteIntegrator (closed interval) - GaussianQuad4Integrator (open interval) - GaussianQuad5Integrator (open interval) - GaussJacobiIntegrator (closed interval) - GaussLaguerreIntegrator (closed interval) - GaussLegendreIntegrator (open interval) - RectangularIntegrator (open interval) - TrapezoidalIntegrator (open interval) - SimpsonIntegrator (open interval) - RombergIntegrator (open interval)

Example

Example 1: Integration using the Romberg's method
<pre>from herschel.ia.numeric.all import * class MyFunction(RealFunction): def calc(self,x): return x*x f = MyFunction() i = RombergIntegrator(-3, 3) print i.integrate(f) # 18.0</pre>

API Summary

Jython Syntax
<code><r>=SomeOpenIntervalIntegrator(<a>,).integrate(<f>)</code>
<code><r>=SomeShutIntervalIntegrator().integrate(<f>)</code>
Properties
double a [INPUT, MANDATORY, default=no default value]
double b [INPUT, MANDATORY, default=no default value]
RealFunction f [INPUT, MANDATORY, default=no default value]
double r [OUTPUT, MANDATORY, default=no default value]

API details

Properties

double a [INPUT, MANDATORY, default=no default value]
Lower limit of integration, only mandatory for open interval integrators.
double b [INPUT, MANDATORY, default=no default value]
Upper limit of integration, only mandatory for open interval integrators. It must be greater or equal than the lower limit.


RealFunction f [INPUT, MANDATORY, default=no default value]
--

The function must implement the calc method; see example below.

double r [OUTPUT, MANDATORY, default=no default value]

Returns the integral of the given function between the specified limits; that is, the area behind the function within these limits.

2.226. IntensityCalculator

Full Name:	herschel.ia.toolbox.image.IntensityCalculator
Type:	Java Class - 
Import:	from herschel.ia.toolbox.image import IntensityCalculator

Description

A class to apply sky estimation algorithms.

A class to estimate the sky intensity through five different sky estimation algorithms : average, median, mean-median, synthetic mode and daophot.

API Summary

Methods
<p>double <code>getAverageIntensity</code>(Doubleld intensities, Doubleld fractions, boolean fractional)</p> <p>Returns the average intensity.</p>
<p>double <code>getAverageIntensityFractionalPixels</code>(Doubleld intensities, Doubleld fractions)</p> <p>Returns the average intensity for fractional pixels.</p>
<p>double <code>getAverageIntensityEntirePixels</code>(Doubleld intensities)</p> <p>Returns the average intensity for entire pixels.</p>
<p>double <code>getMedianIntensity</code>(Doubleld intensities, Doubleld fractions, boolean fractional)</p> <p>Returns the median intensity.</p>
<p>double <code>getMedianIntensityFractionalPixels</code>(Doubleld intensities, Doubleld fractions)</p> <p>Returns the median intensity for fractional pixels.</p>
<p>double <code>getMedianIntensityEntirePixels</code>(Doubleld intensities)</p> <p>Returns the median intensity for entire pixels.</p>
<p>double[] <code>getMeanMedianIntensityAndError</code>(Doubleld intensities, Doubleld fractions, boolean fractional)</p> <p>Returns the mean-median intensity and error.</p>
<p>double[] <code>getSyntheticModeIntensityAndError</code>(Doubleld intensities, Doubleld fractions, boolean fractional)</p> <p>Returns the synthetic mode intensity and error.</p>
<p>double[] <code>getDaophotIntensityAndError</code>(Doubleld intensities, Doubleld fractions, boolean fractional)</p> <p>Returns the daophot intensity and error.</p>
<p><code>sort</code>(Doubleld intensities, Doubleld fractions)</p> <p>Sorting.</p>
<p><code>quickSort</code>(Doubleld intensities, Doubleld fractions, int lo0, int hi0)</p> <p>Quick sort.</p>

API details

Methods

```
double getAverageIntensity(DoubleId intensities, DoubleId fractions, boolean fractional)
```

Returns the average intensity.

Returns the average intensity for the given list of intensities and corresponding fractions.

Arguments

[DoubleId](#) **intensities** [INPUT, MANDATORY]

[DoubleId](#) **fractions** [INPUT, MANDATORY]

boolean **fractional** [INPUT, MANDATORY]

Return

double

Returns the average intensity for the given list of intensities and corresponding fractions.

```
double getAverageIntensityFractionalPixels(DoubleId intensities, DoubleId fractions)
```

Returns the average intensity for fractional pixels.

Returns the average intensity for the given list of intensities and corresponding fractions for fractional pixels.

Arguments

[DoubleId](#) **intensities** [INPUT, MANDATORY]

[DoubleId](#) **fractions** [INPUT, MANDATORY]

Return

double

Returns the average intensity for the given list of intensities and corresponding fraction for fractional pixels.

```
double getAverageIntensityEntirePixels(DoubleId intensities)
```

Returns the average intensity for entire pixels.

Returns the average intensity for the given list of intensities and corresponding fractions for entire pixels.

Argument

[DoubleId](#) **intensities** [INPUT, MANDATORY]

Return

double

Returns the average intensity for the given list of intensities for entire pixels.

```
double getMedianIntensity(DoubleId intensities, DoubleId fractions, boolean fractional)
```

Returns the median intensity.


```
double getMedianIntensity(DoubleId intensities, DoubleId fractions, boolean fractional)
```

Returns the median intensity for the given list of intensities and corresponding fractions.

Arguments

[DoubleId](#) **intensities** [INPUT, MANDATORY]

[DoubleId](#) **fractions** [INPUT, MANDATORY]

boolean **fractional** [INPUT, MANDATORY]

Return

double

Returns the median intensity for the given list of intensities and corresponding fractions.

```
double getMedianIntensityFractionalPixels(DoubleId intensities, DoubleId fractions)
```

Returns the median intensity for fractional pixels.

Returns the median intensity for the given list of intensities and corresponding fractions for fractional pixels.

Arguments

[DoubleId](#) **intensities** [INPUT, MANDATORY]

[DoubleId](#) **fractions** [INPUT, MANDATORY]

Return

double

Returns the median intensity for the given list of intensities and corresponding fraction for fractional pixels.

```
double getMedianIntensityEntirePixels(DoubleId intensities)
```

Returns the median intensity for entire pixels.

Returns the median intensity for the given list of intensities and corresponding fractions for entire pixels.

Argument

[DoubleId](#) **intensities** [INPUT, MANDATORY]

Return

double

Returns the median intensity for the given list of intensities for entire pixels.

```
double[] getMeanMedianIntensityAndError(DoubleId intensities, DoubleId fractions, boolean fractional)
```

Returns the mean-median intensity and error.

Returns the mean-median intensity and error for the given list of intensities and corresponding intensities.

Arguments

[DoubleId](#) **intensities** [INPUT, MANDATORY]

```
double[] getMeanMedianIntensityAndError(DoubleId intensities,
DoubleId fractions, boolean fractional)
```

[DoubleId](#) **fractions** [INPUT, MANDATORY]

boolean **fractional** [INPUT, MANDATORY]

Return

double[]

Returns the mean-median intensity and error for the given list of intensities and corresponding intensities.

```
double[] getSyntheticModeIntensityAndError(DoubleId intensities,
DoubleId fractions, boolean fractional)
```

Returns the synthetic mode intensity and error.

Returns the synthetic mode intensity and error for the given list of intensities and corresponding fractions.

Arguments

[DoubleId](#) **intensities** [INPUT, MANDATORY]

[DoubleId](#) **fractions** [INPUT, MANDATORY]

boolean **fractional** [INPUT, MANDATORY]

Return

double[]

Returns the synthetic mode intensity and error for the given list of intensities and corresponding fractions.

```
double[] getDaophotIntensityAndError(DoubleId intensities,
DoubleId fractions, boolean fractional)
```

Returns the daophot intensity and error.

Returns the daophot intensity and error for the given list of intensities and corresponding fractions.

Arguments

[DoubleId](#) **intensities** [INPUT, MANDATORY]

[DoubleId](#) **fractions** [INPUT, MANDATORY]

boolean **fractional** [INPUT, MANDATORY]

Return

double[]

Returns the daophot intensity and error for the given list of intensities and corresponding fractions.

```
sort(DoubleId intensities, DoubleId fractions)
```

Sorting.

Sorts the given list of intensities in ascending order and sorts the list of fractions in the same way.

Arguments

[DoubleId](#) **intensities** [INPUT, MANDATORY]

[DoubleId](#) **fractions** [INPUT, MANDATORY]

```
quickSort(Double1d intensities, Double1d fractions, int lo0, int hi0)
```

Quick sort.

Sorts the given list of intensities and fractions for elements between the given indices, in ascending order.

Arguments


[Double1d](#) **intensities** [INPUT, MANDATORY]

[Double1d](#) **fractions** [INPUT, MANDATORY]

int **lo0** [INPUT, MANDATORY]

int **hi0** [INPUT, MANDATORY]

2.227. IntensityMapGui

Full Name:	herschel.ia.gui.cube.IntensityMapGui
Type:	Java Class - 
Import:	from herschel.ia.gui.cube import IntensityMapGui

Description

A class to deal with line intensity map

API Summary


Method
ArrayList getIntegratedMaps() Returns the integrated images in a list

API details

Method

ArrayList getIntegratedMaps()
Returns the integrated images in a list
Return
ArrayList
Returns the integrated images in a list

2.228. interruptable

Full Name:	herschel.ia.task.example.InterruptableTask
Alias:	interruptable
Type:	Java Task - 
Import:	from herschel.ia.task.example import InterruptableTask
Category:	developer

Description

Demonstrate the use of the Ctrl-s on top a task

Show how long running activities can be interrupted. Capture Ctrl-s events and provide a recovery action

Examples

Example 1: run the task

```
from herschel.ia.task.example import InterruptableTask
interruptable = InterruptableTask()
interruptable()
#Press Ctrl-s in Console to stop
```

Example 2: run the task with 1000 iteration

```
interruptable(1000)
```

API Summary

Jython Syntax

```
interruptable([<iterations>])
```

Property

```
Integer iterations [INPUT, OPTIONAL, default=10000]
```

Limitations

The current implementation allows deleting of packages and class, no pre-filter is performed

API details

Property

```
Integer iterations [INPUT, OPTIONAL, default=10000]
```

The number of iterations to be performed

History

- 2004-07-13 - NdC: first release.

- 2010-01-20 - JDS: fix documentation

2.229. InverseDFT2dTask

Full Name:	herschel.ia.toolbox.image.InverseDFT2dTask
Type:	Java Task - 
Import:	from herschel.ia.toolbox.image import InverseDFT2dTask

Description

A Task for two dimensional inverse Fast Fourier Transforms.

A Task that calculates the inverse 2D Discrete Fourier Transform and the power spectrum.

API Summary


Properties
Numeric2dData image [INPUT, MANDATORY, default=No default value]
Complex2d transform [OUTPUT, MANDATORY, default=No default value]
Double2d spectrum [OUTPUT, MANDATORY, default=No default value]

API details

Properties

Numeric2dData image [INPUT, MANDATORY, default=No default value]
The input image.
Complex2d transform [OUTPUT, MANDATORY, default=No default value]
The transform.
Double2d spectrum [OUTPUT, MANDATORY, default=No default value]
The spectrum.

2.230. InverseFFT2dTask

Full Name:	herschel.ia.toolbox.image.InverseFFT2dTask
Type:	Java Task - 
Import:	from herschel.ia.toolbox.image import InverseFFT2dTask

Description

A Task for two dimensional inverse Fast Fourier Transforms.

A Task that calculates the inverse 2D Fast Fourier Transform and the power spectrum.

API Summary


Properties
Numeric2dData image [INPUT, MANDATORY, default=No default value]
Complex2d transform [OUTPUT, MANDATORY, default=No default value]
Double2d spectrum [OUTPUT, MANDATORY, default=No default value]

API details

Properties

Numeric2dData image [INPUT, MANDATORY, default=No default value]
The input image.
Complex2d transform [OUTPUT, MANDATORY, default=No default value]
The transform.
Double2d spectrum [OUTPUT, MANDATORY, default=No default value]
The spectrum.

2.231. INVERSE

Full Name:	herschel.ia.numeric.toolbox.matrix.MatrixInverse
Alias:	INVERSE
Type:	Java Class - 
Import:	from herschel.ia.numeric.toolbox.matrix import MatrixInverse

Description

Returns the inverse of a square matrix.

Example

Example 1: Apply INVERSE to a Float2d matrix

```
x=Float2d([ [1,2],[3,4] -])
print INVERSE(x) # [ [-2.0,1.0], [1.5,-0.5] -]
```

API Summary

Jython Syntax

```
<y>=INVERSE(<x>)
```

Properties

any square matrix **x** [INPUT, MANDATORY, default=no default value]

double **y** [INPUT, NOT_MANDATORY, default=false]

Miscellaneous

Does not work for complex matrices.

API details

Properties


any square matrix **x** [INPUT, MANDATORY, default=no default value]

Any square matrix

double **y** [INPUT, NOT_MANDATORY, default=false]

Returns a double

2.232. IS_FINITE

Full Name:	herschel.ia.numeric.toolbox.basic.IsFinite
Alias:	IS_FINITE
Type:	Java Class - 
Import:	from herschel.ia.numeric.toolbox.basic import IsFinite

Description

Returns a boolean array where each element is true if the corresponding element input array is a finite number, false otherwise.

Example

Example 1: Apply IS_FINITE on a Double1d

```
x=Double1d([1,Double.NaN,Double.NEGATIVE_INFINITY,Double.POSITIVE_INFINITY])
print IS_FINITE(x) # [true,false,false,false]
```

API Summary

Jython Syntax

```
<y>=IS_FINITE (<x>)
```

Properties

any array or number **x** [INPUT, MANDATORY, default=no default value]

boolean array or a boolean **y** [OUTPUT, MANDATORY, default=no default value]

API details

Properties

any array or number **x** [INPUT, MANDATORY, default=no default value]

An array or a number


boolean array or a boolean **y** [OUTPUT, MANDATORY, default=no default value]

Returns a boolean array (or a boolean if the input is not an array) where each element is true if the corresponding element input array is a finite number, false otherwise

See also

- [IS_NAN](#)
- [IS_INFFINITE](#)

2.233. IS_INFINITE

Full Name:	herschel.ia.numeric.toolbox.basic.IsInfinite
Alias:	IS_INFINITE
Type:	Java Class - 
Import:	from herschel.ia.numeric.toolbox.basic import IsInfinite

Description

Returns a boolean array where each element is true if the corresponding element input array is infinitely large in magnitude, false otherwise.

Example

Example 1: Apply IS_INFINITE on a Double1d

```
x=Double1d([1,Double.NaN,Double.NEGATIVE_INFINITY,Double.POSITIVE_INFINITY])
print IS_INFINITE(x) # [false,false,true,true]
```

API Summary

Jython Syntax

```
<y>=IS_INFINITE (<x>)
```

Properties

[any array or number **x** \[INPUT, MANDATORY, default=no default value\]](#)
[boolean array or a boolean **y** \[OUTPUT, MANDATORY, default=no default value\]](#)

API details

Properties

any array or number **x [INPUT, MANDATORY, default=no default value]**

An array or a number


boolean array or a boolean **y [OUTPUT, MANDATORY, default=no default value]**

Returns a boolean array (or a boolean if the input is not an array) where each element is true if the corresponding element input array is input array is infinitely large in magnitude, false otherwise.

See also

- [IS_NAN](#)
- [IS_FINITE](#)

2.234. IS_NAN

Full Name:	herschel.ia.numeric.toolbox.basic.IsNaN
Alias:	IS_NAN
Type:	Java Class - 
Import:	from herschel.ia.numeric.toolbox.basic import IsNaN

Description

Returns a boolean array where each element is true if the corresponding element input array is flagged as Not a Number, false otherwise.

Example

Example 1: Apply IS_FINITE on a Double1d

```
x=Double1d([1,Double.NaN,Double.NEGATIVE_INFINITY,Double.POSITIVE_INFINITY])
print IS_NAN(x) # [false,true,false,false]
```

API Summary

Jython Syntax

```
<y>=IS_NAN(<x>)
```

Properties

[any array or number **x** \[INPUT, MANDATORY, default=no default value\]](#)
[boolean array or a boolean **y** \[OUTPUT, MANDATORY, default=no default value\]](#)

API details

Properties

any array or number **x [INPUT, MANDATORY, default=no default value]**

An array or a number


boolean array or a boolean **y [OUTPUT, MANDATORY, default=no default value]**

Returns a boolean array (or a boolean if the input is not an array) where each element is true if the corresponding element of the input array is flagged as Not a Number, false otherwise.

See also

- [IS_FINITE](#)
- [IS_INFFINITE](#)

2.235. JavaQuery

Full Name:	herschel.ia.pal.query.JavaQuery
Type:	Java Class - 
Import:	from herschel.ia.pal.query import JavaQuery

Description

java query formulates a query on anything of a ProductRef, it allow specific optimizations to be coded in Java.

Typically there's no JythonInterpreter involved. So this type of query is faster than other types(e.g. MetaQuery, FullQuery) of queries.


Example

Example 1: Untitled
<pre>Example of a java query: q=JavaQuery(MyProduct.class,MetaKeyValuePredicate("obsid","1342187145"))</pre>

See also

- [???](#)

2.236. KURTOSIS

Full Name:	herschel.ia.numeric.toolbox.basic.Kurtosis
Alias:	KURTOSIS
Type:	Java Class - 
Import:	from herschel.ia.numeric.toolbox.basic import Kurtosis

Description

Yields the kurtosis of the elements in the input array.

The kurtosis is the fourth moment divided by the standard deviation raised to the fourth power. In addition, this implementation subtracts 3 since 3 is the kurtosis for a normal distribution.

Example

Example 1: Apply KURTOSIS on a Float1d

```
x=Float1d([1,3,2,3,4])
print KURTOSIS(x) # --1.74840236686
```

API Summary

Jython Syntax

```
<y>=KURTOSIS(<x>)
```

Properties

[any scalar array **x** \[INPUT, MANDATORY, default=no default value\]](#)

[double **y** \[OUTPUT, MANDATORY, default=no default value\]](#)

API details

Properties

any scalar array **x [INPUT, MANDATORY, default=no default value]**

The input array integral or floating-point arrays; the former implicitly transformed to a double array.

double **y [OUTPUT, MANDATORY, default=no default value]**


Returns a double

See also

- [MEAN](#)
- [MEDIAN](#)
- [SKEWNESS](#)
- [STDDEV](#)

- [VARIANCE](#)

2.237. LayerStruct

Full Name:	herschel.ia.gui.explorer.table.LayerStruct
Type:	Java Class - 
Import:	from herschel.ia.gui.explorer.table import LayerStruct

Description

LayerStruct is a data structure used in OverPlotter/TablePlotter. It has many methods which allow users to access extracted data and flags. It has many setters (setXXX) and getters (getXXX). It works as a bookkeeping as well. It keeps tracks on the properties of the active layer. It saves all the personalities of an active layer when the layer becomes inactive and re-display them when the layer becomes active.

API Summary

Constructors
LayerStruct(TableDataset table, int layerID)
LayerStruct(paramType table, int layerID, String layerName)
LayerStruct(TableDataset table, int layerID, Bool2d flags)
LayerStruct(TableDataset table, int layerID, Bool1d flags)
Methods
TableDataset getExtractedTableDataset()
BooleanArray getFlags()

API details

Constructors

<code>LayerStruct(TableDataset table, int layerID)</code>
<p>Arguments</p> <p>TableDataset table [INPUT, MANDATORY, default=no default value] The input table contains one or more column and each column can have numeric or none numeric data or multi-dimension. However, such columns will be filtered out when the table is plotted.</p> <p>int layerID [INPUT, MANDATORY, default=no default value] This layerID is used as a layer counter to decide the color and symbol in LayerStatus Each layer has an unique ID. The id starts from 0, and increase by 1 each time a new layer is created.</p>
<code>LayerStruct(paramType table, int layerID, String layerName)</code>
<p>Arguments</p> <p>paramType table [INPUT, MANDATORY, default=no default value] The input table contains one or more column and each column can have numeric or none numeric data or multi-dimension. However, such columns will be filtered out when the table is plotted.</p> <p>int layerID [INPUT, MANDATORY, default=no default value]</p>

LayerStruct(paramType table, int layerID, String layerName)
<p>This layerID is used as a layer counter to decide the color and symbol in LayerStatus Each layer has an unique ID. The id starts from 0, and increase by 1 each time a new layer is created.</p> <p>String layerName [INPUT, MANDATORY, default=no default value]</p> <p>The layerName must be unique for each table.</p>

LayerStruct(TableDataset table, int layerID, Bool2d flags)
<p>Arguments</p> <p>TableDataset table [INPUT, MANDATORY, default=no default value]</p> <p>The input table contains one or more column and each column can have numeric or none numeric data or multi-dimension. However, such columns will be filtered out when the table is plotted.</p> <p>int layerID [INPUT, MANDATORY, default=no default value]</p> <p>This layerID is used as a layer counter to decide the color and symbol in LayerStatus Each layer has an unique ID. The id starts from 0, and increase by 1 each time a new layer is created.</p> <p>Bool2d flags [INPUT, MANDATORY, default=no default value]</p> <p>The flags must have the same rank and size as the table parameter</p>

LayerStruct(TableDataset table, int layerID, Bool1d flags)
<p>Arguments</p> <p>TableDataset table [INPUT, MANDATORY, default=no default value]</p> <p>The input table contains one or more column and each column can have numeric or none numeric data or multi-dimension. However, such columns will be filtered out when the table is plotted.</p> <p>int layerID [INPUT, MANDATORY, default=no default value]</p> <p>This layerID is used as a layer counter to decide the color and symbol in LayerStatus Each layer has an unique ID. The id starts from 0, and increase by 1 each time a new layer is created.</p> <p>Bool1d flags [INPUT, MANDATORY, default=no default value]</p> <p>The the length of the flags array must equal to the number of columns of the input table</p>

Methods


TableDataset getExtractedTableDataset()
<p>Return</p> <p>TableDataset</p> <p>- return the extracted dataset</p>

BooleanArray getFlags()
<p>Return</p> <p>BooleanArray</p> <p>- return the flags. The flags can be either Bool1d or Bool2d. 1/12/10 Fixed bug, the returned flags contains an extra column which is for TPL use. This column has to be removed.</p>

History

- 2006-11-06 - first: version
- 2008-04-22 re-factored this code
- 2008-12-14 - adding: URM documentation
- 2010-01-13 - Implement: HCSS-9144
- 2010-02-01 - Implement: HCSS-8996 and HCSS-9331

2.238. LevenbergMarquardtFitter

Full Name:	herschel.ia.numeric.toolbox.fit.LevenbergMarquardtFitter
Type:	Java Class - 
Import:	from herschel.ia.numeric.toolbox.fit import LevenbergMarquardtFitter

Description

Non-linear fitter using the Levenberg-Marquardt method.

Implementation of the Levenberg-Marquardt algorithm to fit the parameters of a non-linear model. It is a gradient fitter which uses partial derivatives to find the downhill gradient. Consequently it ends in the first minimum it finds.

An introduction to the use of fit package can be found [here](#).

At least once have a look at the [background](#) on the organization and structure of the package.

The fit/demo directory contains worked [examples](#) on almost all aspects of of the package.

Example

Example 1: LevenbergMarquardtFitter (for non-linear models)

```
# assume x and y are Double1d data arrays.
gauss = GaussModel( -)          # Gaussian
gauss += PolynomialModel( 1 -)  # add linear background
print gauss.getNumberOfParameters() # 5 (= 3 for Gauss + 2 for line)
gauss.keepFixed( Int1d([2], Double1d([0.1])) # keep width fixed at 0.1
lmfit = LevenbergMarquardtFitter( x, gauss -)
param = lmfit.fit( y -)
print param.length()           # 4 (= 5 -- 1 fixed)
stdev = lmfit.getStandardDeviation() # stdevs on the parameters
chisq = lmfit.getChiSquared()
scale = lmfit.getScale()        # noise scale
yfit = lmfit.getResult()        # fitted values
yband = lmfit.monteCarloError() # 1 sigma confidence region
# for diagnostics (or just for fun)
lmfit = LevenbergMarquardtFitter( x, gauss -)
lmfit.setVerbose( 10 -)        # report every 10th iteration
plotter = IterationPlotter()    # from herschel.ia.toolbox.fit
lmfit.setPlotter( plotter, 20 -) # make a plot every 20th iteration
param = lmfit.fit( y -)
```


Limitations

1. LMF is **not** guaranteed to find the global minimum.
2. LMF does **not** work with limits.
3. The calculation of the evidence is an Gaussian approximation which is only exact for linear models with a fixed scale.

Miscellaneous

In case of problems look at the [trouble shooting](#) section.

2.239. LinearInterpolator

Full Name:	herschel.ia.numeric.toolbox.interp.LinearInterpolator
Alias:	LinearInterpolator
Type:	Java Class - 
Import:	from herschel.ia.numeric.toolbox.interp import LinearInterpolator

Description

Given a set of knots (x/y data), this function computes values at arbitrary positions

by linearly interpolating the y value based on the knot lower and higher than the input x position. This can only be applied to numeric arrays of rank 1.

Example

Example 1: Create and apply a LinearInterpolator on a Double1d

```
# create some knots
x=Double1d.range(10) # [0.0,1.0,2.0,3.0,4.0,5.0,6.0,7.0,8.0,9.0]
f=LinearInterpolator(x,SQUARE(x))
u=Double1d([1,1.5,2,2.5,3,3.5])
print f(u) # [1.0,2.5,4.0,6.5,9.0,12.5]
```

API Summary

Jython Syntax

```
<f>=LinearInterpolator(<x>,<y>[,allowExtrapolation])
```

Properties

[Double1d x](#) [INPUT, MANDATORY, default=no default value]

[Double1d y](#) [INPUT, NOT MANDATORY, default=false]

[boolean allowExtrapolation](#) [INPUT, NOT MANDATORY, default=false]

API details

Properties

[Double1d x](#) [INPUT, MANDATORY, default=no default value]

The knots are Double1d

[Double1d y](#) [INPUT, NOT_MANDATORY, default=false]

The knots are Double1d

[boolean allowExtrapolation](#) [INPUT, NOT_MANDATORY, default=false]


Extrapolation is not allowed by default. Boolean.TRUE allows extrapolation.

See also

- [CubicSplineInterpolator](#)

- [NearestNeighborInterpolator](#)

2.240. LineIntensityMapTask

Full Name:	herschel.ia.toolbox.cube.LineIntensityMapTask
Type:	Java Task - 
Import:	from herschel.ia.toolbox.cube import LineIntensityMapTask

Description

LineIntensityMapTask

a task to compute a line intensity integration map

API Summary

Properties
Cube cube [INPUT, MANDATORY, default=no default value]
Cube cube [INPUT, MANDATORY, default=no default value]
Integer continuumSubtraction [INPUT, MANDATORY, default=no default value]
DoubleId regionStartArray [INPUT, MANDATORY, default=no default value]
DoubleId regionEndArray [INPUT, MANDATORY, default=no default value]
type PolyDegree [Integer, MANDATORY, default=no default value]
String fitModel [INPUT, MANDATORY, default=no default value]
String mode [INPUT, MANDATORY, default=no default value]
type centralPos [double, MANDATORY, default=no default value]
DoubleId rangeForFit [INPUT, MANDATORY, default=no default value]
String typeOfLine [INPUT, MANDATORY, default=no default value]
SimpleImage lineIntMap [OUTPUT, MANDATORY, default=no default value]
SimpleCube fittedLineCube [OUTPUT, MANDATORY, default=no default value]
SimpleCube continuumCube [OUTPUT, MANDATORY, default=no default value]
Cube cubeWithoutContinuum [OUTPUT, MANDATORY, default=no default value]
Cube residualCube [OUTPUT, MANDATORY, default=no default value]

API details

Properties

Cube cube [INPUT, MANDATORY, default=no default value]
The Cube from which we want to compute a line intensity map.
Cube cube [INPUT, MANDATORY, default=no default value]
The Cube from which we want to compute a line intensity map.

Integer continuumSubtraction [INPUT, MANDATORY, default=no default value]
the continuum subtraction mode: 0 No subtraction , 1 fit on an region of the spectrum, (this mode need the parameter regionStartArray regionEndArray to be filled) 2 continuum subtraction combined with the line fit in one model
DoubleId regionStartArray [INPUT, MANDATORY, default=no default value]
The array of the firsts indexes of the regions to be use for the continuum subtraction
DoubleId regionEndArray [INPUT, MANDATORY, default=no default value]
The array of the lasts indexes of the regions to be use for the continuum subtraction
type PolyDegree [Integer, MANDATORY, default=no default value]
the degree of the polynomial for the continuum fit.
String fitModel [INPUT, MANDATORY, default=no default value]
the name of the model to be used for the line fitting. in the first release can only be "Gaussian" but in the futur voight profil and sinc (??) will be added
String mode [INPUT, MANDATORY, default=no default value]
define the fitting mode (automatic or manual , with initial guesses)
type centralPos [double, MANDATORY, default=no default value]
the guess on the central position if needed
DoubleId rangeForFit [INPUT, MANDATORY, default=no default value]
an array containing the limit on the validity domain for the Fitt : is sued to reject false detection
String typeOfLine [INPUT, MANDATORY, default=no default value]
a string which said if the line to fitt is an EMISSION line or an ABSORPTION line
SimpleImage lineIntMap [OUTPUT, MANDATORY, default=no default value]
The main output , the map of the integrated flux on the fitted lines without continuum
SimpleCube fittedLineCube [OUTPUT, MANDATORY, default=no default value]
The cube containing the final fitted line, without the continuum
SimpleCube continuumCube [OUTPUT, MANDATORY, default=no default value]
The cube of the fitted continumm
Cube cubeWithoutContinuum [OUTPUT, MANDATORY, default=no default value]
Contain the original cube after subtraction of the continuum cube: Each spectrum is "flattened"


<code>Cube residualCube [OUTPUT, MANDATORY, default=no default value]</code>
--

the residual Cube which contain the final residual from cube minus continuum minus fitted line
--

See also

- [???](#)
- [???](#)

2.241. ListContext

Full Name:	herschel.ia.pal.ListContext
Type:	Java Class - 
Import:	from herschel.ia.pal import ListContext

Description

Groups products (or other Contexts that in turn group products) in a

list-like structure. Products grouped in a ListContext can be subsequently retrieved by an index (with index=0 corresponding to the first product in the list), through the 'refs' method. Note: users may be confused as to why there is a need to have to go through a 'refs' method to add or access products from a ListContext. This is due to a technical limitation in the design which will be addressed in due course.

Examples

Example 1: Adding a product to a ListContext product = Product() ref =

```
storage.save(product) # the ref is a ProductRef object listcontext
= ListContext() listcontext.refs.add(ref)
```

Example 2: Getting the first product from a ListContext ref_first =

```
lcontext.refs.get(0) product = ref_first.product
```

Example 3: Saving a ListContext to ProductStorage (same way as any other

```
product) ref_context = storage.save(listcontext)
```

API Summary

Method
List getRefs() get the list of ProductRefs stored. From this list, you can put

API details

Method

List getRefs()
get the list of ProductRefs stored. From this list, you can put products into the ListContext, or retrieve products by index.
Return
List
A list of product refs.
Examples
Putting a product into the the ListContext ref =

List `getRefs()`

```
storage.save(product) # the ref is a ProductRef object  
listcontext.refs.add(ref)
```


Getting the first product from the ListContext `ref_first =`

```
lcontext.refs.get(0)
```

See also

- [???](#)

2.242. localStoreCopier

Full Name:	herschel.ia.toolbox.util.LocalStoreCopierTask
Alias:	localStoreCopier
Type:	Java Task - 
Import:	from herschel.ia.toolbox.util import LocalStoreCopierTask
Category:	task

Description

Creates a (possibly compressed) copy of a Local Store. This copy can be distributed to other HIPE installations and imported easily.

Example

Example 1: Copying a local store into a zip

```
store = PoolManager.getPool("mib")
localStoreCopier(store = -"mib", archive = -"~/mycopy", compression = -"ZIP")
```

API Summary

Jython Syntax

```
localStoreCopier(<store> [, warn], <archive> [, <compression>])
```

Properties

[String store](#) [INPUT, MANDATORY, default=null]

[Boolean warn](#) [INPUT, OPTIONAL, default=null]

[String archive](#) [INPUT, MANDATORY, default=null]

[String compression](#) [INPUT, OPTIONAL, default="ZIP"]

Limitations

The order of parameters is awkward (required to have a balanced automatic UI)

API details

Properties

[String store](#) [INPUT, MANDATORY, default=null]

LocalStore id where to save the product.

[Boolean warn](#) [INPUT, OPTIONAL, default=null]

If true, warn if overwriting

[String archive](#) [INPUT, MANDATORY, default=null]

Path of the copy


`String` `compression` [INPUT, OPTIONAL, default="ZIP"]

The type of compression in the output file. Valid values are "NONE", "ZIP" (, "TAR" and "TGZ")

History

- 2010-04-28 - JDS: first release

2.243. localStoreWriter

Full Name:	herschel.ia.toolbox.util.LocalStoreWriterTask
Alias:	localStoreWriter
Type:	Java Task - 
Import:	from herschel.ia.toolbox.util import LocalStoreWriterTask
Category:	task

Description

Saves a product in a Local Store, either already defined, or a new

one chosen by name by the user. The gui allows to create/reconfigure new local stores in the lower part (even outside of the default location), as a plus over the functionality available solely via console.

Example

Example 1: Saving a product into a local store

```
p = Product()
#mib is the default location or has been configured via GUI
localStoreWriter(product = p, store = "-mib")
```

API Summary

Jython Syntax

```
localStoreWriter(<product>, <store>)
```

Properties

[Product](#) **product** [INPUT, MANDATORY, default=null]

[String](#) **store** [INPUT, MANDATORY, default=null]

API details

Properties

Product **product** [INPUT, MANDATORY, default=null]

Product to be saved.


String **store** [INPUT, MANDATORY, default=null]

LocalStore id where to save the product. Name cannot contain blanks or colons.

History

- 2008-11-11 - JSS: first release
- 2009-02-20 - JDS: cleanup
- 2009-08-30 - JSS: store name and location fields separated in the modifier

2.244. LOG10

Full Name:	herschel.ia.numeric.toolbox.basic.Log10
Alias:	LOG10
Type:	Java Class - 
Import:	from herschel.ia.numeric.toolbox.basic import Log10

Description

Gives the logarithm with base 10 of a number or a numeric array: $y = \text{LOG}_{10}(x)$.

Example

Example 1: Apply LOG10 on a Double1d

```
x=Double1d([1,10,100])
print LOG10(x) # [0.0,0.9999999999999999,1.999999999999998]
print ROUND(LOG10(x)) # [0.0,1.0,2.0]
```

API Summary

Jython Syntax

```
<y>=LOG10(<x>)
```

Properties

[any array or number **x** \[INPUT, MANDATORY, default=no default value\]](#)

[boolean array or a boolean **y** \[OUTPUT, MANDATORY, default=no default value\]](#)

API details

Properties

any array or number **x [INPUT, MANDATORY, default=no default value]**

An array or a number


boolean array or a boolean **y [OUTPUT, MANDATORY, default=no default value]**

Returns an array (or a number if the input is not an array) where each element is the the logarithm with base 10 of the corresponding element of the input array

See also

- [LOG](#)
- [LogN](#)

2.245. LOG

Full Name:	herschel.ia.numeric.toolbox.basic.Log
Alias:	LOG
Type:	Java Class - 
Import:	from herschel.ia.numeric.toolbox.basic import Log

Description

Computes the function $y=\text{LOG}_e(x)$, the natural logarithm.

Example

Example 1: Apply LOG on a Double1d

```
x=Double1d([0,1])
print LOG(EXP(x)) # [0.0,1.0]
```

API Summary

Jython Syntax

```
<y>=LOG(<x>)
```

Properties

[any array or number **x** \[INPUT, MANDATORY, default=no default value\]](#)

[boolean array or a boolean **y** \[OUTPUT, MANDATORY, default=no default value\]](#)

API details

Properties

any array or number **x [INPUT, MANDATORY, default=no default value]**

An array or a number


boolean array or a boolean **y [OUTPUT, MANDATORY, default=no default value]**

Returns an array (or a number if the input is not an array) where each element is the the natural logarithm of the corresponding element of the input array

See also

- [LOG10](#)
- [LogN](#)

2.246. LogN

Full Name:	herschel.ia.numeric.toolbox.basic.LogN
Type:	Java Class - 
Import:	from herschel.ia.numeric.toolbox.basic import LogN

Description

Gives the logarithm with base N of a number or a numeric array: $y = \text{LOG}_n(x)$.

Example

Example 1: Apply LogN on a Int1d
<pre>x=2**Int1d.range(6) # [1,2,4,8,16,32] print Int1d(ROUND(LogN(2)(x))) # [0,1,2,3,4,5] print Int1d(ROUND(x.apply(LogN(2)))) # [0,1,2,3,4,5]</pre>

API Summary

Jython Syntax
<code><y>=LogN(<base>)(<x>)</code>
Properties
any array or number x [INPUT, MANDATORY, default=no default value]
a number base [INPUT, MANDATORY, default=no default value]
boolean array or a boolean y [OUTPUT, MANDATORY, default=no default value]

Limitations

Does not work for complex values.

API details


Properties

any array or number x [INPUT, MANDATORY, default=no default value]
An array or a number
a number base [INPUT, MANDATORY, default=no default value]
The base n
boolean array or a boolean y [OUTPUT, MANDATORY, default=no default value]
Returns an array (or a number if the input is not an array) where each element is the the logarithm with base n of the corresponding element of the input array

See also

- [LOG](#)
- [LOG10](#)

2.247. Long1d


Full Name:	herschel.ia.numeric.Long1d
Type:	Java Class - 
Import:	from herschel.ia.numeric import Long1d

Description

A rectangular numeric long array of rank 1.

Numeric arrays are part of the Numeric library. An explanation on how to use them, you can find in chapter **Numeric** -> **Arrays**. (Yes, it should be a link)

2.248. Long2d


Full Name:	herschel.ia.numeric.Long2d
Type:	Java Class - 
Import:	from herschel.ia.numeric import Long2d

Description

A rectangular numeric long array of rank 2.

Numeric arrays are part of the Numeric library. An explanation on how to use them, you can find in chapter **Numeric** -> **Arrays**. (Yes, it should be a link)

2.249. Long3d


Full Name:	herschel.ia.numeric.Long3d
Type:	Java Class - 
Import:	from herschel.ia.numeric import Long3d

Description

A rectangular numeric long array of rank 3.

Numeric arrays are part of the Numeric library. An explanation on how to use them, you can find in chapter **Numeric** -> **Arrays**. (Yes, it should be a link)

2.250. Long4d


Full Name:	herschel.ia.numeric.Long4d
Type:	Java Class - 
Import:	from herschel.ia.numeric import Long4d

Description

A rectangular numeric long array of rank 4.

Numeric arrays are part of the Numeric library. An explanation on how to use them, you can find in chapter **Numeric** -> **Arrays**. (Yes, it should be a link)

2.251. Long5d


Full Name:	herschel.ia.numeric.Long5d
Type:	Java Class - 
Import:	from herschel.ia.numeric import Long5d

Description

A rectangular numeric long array of rank 5.

Numeric arrays are part of the Numeric library. An explanation on how to use them, you can find in chapter **Numeric** -> **Arrays**. (Yes, it should be a link)

2.252. LorentzModel

Full Name:	herschel.ia.numeric.toolbox.fit.LorentzModel
Type:	Java Class - 
Import:	from herschel.ia.numeric.toolbox.fit import LorentzModel

Description

Lorentzian Model.

$$f(x;p) = p_0 * (p_2^2 / ((x - p_1)^2 + p_2^2))$$

p_0 = amplitude p_1 = x-shift p_2 = gamma (width)

The parameters are initialized at $\{1/\pi, 0.0, 1.0\}$ where the integral over the function equals 1.

This model is also known as Cauchy or Cauchy-Lorentz.


See [example](#)

Example

Example 1: LorentzModel

```
lorentz = LorentzModel()  
lorentz.setParameters( Double1d( [5, 4, 1] -) -)  
print lorentz( Double1d.range( 41 -) -/ 5 -)      # from [0,8]  
# ... fitter etc. see LevenbergMarquardtFitter
```

2.253. LUDecomposition

Full Name:	herschel.ia.numeric.toolbox.matrix.LUDecomposition
Alias:	LUDecomposition
Type:	Java Class - 
Import:	from herschel.ia.numeric.toolbox.matrix import LUDecomposition

Example

Example 1: # Cava, Matrix Test Report:
<pre>##### ### LUDecomposition ### ##### # ### HIPE ### # A = Double2d([[2.0, 1.0, 1.0],[4.0, --6.0, 0.0],[-2.0, 7.0, 2.0]]) print A [[2.0,1.0,1.0], [4.0,-6.0,0.0], [-2.0,7.0,2.0]] B = Double2d([[3.0],[-8.0],[10.0]]) print B [[3.0], [-8.0], [10.0]] res=B.apply(LUDecomposition(A)) print res [[1.0], [2.0], [-1.0]] #</pre>

API Summary

Jython Syntax
<pre>A=Double2d() B=Double2d() res=B.apply(LUDecomposition(A)) lud = LUDecomposition(A) res = B.apply(lud) lowerTriangularFactor = lud.l upperTrangularFactor = lud.u determinant = lud.det integerPivot = lud.pivot doublePivot = lud.doublePivot flag = lud.isSingular();</pre>

Property`Double2d A [INPUT, MANDATORY, default=no default value]`


API details

Property

`Double2d A [INPUT, MANDATORY, default=no default value]`

Input must be a Double2d or Float2d array.

2.254. ManualContourPanel

Full Name:	herschel.ia.toolbox.image.gui.ManualContourPanel
Type:	Java Class - 
Import:	from herschel.ia.toolbox.image.gui import ManualContourPanel

Description

A panel for the ManualContourTask.

A panel to serve as GUI for the ManualContourTask.

API Summary

Constructor
ManualContourPanel() The construction of a new ManualContourPanel.
Methods
JPanel getButtonPanel() Returns the button panel.
Action getRunAction() Returns the run action for this ManualContourPanel.
Action getResetAction() Returns the reset action for this ManualContourPanel.
setSiteEventHandler(SiteEventHandler handler) Setting the site event handler.
setTask(TaskApi task) Associating the task.
setVariableSelection(VariableSelection selection) Setting the variable selection
Map getSelectionMap() Returns the associated selection map.
SiteEventHandler getHandler() Returns the associated site event handler.
TaskApi getTask() Returns the associated task.
DoubleId getContourValues() Returns the contour values.
DefaultListModel getContourValuesModel() Returns the default list model for the contour values.

API details

Constructor

ManualContourPanel()
The construction of a new ManualContourPanel.
The construction of a new ManualContourPanel.

Methods

JPanel getButtonPanel()
Returns the button panel.
Returns the button panel for this ManualContourPanel.
Return
JPanel
Returns the button panel for this ManualContourPanel.

Action getRunAction()
Returns the run action for this ManualContourPanel.
Returns the run action for this ManualContourPanel.
Return
Action
Returns the run action for this ManualContourPanel.

Action getResetAction()
Returns the reset action for this ManualContourPanel.
Returns the reset action for this ManualContourPanel.
Return
Action
Returns the reset action for this ManualContourPanel.

setSiteEventHandler(SiteEventHandler handler)
Setting the site event handler.
Sets the site event handler for this ManualContourPanel to the given site event handler.
Argument
SiteEventHandler handler [INPUT, MANDATORY]

setTask(TaskApi task)
Associating the task.

setTask(TaskApi task)

Associates the given task to this ManualContourPanel.

Argument

TaskApi **task** [INPUT, MANDATORY]

setVariableSelection(VariableSelection selection)

Setting the variable selection

Sets the variable selection for this ManualContourPanel to the given variable selection.

Argument

VariableSelection **selection** [INPUT, MANDATORY]

Map [getSelectionMap\(\)](#)

Returns the associated selection map.

Returns the selection map associated with this ManualContourPanel.

Return

[Map](#)

Returns the selection map associated with this ManualContourPanel.

SiteEventHandler [getHandler\(\)](#)

Returns the associated site event handler.

Returns the site event handler associated with this ManualContourPanel.

Return

SiteEventHandler

Returns the site event handler associated with this ManualContourPanel.

TaskApi [getTask\(\)](#)

Returns the associates task.

Returns the task associated with this ManualContourPanel.

Return

TaskApi

Returns the task associated with this ManualContourPanel.

Double1d [getContourValues\(\)](#)

Returns the contour values.

Returns the contour values for this ManualContourPanel.

Return

[Double1d](#)

Returns the contour values for this ManualContourPanel.

[DefaultListModel](#) `getContourValuesModel()`

Returns the default list model for the contour values.


Returns the default list model for the contour values for this ManualContourPanel.

Return

[DefaultListModel](#)

Returns the default list model for this contour values for this ManualContourPanel.

2.255. ManualContourTask

Full Name:	herschel.ia.toolbox.image.ManualContourTask
Type:	Java Task - 
Import:	from herschel.ia.toolbox.image import ManualContourTask

Description

A Task for making contours.

A Task for making an ImageContour for a given image for a given list of contour values.

API Summary


Properties
Image image [INPUT, MANDATORY, default=No default value]
DoubleId values [INPUT, MANDATORY, default=No default value]
Color[] colors [INPUT, OPTIONAL, default=No default value]
ImageContour contours [OUTPUT, MANDATORY, default=No default value]

API details

Properties

Image image [INPUT, MANDATORY, default=No default value]	The image.
DoubleId values [INPUT, MANDATORY, default=No default value]	The contour values.
Color[] colors [INPUT, OPTIONAL, default=No default value]	The colors to visualise the contours.
ImageContour contours [OUTPUT, MANDATORY, default=No default value]	The ImageContour.

2.256. ManyToOneSpectrumTask

Full Name:	herschel.ia.toolbox.spectrum.ManyToOneSpectrumTask
Type:	Java Task - 
Import:	from herschel.ia.toolbox.spectrum import ManyToOneSpectrumTask

Description

Abstract base class for tasks processing { @link SpectrumContainer} data structures or arrays of such by typically applying the processing on a suitable selection of point spectra included in the containers.

Various selection mechanisms are in place that allow selecting specific individual spectra for the processing.

This family of operations typically reduces the spectra to a single spectrum, hence we call it a 'many-to-one operation'. Typical example is the average.

API Summary

Properties
Object ds [INPUT, OPTIONAL, default=no default value.]
Object selection [INPUT, OPTIONAL, default=None.]
PyDictionary Map< > selection lookup [INPUT, OPTIONAL, default=no default value.]
PyList selection index [INPUT, OPTIONAL, default=No default value.]
Object segments [INPUT, OPTIONAL, default=no default value.]
Boolean overwrite [INPUT, OPTIONAL, default=False.]
String variant [INPUT, OPTIONAL, default=no default value.]
SpectrumContainer result [OUTPUT, OPTIONAL, default=no default value]

API details

Properties


Object ds [INPUT, OPTIONAL, default=no default value.]
Input data to be processed by the task. Several types are possible: <ul style="list-style-type: none"> • SpectrumContainer • Array of SpectrumContainer, i.e. SpectrumContainer[] • List of SpectrumContainer, i.e. List • Any product with implementations of SpectrumContainer inside.
Object selection [INPUT, OPTIONAL, default=None.]
Specify what point spectra to restrict to. Different ways to specify these selections are possible:

Object selection [INPUT, OPTIONAL, default=None.]
<ul style="list-style-type: none"> • Specify a list of indices (in jython) of the point spectra for which the processing should be applied. • Use a dictionary (in jython) to specify a selection model (lookup for attributes, filter attributes to be included in ranges or intervals). • Pass a string representation of a jython list or a jython dictionary. The string will be parsed and the processing delegated to the two cases above. • Pass any java instance that implements the SelectionModel interface (for the advanced user).
PyDictionary Map> selection_lookup [INPUT, OPTIONAL, default=no default value.]
Specify a PyDictionary with keys given by the attribute name to look up and a list of admissible values. Behind the scenes, a DiscreteValueSelectionModel is instantiated.
PyList selection_index [INPUT, OPTIONAL, default=No default value.]
Specify a PyList with the indices of the point spectra to be considered.
Object segments [INPUT, OPTIONAL, default=no default value.]
Specify what segments to restrict to. There are two options available: <ul style="list-style-type: none"> • Either pass an instance of a SegmentSelection which gives the information on what segments for each point spectrum included in the container. • Specify a PyList of segment indices.
Boolean overwrite [INPUT, OPTIONAL, default=False.]
Specify whether the input data container can be reused - the values found therein is overwritten.
String variant [INPUT, OPTIONAL, default=no default value.]
Specify the variant of processing mode you would like to run (including flags / weights / ...). HCSS defaults include: "flux" / "flux-wave" / "flux-wave-weight" / "flux-flag-wave" / "flux-weight-flag-wave"
SpectrumContainer result [OUTPUT, OPTIONAL, default=no default value]
Result object containing the results of the operation applied.

History

- 2009-05-20 - meli: Initial.

2.257. MapContext

Full Name:	herschel.ia.pal.MapContext
Type:	Java Class - 
Import:	from herschel.ia.pal import MapContext

Description

Groups products (or other Contexts that in turn group products) in a

map-like structure. Products grouped in a MapContext can be subsequently retrieved by key, through the 'refs' method. Note: users may be confused as to why there is a need to have to go through a 'refs' method to add or access products from a MapContext. This is due to a technical limitation in the design which will be addressed in due course.

Examples

Example 1: Adding a product to a MapContext product = Product() ref =

```
storage.save(product) # the ref is a ProductRef object
mapcontext = MapContext()
mapcontext.refs.put("john", ref)
```

Example 2: Getting a product from a MapContext ref_john =

```
mapcontext.refs.get("john") product = ref_john.product
```

Example 3: Saving a MapContext to ProductStorage (same way as any other

```
product) ref_context = storage.save(mapcontext)
```

API Summary

Method
Map getRefs() get the 'map' of ProductRefs stored. From this 'map', you can put

API details

Method

Map getRefs()
get the 'map' of ProductRefs stored. From this 'map', you can put products into the MapContext, or retrieve products by key.
Return
Map
A map of product refs.
Examples
Putting a product into the the MapContext ref =

Map `getRefs()`

```
storage.save(product) # the ref is a ProductRef object
storage.save(product) # the ref is a ProductRef object
mapcontext.refs.put("john", ref)
```


Getting the product from the MapContext with key "john"

```
ref_john = lcontext.refs.get("john")
```

See also

- [???](#)

2.258. MatrixMultiply

Full Name:	herschel.ia.numeric.toolbox.matrix.MatrixMultiply
Type:	Java Class - 
Import:	from herschel.ia.numeric.toolbox.matrix import MatrixMultiply

Description

Performs matrix multiplication of numeric or logical matrices.

Examples

Example 1: Different syntax forms

```
# spelled out:
c=MatrixMultiply(b)(a)
c=a.apply(MatrixMultiply(b))
#
# reusing an instance of a matrix-multiplier:
f=MatrixMultiply(b)
c=f(a)
c=a.apply(f)
```

Example 2: Matrix multiplication examples

```
A = Int2d([[1,2,3], [2,3,4]])
B = Int2d([[1,2], [2,3], [3,4]])
X = Int1d([1,2])
Y = Int1d([1,2,3])
#
print A.apply(MatrixMultiply(B))
[[14, 20], [20, 29]]
print X.apply(MatrixMultiply(A))
[5, 8, 11]
print A.apply(MatrixMultiply(Y))
[14, 20]
```

API Summary

Jython Syntax

```
<y>=MatrixMultiply(<b>)(<a>)
```

Properties

[Array1dData](#) or [Array2dData](#) **a** [INPUT, MANDATORY, default=no default value]

[Array1dData](#) or [Array2dData](#) **b** [INPUT, MANDATORY, default=no default value]

Miscellaneous

Complex arrays are not supported yet.

API details

Properties


<code>Array1dData</code> or <code>Array2dData</code> a [INPUT, MANDATORY, default=no default value]
--

Input must be a rank one or rank two array of any type, but String or Complex.
--

<code>Array1dData</code> or <code>Array2dData</code> b [INPUT, MANDATORY, default=no default value]
--

Input must be an array of the same type as 'a'. If 'a' has rank one, 'b' must have rank two. If 'b' has rank one, 'a' must have rank two.

2.259. MatrixSolve

Full Name:	herschel.ia.numeric.toolbox.matrix.MatrixSolve
Type:	Java Class - 
Import:	from herschel.ia.numeric.toolbox.matrix import MatrixSolve

Description

Solves systems of linear equations of type $A x = b$.

A system of equations is a collection of equations that you deal with all together at once.

Example

Example 1: Apply MatrixSolve

```
# Solve
# -| 1  2 -| -|x1|  -| 8|
# -|      -| -|  -| = -|  -|
# -| 3  4 -| -|x2|  -|18|
#
A=Float2d([ [1,2],[3,4] -])
b=Double1d([8.0,18.0])
print MatrixSolve(b)(A) # [2.0,3.0]
print A.apply(MatrixSolve(b))
```

API Summary

Jython Syntax

```
<x>=MatrixSolve(<b>)(<A>)
```

Properties

[any matrix **A** \[INPUT, MANDATORY, default=no default value\]](#)

[Double1d **b** \[INPUT, MANDATORY, default=no default value\]](#)

Miscellaneous

Does not work for complex matrices.

API details

Properties


[any matrix **A** \[INPUT, MANDATORY, default=no default value\]](#)

Any matrix

[Double1d **b** \[INPUT, MANDATORY, default=no default value\]](#)

Input must be a Double1d array.

2.260. MAX

Full Name:	herschel.ia.numeric.toolbox.basic.Max
Alias:	MAX
Type:	Java Class - 
Import:	from herschel.ia.numeric.toolbox.basic import Max

Description

Yields the numerically largest element in the input array.

Returns a scalar of the same type as the type of the input array.

Example

Example 1: Apply MAX on a Int2d
<pre>x=Int2d([[1,2], [-1,3] -]) print MAX(x) # 3 print MAX(x, 0) # [1,3] print MAX(x, 1) # [2,3]</pre>

API Summary

Jython Syntax
<code><y>=MAX(<x>, [, <dim>])</code>
Properties
any scalar array x [INPUT, MANDATORY, default=no default value]
integer dim [INPUT, MANDATORY, default=no default value]
float or double array y [OUTPUT, MANDATORY, default=no default value]

API details


Properties

any scalar array x [INPUT, MANDATORY, default=no default value]
The input array can be any scalar array.
integer dim [INPUT, MANDATORY, default=no default value]
The dimension to compute the calculation along
float or double array y [OUTPUT, MANDATORY, default=no default value]
Returns a scalar of the same type as the type of the input array.

See also

- [MIN](#)

2.261. MEAN

Full Name:	herschel.ia.numeric.toolbox.basic.Mean
Alias:	MEAN
Type:	Java Class - 
Import:	from herschel.ia.numeric.toolbox.basic import Mean

Description

Yields the average value of the elements in the input array.

Example

Example 1: Apply MEAN on a Float1d

```
x=Float1d([1,3,2,3,4])
print MEAN(x) # 2.6
```

API Summary

Jython Syntax

```
<y>=MEAN(<x>)
```

Properties

[any scalar array **x** \[INPUT, MANDATORY, default=no default value\]](#)

[double **y** \[OUTPUT, MANDATORY, default=no default value\]](#)

API details

Properties

any scalar array **x [INPUT, MANDATORY, default=no default value]**

The input array integral or floating-point arrays; the former implicitly transformed to a double array.


double **y [OUTPUT, MANDATORY, default=no default value]**

Returns a double

See also

- [MEDIAN](#)
- [STDDEV](#)
- [VARIANCE](#)

2.262. MeanSmoothingTask

Full Name:	herschel.ia.toolbox.image.MeanSmoothingTask
Type:	Java Task - 
Import:	from herschel.ia.toolbox.image import MeanSmoothingTask

Description

A Task to smooth an image by applying a mean filter.

A Task to smooth an image using the average (mean) filter. The average filter computes the sum of all pixel values in the filter window and then divides the sum by the number of pixels in the filter window. In order to filter pixels located near the edges of the image.

API Summary


Properties
Image image [INPUT, MANDATORY, default=No default value]
Integer width [INPUT, MANDATORY, default=Default value : Integer(3)]
Image smoothed [OUTPUT, MANDATORY, default=No default value]

API details

Properties

Image image [INPUT, MANDATORY, default=No default value]
The image before smoothing.
Integer width [INPUT, MANDATORY, default=Default value : Integer(3)]
The width of the filter window.
Image smoothed [OUTPUT, MANDATORY, default=No default value]
The image after smoothing.

2.263. MedianAbsoluteDeviation

Full Name:	herschel.ia.numeric.toolbox.basic.MedianAbsoluteDeviation
Type:	Java Class - 
Import:	from herschel.ia.numeric.toolbox.basic import MedianAbsoluteDeviation

Description

The median standard deviation is one of several ways to estimate an error of the median.

Algorithm:

For an array **data** the algorithm calculates the new array $| \text{data}[i] - \text{median}(\text{data}) |$ for all values i ($|$ indicates the absolute, positive value of the difference). The algorithm returns the median of the resulting array.

Example

Example 1: apply MedianAbsoluteDeviation to an Int1d array

```
data = Int1d.range(9)
median = MEDIAN(data)
dev = data.apply( MedianAbsoluteDeviation(median) - )
```

API Summary

Properties
any 1-5d array of integral type x [INPUT, MANDATORY, default=no default value]
the median of the input array x median [INPUT, MANDATORY, default=no default value]


API details

Properties

```
any 1-5d array of integral type x [INPUT, MANDATORY, default=no
default value]
```

```
the median of the input array x median [INPUT, MANDATORY,
default=no default value]
```

2.264. MEDIAN

Full Name:	herschel.ia.numeric.toolbox.basic.Median
Alias:	MEDIAN
Type:	Java Class - 
Import:	from herschel.ia.numeric.toolbox.basic import Median

Description

Yields the central value of the elements in integral and floating point arrays.

If the array contains any NaN value, NaN is returned.

Example

Example 1: Apply MEDIAN on a Float1d
<pre>x=Float1d([1,3,2,3,4]) print MEDIAN(x) # 3.0 x=Float1d([1,Double.NaN,2,3,4]) print MEDIAN(x) # NaN #To remove NaN values (original array is not modified): print MEDIAN(NAN_FILTER(x)) #2.5 (MEDIAN is applied on [1,2,3,4] array)</pre>

API Summary

Jython Syntax
<y>=MEDIAN(<x>)
Properties
any scalar array x [INPUT, MANDATORY, default=no default value]
double y [OUTPUT, MANDATORY, default=no default value]

API details


Properties

any scalar array x [INPUT, MANDATORY, default=no default value]
The input array integral or floating-point arrays; the former implicitly transformed to a double array.
double y [OUTPUT, MANDATORY, default=no default value]
Returns the central value of the elements in integral and floating point arrays. NaN is returned if the array contains any NaN value.

See also

- [MEAN](#)
- [STDDEV](#)
- [VARIANCE](#)

2.265. MedianSmoothingTask

Full Name:	herschel.ia.toolbox.image.MedianSmoothingTask
Type:	Java Task - 
Import:	from herschel.ia.toolbox.image import MedianSmoothingTask

Description

A Task to smooth an image by applying a median filter.

A Task to smooth an image using the median filter. The median filter computes the median of all pixel values in the filter window.

API Summary


Properties
Image image [INPUT, MANDATORY, default=No default value]
Integer width [INPUT, MANDATORY, default=Default value : Integer(3)]
Image smoothed [OUTPUT, MANDATORY, default=No default value]

API details

Properties

Image image [INPUT, MANDATORY, default=No default value]
The image before smoothing.
Integer width [INPUT, MANDATORY, default=Default value : Integer(3)]
The width of the filter window.
Image smoothed [OUTPUT, MANDATORY, default=No default value]
The image after smoothing.

2.266. MetaQuery

Full Name:	herschel.ia.pal.query.MetaQuery
Type:	Java Class - 
Import:	from herschel.ia.pal.query import MetaQuery

Description

Meta data query formulates a query on the meta data of a Product.

Typically this type of query is slower than an Attribute Query, but faster than a full query on the Product Access Layer.


Example

Example 1: Example of an query on meta data <code>q=MetaQuery(MyProduct.class,"p",</code>
<code>"p.meta['creator'].value == -'Me'")</code>

See also

- [???](#)

2.267. MIN

Full Name:	herschel.ia.numeric.toolbox.basic.Min
Alias:	MIN
Type:	Java Class - 
Import:	from herschel.ia.numeric.toolbox.basic import Min

Description

Yields the numerically smallest element in the input array.

Returns a scalar of the same type as the type of the input array.

Example

Example 1: Apply MIN on a Int2d

```
x=Int2d( [ [1,2], [-1,3] -])
print MIN(x)      # --1
print MIN(x, 0)  # [-1, 2]
print MIN(x, 1)  # [ 1,-1]
```

API Summary

Jython Syntax

```
<y>=MIN(<x>, [ ,<dim>])
```

Properties

[any scalar array **x** \[INPUT, MANDATORY, default=no default value\]](#)

[integer **dim** \[INPUT, MANDATORY, default=no default value\]](#)

[float or double array **y** \[OUTPUT, MANDATORY, default=no default value\]](#)

API details

Properties

any scalar array **x [INPUT, MANDATORY, default=no default value]**

The input array can be any scalar array.

integer **dim [INPUT, MANDATORY, default=no default value]**

The dimension to compute the calculation along


float or double array **y [OUTPUT, MANDATORY, default=no default value]**

Returns a scalar of the same type as the type of the input array.

See also

- [MAX](#)

2.268. ModelFitTask

Full Name:	herschel.ia.toolbox.fit.ModelFitTask
Alias:	ModelFitTask
Type:	Java Task - 
Import:	from herschel.ia.toolbox.fit import ModelFitTask
Category:	generic task

Description

generic module: FitterTask

Task shell around the package herschel.ia.numeric.toolbox.fit. Both a command line and a GUI interface are provided. With the GUI interface there is more of the Fitter package accessible than on this task's command line. However not everything possible within the fitter package is accessible via this task, GUI or otherwise. Most text fields in the GUI accept either numerics or JIDE variables or JIDE expressions.

Example

Example 1: ModelFitTask
<pre> from herschel.hifi.generic.task import FitterTask # Assume that `tt` and/or `data` are DoubleI'd's # usage on command line: ft = ModelFitTask()(x=tt, y=data, modelname="GaussModel" -) ft.fittername = -"LevenbergMarquardtFitter" ft() # performs the execute method print ft.result # parameters of the fit print ft.stdev # standard deviations print ft.fittername # name of the fitter # Etcetera, etc. # use the GUI. ft = ModelFitTask() ft.gui = 1 # start the GUI # from the GUI select the data, model, fitter, properties etc and run. print ft.result # parameters of the fit print ft.stdev # standard deviations # Etcetera as before. </pre>

API Summary

Jython Syntax
see example below
Properties
NumericData x [INPUT, OPTIONAL, default=null]
DoubleArray y [INPUT, MANDATORY, default=no]
Boolean indexview [INPUT, OPTIONAL, default=true]
DoubleArray weights [INPUT, OPTIONAL, default=null]
AbstractModel model [INPUT, OPTIONAL, default=null]
String modelname [INPUT, OPTIONAL, default=no]
ArrayI'dData modelarg [INPUT, OPTIONAL, default=null]
Fitter fitter [INPUT, OPTIONAL, default=null]

Properties
String fittername [INPUT, OPTIONAL, default=no]
DoubleId result [OUTPUT, OPTIONAL, default=n/a]
DoubleId parameters [OUTPUT, OPTIONAL, default=n/a]
Double chisq [OUTPUT, OPTIONAL, default=n/a]
DoubleId prior [INPUT, OPTIONAL, default=null]
Double scale [OUTPUT, OPTIONAL, default=n/a]
IterationPlotable plotter [INPUT, OPTIONAL, default=null]
Integer plotfreq [INPUT, OPTIONAL, default=10]
Integer printfreq [INPUT, OPTIONAL, default=0]
Boolean auto [INPUT, OPTIONAL, default=true]
Double fixed [INPUT, OPTIONAL, default=1.0]
Double mixed [INPUT, OPTIONAL, default=0.0]
Double tolerance [INPUT, OPTIONAL, default=0.01]
Integer iterations [INPUT, OPTIONAL, default=10000]
Double temperature [INPUT, OPTIONAL, default=0.0]
Double cooling [INPUT, OPTIONAL, default=0.95]
Integer tempsteps [INPUT, OPTIONAL, default=100]
DoubleId initialpars [INPUT, OPTIONAL, default=from model]
DoubleId highlimits [INPUT, OPTIONAL, default=null]
DoubleId lowlimits [INPUT, OPTIONAL, default=null]
IntId keepfixed [INPUT, OPTIONAL, default=null]
DoubleId fixedvalues [INPUT, OPTIONAL, default=null]
Boolean gui [INPUT, OPTIONAL, default=false]

Limitations

Not everything which is possible using the package directly is accessible via this task.

Miscellaneous

In case of problems look at the [trouble shooting](#) section.

API details

Properties

NumericData x [INPUT, OPTIONAL, default=null]
The independent variable(s) of the fit problem. If not provided, it will use indices ranging from 0 to len(y). If y is more-dimensional the proper 2-d indices are provided.
DoubleArray y [INPUT, MANDATORY, default=no]
The data to be fitted. It can be more-dimensional. E.g. a 2-dimensional map.
Boolean indexview [INPUT, OPTIONAL, default=true]
Applicable parameters appear in the same order as in the y DoubleArray This parameter is only active when y has 2 dimensions or more.


DoubleArray weights [INPUT, OPTIONAL, default=null]
The weights to be used in the fit. Can be more-dimensional. When provided it should be the same size as parameter y.
AbstractModel model [INPUT, OPTIONAL, default=null]
The model to be fitted. One of "model" or "modelname" is MANDATORY
String modelname [INPUT, OPTIONAL, default=no]
The name of the model to be fitted. One of "model" or "modelname" is MANDATORY
ArrayldData modelarg [INPUT, OPTIONAL, default=null]
Arguments, if any, needed in the Constructor of the model.
Fitter fitter [INPUT, OPTIONAL, default=null]
The fitter to be used. One of "fitter" or "fittername" is MANDATORY
String fittername [INPUT, OPTIONAL, default=no]
The name of the fitter to be used. One of "fitter" or "fittername" is MANDATORY
Doubleld result [OUTPUT, OPTIONAL, default=n/a]
The parameters of the fit. Also the default result of the task.
Doubleld parameters [OUTPUT, OPTIONAL, default=n/a]
The parameters of the fit. (same as result)
Double chisq [OUTPUT, OPTIONAL, default=n/a]
Chi-squared of the fit.
Doubleld prior [INPUT, OPTIONAL, default=null]
Prior ranges for the parameters, needed when the evidence is requested. When noise scaling is set to auto=true (default), the prior needs 1 extra value as prior for the noise scale.
Double scale [OUTPUT, OPTIONAL, default=n/a]
Noise scale of the data (when auto or mixed is selected)
IterationPlotable plotter [INPUT, OPTIONAL, default=null]
make a plot of the fit at each "plotfreq" iteration. A plotter is provided at <code>herschel.ia.toolbox.fit.IterationPlotter</code>
Integer plotfreq [INPUT, OPTIONAL, default=10]
to be used in conjunction with plotter.
Integer printfreq [INPUT, OPTIONAL, default=0]
Report about the present parameter settings every printfreq-th iteration.
Boolean auto [INPUT, OPTIONAL, default=true]
Select automatic noise scaling. The noise level in the data is not exactly known.

Double fixed [INPUT, OPTIONAL, default=1.0]
Fixed noise scale. The noise level is known.
Double mixed [INPUT, OPTIONAL, default=0.0]
Automatic noise scaling with a minimum. The noise level is not exactly known, but a minimum level is known.
Double tolerance [INPUT, OPTIONAL, default=0.01]
Stopping criterion for iterative fitting.
Integer iterations [INPUT, OPTIONAL, default=10000]
Maximum number of iterations in iterative fitters.
Double temperature [INPUT, OPTIONAL, default=0.0]
Starting temperature in annealing amoeba fitting.
Double cooling [INPUT, OPTIONAL, default=0.95]
Cooling step in annealing amoeba fitting.
Integer tempsteps [INPUT, OPTIONAL, default=100]
Number of exploratory steps at each temperature.
DoubleId initialpars [INPUT, OPTIONAL, default=from model]
Initial parameters in iterative fitters.
DoubleId highlimits [INPUT, OPTIONAL, default=null]
Upper limits of the parameters (only in AmoebaFitter)
DoubleId lowlimits [INPUT, OPTIONAL, default=null]
Lower limits of the parameters (only in AmoebaFitter)
IntId keepfixed [INPUT, OPTIONAL, default=null]
Keep these parameters fixed. (Parameter numbering starts at 0)
DoubleId fixedvalues [INPUT, OPTIONAL, default=null]
Values at which the parameters should be kept.
Boolean gui [INPUT, OPTIONAL, default=false]
Allows to handle this task using a gui.

History

- 15-10-2006 DK

2.269. MosaicTask

Full Name:	herschel.ia.toolbox.image.MosaicTask
Type:	Java Task - 
Import:	from herschel.ia.toolbox.image import MosaicTask

Description

A Task for making mosaics.

A Task to make mosaics in a naive way (i.e. by simple co-adding).

API Summary

Properties
ArrayList images [INPUT, MANDATORY, default=No default value]
boolean oversample [INPUT, OPTIONAL, default=Default value : true]
SimpleImage mosaic [OUTPUT, MANDATORY, default=No default value]

Miscellaneous


Two blank maps are created : one to represent the total signal and one to represent the total exposure. For each pixel in each image, the pixel value is added into the total signal map and its exposure (or one) in the total exposure map (if not flagged out), taking only the overlap into account. After all pixels in each image have been mapped, the total signal map is divided by the total exposure map to produce the mosaic.

API details

Properties

ArrayList images [INPUT, MANDATORY, default=No default value]
The input images.
boolean oversample [INPUT, OPTIONAL, default=Default value : true]
Indication whether oversampling should be done.
SimpleImage mosaic [OUTPUT, MANDATORY, default=No default value]
The mosaic.

2.270. MultiplySpectrum

Full Name:	herschel.ia.toolbox.spectrum.MultiplySpectrum
Type:	Java Task - 
Import:	from herschel.ia.toolbox.spectrum import MultiplySpectrum

Description

Task for multiplying the flux data included in a spectrum container with a scalar or for multiplying two spectrum containers with each other on a spectrum-by-spectrum basis.

For the scalar mode, 'ds' and 'param' need to be specified - for the pair-wise mode, 'ds1' and 'ds2' need to be set. In the pair-wise mode, the task iterates through the two containers and computes the pair-operation. In case the size of the two containers is different, the iteration stops at the minimum size. Hence, the remaining spectra in the larger container are ignored. In case one of the datasets only contains a single spectrum, the task always refers to this single spectrum while iterating over all the spectra in the other dataset.

Different selection schemes are available for selecting the point spectra or the segments to be processed. The most simple scheme is to specify lists of point spectrum indices (`selection=[1, 3, 4, 2]`). In this case, the operation (scalar- or pair-operation) is just taken over the point spectra with corresponding indices. Instead of specifying a list of indices (which requires the knowledge of these) other recipes to specify a selection are available. Typical recipes test attribute values to match certain values or to be included in a given interval. These recipes we refer to as selection models. The different options to specify selections can be combined. Here, an AND logic is adopted.

Similarly, you can specify what segments to consider. In the most simple case, you can just specify the indices of segments to be considered (for the scalar operation `segments=[1, 3, 4]` and for the pair operation `segments1=[1, 3, 4]`, `segments2=[3, 6, 5]`). In more advanced situations you can use a `SegmentSelection` object. Note that the pairs of segments to be processed need to be consistent - otherwise the processing will fail.

You can specify whether the original datasets should be overwritten with the 'overwrite' flag. By default no data is overwritten. In the pair-wise mode, it is not always possible to overwrite the data - in particular if non-trivial segment selections are specified.

For the pair-wise mode, there are further options that can be selected by the 'variant' (e.g. whether to use flags and weights).

Example

Example 1: from Jide:

```
from herschel.ia.toolbox.spectrum import MultiplySpectrum
multiply = MultiplySpectrum()
multipliedByFactor = multiply(ds=spectra, param=2.1)
multipliedByFactor = multiply(ds=spectra, selection={"bbtype": [6031]},
param=2.1)
multipliedByFactor = multiply(ds=spectra, selection=[0,1,2,3], param=2.1)
multipliedByFactor = multiply(ds=spectra, selection={"Chopper":
([-4.4,5.9],0.2), -"bbtype": [6031]}, param=2.1)
multipliedByFactor = multiply(ds=spectra, selection={"LoFrequency":
(4000.0,5000.0)}, param=2.1)
multipliedPairWise = multiply(ds1=spectra1, ds2=spectra2)
multipliedPairWise = multiply(ds1=spectra1, ds2=spectra2, selection={"bbtype":
[6031]})
multipliedPairWise = multiply(ds1=spectra1, ds2=spectra2, selection=[0,1,2,3])
multipliedPairWise = multiply(ds1=spectra1, ds2=spectra2, selection={"Chopper":
([-4.4,5.9],0.2), -"bbtype": [6031]})
```

Example 1: from Jide:

```
multipliedPairWise = multiply(ds1=spectral1, ds2=spectra2,
selection={"LoFrequency":(4000.0,5000.0)})
```

API Summary

Properties
SpectrumContainer ds [INPUT, OPTIONAL, default=no default value.]
Double param [INPUT, OPTIONAL, default=no default value.]
Object selection [INPUT, OPTIONAL, default=None.]
PyDictionary Map< selection_lookup [INPUT, OPTIONAL, default=no default value.]
PyList selection_index [INPUT, OPTIONAL, default=No default value.]
Boolean overwrite [INPUT, OPTIONAL, default=False.]
SpectrumContainer ds1 [INPUT, OPTIONAL, default=no default value.]
SpectrumContainer ds2 [INPUT, OPTIONAL, default=no default value.]
Object segments [INPUT, OPTIONAL, default=no default value.]
Object segments1 [INPUT, OPTIONAL, default=no default value.]
Object segments2 [INPUT, OPTIONAL, default=no default value.]
String variant [INPUT, OPTIONAL, default=no default value.]
SpectrumContainer result [OUTPUT, OPTIONAL, default=no default value]

API details

Properties

SpectrumContainer ds [INPUT, OPTIONAL, default=no default value.]
Input container to be processed by the task in case the task is used as a scalar operation.
Double param [INPUT, OPTIONAL, default=no default value.]
The scalar parameter to be considered when the task is used as scalar operation.
Object selection [INPUT, OPTIONAL, default=None.]
Specify what point spectra the operation should be applied to. Different ways to specify these selections are possible: <ul style="list-style-type: none"> • Specify a list of indices (in jython) of the point spectra for which the add should be applied. • Use a dictionary (in jython) to specify a selection model (lookup for attributes, filter attributes to be included in ranges or intervals). • Pass a string representation of a jython list or a jython dictionary. The string will be parsed and the processing delegated to the two cases above. • Pass any java instance that implements the SelectionModel interface (for the advanced user).

PyDictionary|Map selection_lookup [INPUT, OPTIONAL, default=no default value.]

Specify a PyDictionary with keys given by the attribute name to look up and a list of admissible values. Behind the scenes, a DiscreteValueSelectionModel is instantiated.

PyList selection_index [INPUT, OPTIONAL, default=No default value.]

Specify a PyList with the indices of the point spectra to be considered.

Boolean overwrite [INPUT, OPTIONAL, default=False.]

Specify whether the input data container can be reused - the values found therein is overwritten.

SpectrumContainer ds1 [INPUT, OPTIONAL, default=no default value.]

First input container for pair-wise operations.

SpectrumContainer ds2 [INPUT, OPTIONAL, default=no default value.]

Second input container for pair-wise operations.

Object segments [INPUT, OPTIONAL, default=no default value.]

Specify what segments the operation should be applied to. There are two options available:

- Either pass an instance of a SegmentSelection which gives the information on what segments for each point spectrum included in the container.
- Specify a PyList of segment indices.

Object segments1 [INPUT, OPTIONAL, default=no default value.]

Specify the segment selection to be associated with 'ds1'.

Object segments2 [INPUT, OPTIONAL, default=no default value.]

Specify the segment selection to be associated with 'ds2'.

String variant [INPUT, OPTIONAL, default=no default value.]

Specify the variant of processing mode you would like to run (including flags / weights / ...). HCSS defaults include: "flux" / "flux-wave" / "flux-wave-weight" / "flux-flag-wave" / "flux-weight-flag-wave"

SpectrumContainer result [OUTPUT, OPTIONAL, default=no default value]

Result object containing the results of the operation applied.

See also


- [SpectrumTask](#)

History

- 2007-08-17 - meli: initial.

- 2008-03-16 - meli: major refactoring of the toolbox.
- 2008-10-03 - meli: segments, javadoc

2.271. NAN_FILTER

Full Name:	herschel.ia.numeric.toolbox.basic.NaNFilter
Alias:	NAN_FILTER
Type:	Java Class - 
Import:	from herschel.ia.numeric.toolbox.basic import NaNFilter

Description

Returns the NaN elements of an array (in a one dimensional array) leaving the original array untouched. If no NaN element is found, the original array is returned.

This function can be used with arrays where the total number of NaN elements are lower than $2^{31}-1$.

Example

Example 1: Apply NAN_FILTER on a Double1d	
<code>x=Double1d([1,2,3])</code>	
<code>print NAN_FILTER(x)</code>	# [1,2,3] Original array
<code>x=Double1d([1,Double.NaN,3])</code>	
<code>print NAN_FILTER(x)</code>	# [1,3] New array
<code>x=Double2d([1,Double.NaN,3][10,11,12])</code>	
<code>print NAN_FILTER(x)</code>	# [1,3,10,11,12] New array

API Summary


Jython Syntax
<code><y>=NAN_FILTER(<x>)</code>
Properties
a float or double array x [INPUT, MANDATORY, default=no default value]
an array y [OUTPUT, MANDATORY, default=no default value]

API details

Properties

a float or double array x [INPUT, MANDATORY, default=no default value]
an array y [OUTPUT, MANDATORY, default=no default value]
Returns the NaN elements of an array (in a one dimensional array) leaving the original array untouched. If no NaN element is found, the original array is returned.

2.272. NearestNeighborInterpolator

Full Name:	herschel.ia.numeric.toolbox.interp.NearestNeighborInterpolator
Alias:	NearestNeighborInterpolator
Type:	Java Class - 
Import:	from herschel.ia.numeric.toolbox.interp import NearestNeighborInterpolator

Description

Given a set of knots (x/y data), this function computes values at arbitrary positions by using the y value associated with the knot closest to the given x position. This can only be applied to numeric arrays of rank 1.

For positions centered exactly between 2 knots, the y value from the smaller knot will be used.

Example

Example 1: Create and apply a NearestNeighborInterpolator on a Double1d

```
# create some knots
x=Double1d.range(10) # [0.0,1.0,2.0,3.0,4.0,5.0,6.0,7.0,8.0,9.0]
f=NearestNeighborInterpolator(x,SQUARE(x))
u=Double1d([1,1.1,2,2.6,3,3.9])
print f(u) # [1.0,1.0,4.0,9.0,9.0,16.0]
```

API Summary

Jython Syntax

```
<f>=NearestNeighborInterpolator(<x>,<y>[,allowExtrapolation])
```

Properties

[Double1d](#) **x** [INPUT, MANDATORY, default=no default value]

[Double1d](#) **y** [INPUT, NOT_MANDATORY, default=false]

[boolean](#) **allowExtrapolation** [INPUT, NOT_MANDATORY, default=false]

API details

Properties

[Double1d](#) **x** [INPUT, MANDATORY, default=no default value]

The knots are Double1d

[Double1d](#) **y** [INPUT, NOT_MANDATORY, default=false]

The knots are Double1d

[boolean](#) **allowExtrapolation** [INPUT, NOT_MANDATORY, default=false]


Extrapolation is not allowed by default. Boolean.TRUE allows extrapolation.

See also

- [CubicSplineInterpolator](#)

- [LinearInterpolator](#)

2.273. NearestNeighbourProjectionTask

Full Name:	herschel.ia.toolbox.spectrum.projection.NearestNeighbourProjectionTask
Type:	Java Task - 
Import:	from herschel.ia.toolbox.spectrum.projection import NearestNeighbourProjection-Task
Category:	task

Description

NearestNeighbourProjectionTask

Creates a spectral cube by converting an image grid's pixels into world coordinates and assigning the nearest spectral data to each pixel.

Spectral Data

Three types of spectral data may be used as input:

1. 3D arrays of corresponding flux, ra, dec, wave, flag, and error data
2. A SpectralSimpleCube
3. A list of SpectralSimpleCube(s)

Target Grid Data is projected onto a target grid. The shape of the spectral cube output is determined by the target grid. If not provided by the user, a target grid may be calculated by the `targetGrid` method directly from the RA and DEC data.

Examples

Example 1: How to create a spectral cube.

```
# Given:
# 1. 3D arrays of flux, ra, and dec input (with variable names flux, ra and
# dec).
# 2. A target image (a TargetGrid object) (with variable name target).
# Create a task.
nnTask = NearestNeighbourProjectionTask()
# Execute the task and create spectral cube.
spectralCube = nnTask(flux = flux, ra = ra, dec = dec, target = target)
```

Example 2: How to create a target grid.

```
# One may generate a target automatically by calling the targetGrid method.
# This method will use the directions defined by 3D arrays of ra and dec to
# create a target
# that encloses the data. The units of RA, DEC, and WAVE will be set to
# defaults if no units
# are provided.
# Create image target grid.
target = nnTask.targetGrid(ra,dec,wave)
```

Example 3: How to create a spectral cube from a list of SpectrometerDetectorSpectrum's.

```
# Given:
# 1. sdsList, an array of SpectrometerDetectorSpectrum objects.
```

Example 3: How to create a spectral cube from a list of SpectrometerDetectorSpectrum's.

```
# 2. detectorArray, a string identifying the detector array to process ("SLW"
or -"SSW").
# Create a SpirePreprocessedCube with SpirePreprocessCubeTask.
task = SpirePreprocessCubeTask()
ppCube = task(sdsList)
flux = ppCube.getFlux(detectorArray)
error = ppCube.getError(detectorArray)
flag = ppCube.getFlag(detectorArray)
ra = ppCube.getRa(detectorArray)
dec = ppCube.getDec(detectorArray)
wave = ppCube.getWave()
# Create and configure NearestNeighbourProjectionTask
nnTask = NearestNeighbourProjectionTask()
nnTask.setWaveUnit(ppCube.getWaveUnit())
nnTask.setFluxUnit(ppCube.getFluxUnit())
nnTask.setRaUnit(ppCube.getRaUnit())
nnTask.setDecUnit(ppCube.getDecUnit())
nnTask.setMetadata(ppCube.getMeta())
# Create target grid
targetGrid = nnTask.targetGrid(ra, dec, wave)
# Execute the task and create spectral cube.
nnCube = nnTask(flux = flux, error = error, flag = flag, ra = ra, dec = dec,
target = targetGrid)
```

API Summary

Properties
SpectralSimpleCube cube [INPUT, OPTIONAL, default=no default value]
SpectralSimpleCube[] cubeList [INPUT, OPTIONAL, default=no default value]
Double3d dec [INPUT, OPTIONAL, default=no default value]
Double3d ra [INPUT, OPTIONAL, default=no default value]
Double3d error [INPUT, OPTIONAL, default=no default value]
Flag flag [INPUT, OPTIONAL, default=no default value]
Double3d flux [INPUT, OPTIONAL, default=no default value]
TargetGrid target [INPUT, MANDATORY, default=no default value]
MetaData meta [INPUT, OPTIONAL, default=no default value]
Unit fluxUnit [INPUT, OPTIONAL, default=no default value]
SpectralSimpleCube spectralSimpleCube [OUTPUT, no default value, default=no default value]

API details

Properties

SpectralSimpleCube cube [INPUT, OPTIONAL, default=no default value]
SpectralSimpleCube element to be projected
SpectralSimpleCube[] cubeList [INPUT, OPTIONAL, default=no default value]
List of SpectralSimpleCube elements to be projected

Double3d dec [INPUT, OPTIONAL, default=no default value]
Declination values corresponding to flux values, having dimensions [Wavenumber,X,Y]. Wavenumber is a spectral dimension. X and Y are spatial dimensions.
Double3d ra [INPUT, OPTIONAL, default=no default value]
Right ascension values corresponding to flux, having dimensions [Wavenumber,X,Y]. Wavenumber is a spectral dimension. X and Y are spatial dimensions.
Double3d error [INPUT, OPTIONAL, default=no default value]
Error on the flux, having dimensions [Wavenumber, X, Y]. Wavenumber is a spectral dimension. X and Y are spatial dimensions.
Flag flag [INPUT, OPTIONAL, default=no default value]
Flag indicating which spectra are invalid, having dimensions [Wavenumber,X,Y]. Wavenumber is a spectral dimension. X and Y are spatial dimensions.
Double3d flux [INPUT, OPTIONAL, default=no default value]
Flux data having dimensions [Wavenumber, X,Y]. Wavenumber is a spectral dimension. X and Y are spatial dimensions.
TargetGrid target [INPUT, MANDATORY, default=no default value]
The target coordinates as a world coordinate system with three axes (two spatial and one spectral).
MetaData meta [INPUT, OPTIONAL, default=no default value]
Metadata to propagate to the SpectralSimpleCube. Regardless of whether this parameter is provided, the task will create the mandatory metadata parameters (e.g. "creator").
Unit fluxUnit [INPUT, OPTIONAL, default=no default value]
Flux unit to set in the SpectralSimpleCube output.
SpectralSimpleCube spectralSimpleCube [OUTPUT, no default value, default=no default value]
The resulting projected cube.

See also


- [???](#)
- [???](#)

History

- 2009-04-22 - CM: [SPIRE SCR-1474] First version in order to meet the requirements of the SCR
- 2009-08-25 - PK: [SPIRE SCR-1474] Updated for new SpectralProjectionAlgorithm interface (see [HCSS-7600]). Implemented project method for 3d arrays. Implemented targetGrid methods for 3d arrays.
- 2009-09-17 - DS: [SPIRE ACT-1976] Added metadata entry to output; Name:"level", Value:"20".

- 2009-09-24 - DS: [SPIRE SPR-1996] Added parameter "meta" to task to allow metadata propagation. Added metadata propagation.
- 2009-10-24 - DS: [SPIRE SPR-1996] Added unit propagation.
- 2009-11-20 - DS: [SPIRE SCR-1474] Added extrapolation to project methods. Added new project methods. Added default target grid method.
- 2009-12-02 - DS: [HCSS SPR-8906] Error handling improvement.
- 2010-01-12 - DS: [HCSS SPR-9026] Corrected WCS's CTYPE3 codes.
- 2010-01-13 - DS: [HCSS SPR-9143] Corrected beamProfile formula.
- 2010-01-19 - DS: [SPIRE-2284] Documentation updated.
- 2010-07-13 - PK: [HCSS-9832] Changed targetGrid methods to generate TargetGrid objects with negative CDELTA. Changed getMinMaxTargetGridCoordinates to search all elements of a TargetGrid for extrema.

2.274. Normalize

Full Name:	herschel.ia.numeric.toolbox.basic.Normalize
Type:	Java Class - 
Import:	from herschel.ia.numeric.toolbox.basic import Normalize

Description

This functionality normalizes sets of data.

Normalization can be done by multiplying the independent variable, by a scaling factor so that the data will have a user-specified peak value or mean value over a user-specified range of dependent variables.

Input data sets is a list of vectors containing N elements, where a single vector can be written as (x1,x2,x3,...,xN-1, y). The Normalize class provides four valid options of normalization.

-Option 1: It multiplies the y values such that the peak y value will have a user-specified constant value.

-Option 2: It normalizes the y values such that the peak y value will be same as the peak value of a user-specified "fiducial" vector set.

-Option 3: It normalizes the y values so that the mean of the y values over a user-specified x-range will have a user-specified constant value.

-Option 4: It normalizes the y values so that the mean of the y values over a user-specified x-range will be same as the mean of a user-specified "fiducial" vector set over the x-range.

Notes:

-The x-range must be specified as a range in all N-1 dimensions.

-The dimensions of input data must be same as the dimensions of the fiducial array.

-When the peak/mean value of input data set equals to zero, an error [condition] is issued.

-When the peak/mean value of input data set has different sign from user-supplied peak/mean value, an error [condition] is issued.

-When there are no data within the x-range within the fiducial data set, an error [condition] is issued.

-The routine can handle any dimension N where $N \geq 2$, and can handle all real and integer data types.

Example

Example 1: from herschel.ia.numeric import *

```
from herschel.ia.numeric.toolbox.basic import Normalize
# Perform normalizations using four valid types.
# Type 1: Normalize y values to the common peak, a user-specified constant.
# Input data array.
arr = Double2d([(0.,1.,8.,40.), (10.,10.,4.,60.), (2.,0.,4.,120.)])
# User specified peak value.
peak = 45.
# Normalize.
norm1 = Normalize(peak, Normalize.PEAK_CNST)(arr)
# Check output.
print norm1
# Output: [[0.0,1.0,8.0,15.0],[10.0,10.0,4.0,22.5],[2.0,0.0,4.0,45.0]]
# Type 2: Normalize y values to the max of a user-supplied -"fiducial" array.
# Input data array.
arr = Int2d([(0,1,8,40), (10,10,4,60), (2,0,4,120)])
```

Example 1: from herschel.ia.numeric import *

```

# User supplied fiducial array.
fiducial = Int2d([(1,1,1,10),(2,2,8,30),(4,6,10,20)])
# Normalize.
norm2 = Normalize(fiducial,Normalize.PEAK_FIDUCIAL)(arr)
# Check output.
print norm2
# Output: [[0,1,8,10],[10,10,4,15],[2,0,4,30]]
# Type 3: Normalize y values to have a user-specified constant mean value
# over a x range.
# Input data array.
arr = Double2d([(0.,1.,8.,40.),(1.,10.,4.,60.),(2.,0.,4.,120.)])
# User specified mean value.
mean = 25.
# User specified x range array.
xrange = Double2d([(0.,0.,0.),(2.,5.,10.)])
norm3 = Normalize(mean,xrange,Normalize.MEAN_CNST)(arr)
# Check output.
print norm3
# Output: [[0.0,1.0,8.0,12.5],[1.0,10.0,4.0,18.75],[2.0,0.0,4.0,37.5]]
# Type 4: Normalize y values to the mean value of a user-supplied fiducial
# array over a x range.
# Input data array.
arr = Double2d([(0.,1.,8.,40.),(10.,10.,4.,60.),(2.,0.,4.,120.)])
# User supplied fiducial array.
fiducial = Double2d([(1.,1.,1.,15.),(2.,2.,8.,45),(4.,6.,10.,25.)])
# User specified x range array.
xrange = Double2d([(0.,0.,0.),(3.,3.,10.)])
# Normalize.
norm4 = Normalize(fiducial,xrange,Normalize.MEAN_FIDUCIAL)(arr)
# Check results.
print norm4
# Output: [[0.0,1.0,8.0,15.0],[10.0,10.0,4.0,22.5],[2.0,0.0,4.0,45.0]]

```

API Summary

Jython Syntax

```
<result>=Normalize(<a>,[<b>,<type>])(<input>)
```

Properties

[its value depends on the Normalize type specified a \[INPUT, MANDATORY, default=no default value\]](#)

[its value depends on the Normalize type specified b \[INPUT, OPTIONAL, default=no default value\]](#)

[Normalization type type \[INPUT, MANDATORY, default=no default value\]](#)

[Input data sets input \[INPUT, MANDATORY, default=no default value\]](#)

[the normalized array result \[OUTPUT, MANDATORY, default=no default value\]](#)

API details

Properties

its value depends on the Normalize type specified a [INPUT, MANDATORY, default=no default value]

-Normalize.PEAK_CNST type: common peak value to use when normalizing the input values (" argument is not required). -Normalize.PEAK_FIDUCIAL type: 'fiducial' array

its value depends on the Normalize type specified a [INPUT, MANDATORY, default=no default value]

where the maximum value to use when normalizing the input values, is extracted (' argument is not required). -Normalize.MEAN_CNST type: mean value, over an 'x' range (argument ''), to be obtained when normalizing the input values (requires '' argument). -Normalize.MEAN_FIDUCIAL type: 'fiducial' array where the mean value, over an 'x' range (argument ''), to be obtained when normalizing the input values, is extracted (requires '' argument).

its value depends on the Normalize type specified b [INPUT, OPTIONAL, default=no default value]

-Normalize.MEAN_CNST type: 'x' range where the normalized input values must have the specified mean value (argument ''). -Normalize.MEAN_FIDUCIAL type: 'x' range where the [normalized input values must have the specified mean value extracted from a fiducial array \(argument ''\)](#).

Normalization type type [INPUT, MANDATORY, default=no default value]

-Normalize.PEAK_CNST: It multiplies the 'y' values such that the peak 'y' value will have a user-specified constant value -Normalize.PEAK_FIDUCIAL: It normalizes the 'y' values such that the peak 'y' value will be same as the peak value of a user-specified "fiducial" vector set. -Normalize.MEAN_CNST: It normalizes the 'y' values so that the mean of the 'y' values over a user-specified x-range will have a user-specified constant value. -Normalize.MEAN_FIDUCIAL: It normalizes the 'y' values so that the mean of the 'y' values over a user-specified x-range will be same as the mean of a user-specified "fiducial" vector set over the x-range.


Input data sets input [INPUT, MANDATORY, default=no default value]

List of vectors containing N elements, where a single vector can be written as (x1,x2,x3,...,xN-1, y)

the normalized array result [OUTPUT, MANDATORY, default=no default value]

ie: norm[(x1_1,x2_1,x3_1,...,xN-1_1, yNorm_1), (x1_2,x2_2,x3_2,...,xN-1_2, yNorm_2), ...]

2.275. NotPresent

Full Name:	herschel.ia.numeric.toolbox.basic.NotPresent
Type:	Java Class - 
Import:	from herschel.ia.numeric.toolbox.basic import NotPresent

Description

Tests whether none of the bits in specified mask are present in the elements of the input array.

The input array must be of integral type (e.g. bytes,integers) and the result is a boolean array.

Example

Example 1: Apply NotPresent on a Int1d: test if a binary 'and' comparison, item by item, with '3' (as mask) is equals to '0'

```
x=Int1d([0,1,2,3])
print NotPresent(3)(x)
#=> [ (0 & 3) == 0, (1 & 3) == 0, (2 & 3) == 0, (3 & 3) == 0 -] =
[true,false,false,false]
print x.apply(NotPresent(3)) #Another way for using NotPresent
```

API Summary

Jython Syntax

```
<y>=NotPresent(<bitmask>)(<x>)
```

Properties

[any array of integral type **x** \[INPUT, MANDATORY, default=no default value\]](#)

[any array of booleans **bitmask** \[INPUT, MANDATORY, default=no default value\]](#)

[boolean array or a boolean **y** \[OUTPUT, MANDATORY, default=no default value\]](#)

API details

Properties

any array of integral type **x [INPUT, MANDATORY, default=no default value]**

The input array must be of integral type (e.g. bytes,integers)

any array of booleans **bitmask [INPUT, MANDATORY, default=no default value]**

An array of booleans


boolean array or a boolean **y [OUTPUT, MANDATORY, default=no default value]**

Returns a boolean array (or a boolean if the input is not an array) where each element is true if the corresponding element in the input array is present, false otherwise.

See also

- [AllPresent](#)
- [AnyPresent](#)

2.276. NumberedDataset

Full Name:	herschel.ia.dataset.spectrum.NumberedDataset
Type:	Java Class - 
Import:	from herschel.ia.dataset.spectrum import NumberedDataset
Category:	Datasets

Description

NumberedDataset is an alternative for the non-existing VectorDataset.

NumberedDataset is an extension of CompositeDataset, which contains datasets identified by number. It contains an iterator over the stored datasets.

Example

Example 1: In Jide:

```
vds = NumberedDataset()           # create a NumberedDataset
s1 = Spectrum1d()                 # create some Datasets
s2 = Spectrum2d()
s3 = ArrayDataset( -)
vds.set( s1 -)                    # set s1 at number -"1"
vds.set( s2 -)                    # set s2 at number -"2"
vds.set( s3, 4 -)                 # set s3 at number -"4"
s4 = vds.get( 1 -)                # s4 equals s1
print vds.getCount()              # yields 3 (3 sets in vds)
print vds.getLastIndex()          # yields 4 (last one is at 4)
it = vds.iterator()               # iterator over the datasets.
while it.hasNext(): print it.next().__class__ # print classes
vds.remove( 2 -)                  # leaves sets at 1 and 4
vds.collapse()                   # renumbers sets to 1 and 2
vds.remove()                      # removes last one, leaves only nr 1
```


Limitations

NumberedDataset still **is** a CompositeDataset and can be addressed as such. If you do so, the special methods of NumberDataset are **not** guaranteed to work.

History

- 06-03-2006 DK.

2.277. ObservationContext

Full Name:	herschel.ia.obs.ObservationContext
Type:	Java Class - 
Import:	from herschel.ia.obs import ObservationContext

Description

An Observation Context is a container of Products applicable to an

specific observation. It provides associations to products which are specific to a single observation (e.g. Telemetry Product, and reduced data products) as well as associations to Products that are applicable to multiple observations (such as the calibration products).

Example

Example 1: from herschel.ia.obs import *

```

from herschel.share.fltdyn.time import FineTime
from herschel.ia.pal import MapContext
obs = ObservationContext()
auxContext = MapContext()
obs.auxiliary=auxContext #error
print obs.isInitialized() #0 (false)
obs.id=1L
obs.odNumber=2L
obs.instrument="HIFI"
obs.modelName="0"
obs.startTime=FineTime(1L)
obs.endTime=FineTime(2L)
print obs.isInitialized() #1 (true)
obs.auxiliary=auxContext #ok
print obs.auxiliary #{description="Unknown", meta=[type, creator,
creationDate, instrument, modelName, startDate, endDate], datasets=[],
history=None, refs=[]}
print obs.isPrepared() #0 (false)
obs.calibration = MapContext()
print obs.isPrepared() #1 (true)
obs.level['level0']=productContext #ok
print obs.level['level0'] #{description="Unknown", meta=[type, creator,
creationDate, instrument, modelName, startDate, endDate], datasets=[],
history=None, refs=[]}
print obs.isReduced() #0 (false)
obs.level['level1']=MapContext()
print obs.isReduced() #1 (true)

```

API Summary

Jython Syntax

```
<obs>=ObservationContext()
```

Property

[ObservationContext obs \[OUTPUT, MANDATORY, default=no default value\]](#)

API details

Property


<code>ObservationContext obs [OUTPUT, MANDATORY, default=no default value]</code>

Returns an empty ObservationContext

See also

- [???](#)
- [MapContext](#)
- [ProductRef](#)
- [???](#)


2.278. OpDayGenerator

Full Name:	herschel.ia.pg.od.OpDayGenerator
Type:	Java Class - 
Import:	from herschel.ia.pg.od import OpDayGenerator

History

- 19 Oct 2007: better exeption handling.
- 19 Nov 2007: add PCAL plugin

2.279. openFile

Full Name:	herschel.ia.toolbox.util.OpenFileTask
Alias:	openFile
Type:	Java Task - 
Import:	from herschel.ia.toolbox.util import OpenFileTask
Category:	task

Description

opens a variable in a viewer

Allows to open via script viewers associated to variables.

Examples

Example 1: Opening a jython script with the editor

```
openFile("/home/user/myscript.py")
```

Example 2: Opening a fits file with its default viewer

```
openFile("/home/user/product.fits")
```

API Summary

Jython Syntax

```
openFile(<variable> [, <viewer>])
```

Properties

```
String file [INPUT, MANDATORY, default=null]
```

```
String viewer [INPUT, OPTIONAL, default=null]
```

API details

Properties

```
String file [INPUT, MANDATORY, default=null]
```

The path of the file to open


```
String viewer [INPUT, OPTIONAL, default=null]
```

The ID of the viewer to open the file with

History

- 2009-02-08 - JDS: first release
- 2009-09-30 - JSS: modifiers are created through getCustomModifiers (SCR HCSS-8253)

2.280. openTask

Full Name:	herschel.ia.toolbox.util.OpenTaskTask
Alias:	openTask
Type:	Java Task - 
Import:	from herschel.ia.toolbox.util import OpenTaskTask
Category:	task

Description

opens a task in its viewer

Allows to open via script viewers associated to registered tasks.

Example

Example 1: Opening FitsReader task filling the file field with variable filename

```
filename = -"/home/user/my.fits"
openTask("fitsReader", variablename="filename")
```

API Summary

Jython Syntax

```
openTask(<taskname> [, primeinputname] [, <varname>])
```

Properties

[String taskname](#) [INPUT, MANDATORY, default=null]

[String primeinputname](#) [INPUT, OPTIONAL, default=null]

[String variablename](#) [INPUT, OPTIONAL, default=null]

API details

Properties

[String taskname](#) [INPUT, MANDATORY, default=null]

Name of the task to be opened

[String primeinputname](#) [INPUT, OPTIONAL, default=null]

Name of the prime input variable if needed to distinguish between tasks with the same name but different prime inputs


[String variablename](#) [INPUT, OPTIONAL, default=null]

Name of the variable to fill the prime input with

History

- 2009-11-03 - JDS: first release

2.281. openVariable

Full Name:	herschel.ia.toolbox.util.OpenVariableTask
Alias:	openVariable
Type:	Java Task - 
Import:	from herschel.ia.toolbox.util import OpenVariableTask
Category:	task

Description

opens a variable in a viewer

Allows to open via script viewers associated to variable types. TODO: document viewers

Example

Example 1: Opening a product variable with its default viewer

```
p = Product()
openVariable("p")
```

API Summary

Jython Syntax

```
openVariable(<variable> [, <viewer>])
```

Properties

[String variable](#) [INPUT, MANDATORY, default=null]

[String viewer](#) [INPUT, OPTIONAL, default=null]

API details

Properties

[String variable](#) [INPUT, MANDATORY, default=null]

Path of the file to be opened


[String viewer](#) [INPUT, OPTIONAL, default=null]

Viewer to open the variable with

History

- 2008-12-15 - JDS: first release
- 2008-12-16 - JDS: added optional parameter viewer

2.282. OverPlotter

Full Name:	herschel.ia.gui.explorer.table.OverPlotter
Type:	Java Class - 
Import:	from herschel.ia.gui.explorer.table import OverPlotter

Description

Overview of OverPlotter

OverPlotter is an extension of TablePlotter and allow users to overlay data on top of each other and to compare. The OverPlotter can be seen as stacking many transparent TablePlotters as layers on top of each other. Most TablePlotter features work in OverPlotter, especially when working on an individual layer. OverPlotter layers have three states, active, secondary active and inactive. Each OverPlotter layer has its own personalities. The personalities are unchanged no matter the layer is active, secondary active and inactive. \

[@jcategory] herschel.ia.gui.explorer.table;

Example

Example 1: Invoke OverPlotter in command line

```
<pre>
from herschel.ia.io.fits import FitsArchive
from herschel.ia.dataset import TableDataset
from herschel.ia.dataset import Product
from herschel.ia.gui.explorer.table import *
from herschel.share.component import *
fits=FitsArchive();
path="/home/zhang/Development/data/"
filename = "loadCurve.fits"
try:
    fits.reader = FitsArchive.HCSS_READER
    p = fits.load(path+filename)
except java.io.IOException:
    fits.reader = FitsArchive.STANDARD_READER
    p = fits.load(path+filename)
table =p.default
wm=WindowManager.getDefault()
#Load OverPlotter
overPlotter=OverPlotter(table)
wm.addWindow('test', overPlotter.component, 1)
#add a new layer
table1=table
overPlotter.object=table1
</pre>
```

API Summary

Constructors

[OverPlotter\(\)](#)

The default constructor

Constructors
OverPlotter(TableDataset tds) A constructor
Methods
JComponent getComponent() This method will return OverPlotter as a pluggable component
String getDescription()
String getName()
setObject(TableDataset data) Initiate a new instance of OverPlotter or add a new layer to the existing OverPlotter.

API details

Constructors

OverPlotter() The default constructor This constructor is used to initialize the <code>_layerCounter</code> to 0.
OverPlotter(TableDataset tds) A constructor This constructor is used to initiate an instance of TablePlotter from the command line. Argument TableDataset tds [INPUT, MANDATORY] Example - Invoke OverPlotter in command line <pre>from herchel.ia.gui.explorer.table import OverPlotter #import OverPlotter opl=OverPlotter(tbs) #dts is a TableDataset defined somewhere else</pre>

Methods

JComponent getComponent() This method will return OverPlotter as a pluggable component This method allows TablePlotter to be used as a plug-ins. The user can plug OverPlotter to his/her own applications. Return JComponent the OverPlotter as a component Example Use OverPlotter as a plug-in <pre><pre></pre>
--

JComponent `getComponent()`

```

from herschel.share.component import *
from javax.swing import *
from java.awt import *
from herschel.ia.io.fits import FitsArchive
from herschel.ia.dataset import TableDataset
from herschel.ia.dataset import Product
from herschel.ia.dataset.gui import *
from herschel.ia.gui.explorer.table import OverPlotter
fits=FitsArchive();
#Change to your data path
path="/home/zhang/Development/data/"
filename = -"loadCurve.fits"
#load the table
try:
    fits.reader = FitsArchive.HCSS_READER
    p = fits.load(path+filename)
except java.io.IOException:
    fits.reader = FitsArchive.STANDARD_READER
    p = fits.load(path+filename)
#create a TableDataset
table =p.default
#create a container to hold the controls/components
pane=JPanel()
#set the layout, there are many different kinds users can choose according
to his/her need
pane.setLayout(BoxLayout.Y_AXIS))
#add control one, a label
title = JLabel("Please see OverPlotter below")
pane.add(title)
#Add OverPlotter to the pane
overPlotter=OverPlotter(table)
pane.add(overPlotter.component)
pane.setPreferredSize(Dimension(overPlotter.component.width,
overPlotter.component.height))
#add to your application
frame =JFrame("OverPlotter as Plug-in demo");
frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE)
frame.getContentPane().add(pane, BorderLayout.CENTER)
frame.pack
frame.visible=1
</pre>

```

String `getDescription()`**Return**[String](#)

the description of the OverPlotter

String `getName()`**Return**[String](#)

the name of this explorer

setObject(TableDataset data)

Initiate a new instance of OverPlotter or add a new layer to the existing OverPlotter.

When this method is called the first time, it will initiate a new instance of OverPlotter and then assign the class variable `_activeOverPlotter` to the newly created object. When it is called again, it will add a new OverPlotter layer on to the existing OverPlotter object, `_activeOverPlotter`.

setObject(TableDataset data)**Argument**

TableDataset **data** [INPUT, MANDATORY, default=no default value]


data is a TableDataset. It will be passed as an active data structure to be plotted.

Example

Create a new OverPlotter with one layer and then add the second layer

```
<pre>
from herschel.ia.gui.explorer.table import OverPlotter
overPlotter = OverPlotter(dataset1)
overPlotter.object = dataset2
</pre>
```


2.283. PackedMask

Full Name:	herschel.ia.numeric.toolbox.mask.PackedMask
Type:	Java Class - 
Import:	from herschel.ia.numeric.toolbox.mask import PackedMask

Description

PackedMask creates and stores compressed mask array data which can be set, unset and checked.

This way no memory is wasted. Also no limit on the total bits of mask (in FixedMask, the maximum number of bits in one mask data is 64.)

Examples

Example 1: #An 2x3 2-d mask array is create by

```
mask=PackedMask("maskname", [2,3,129])
#The previous command defines a mask holding 129 bits for each element of a
2x3 array.
#However, since mask bits are stored as integers, and an integer is 32 bits
long, five
#integers (129/32 + 1) are needed to store <emphasis>at least</emphasis> 129
bits for
#each array element (four integers would stop at 128 bits). Therefore the
mask ends up
#holding 32 x 5 = 160 bits.
#The mask is stored internally as a 2x3x5 integer array, as shown by the
following command:
print mask.getMask()
#[ [ [0,0,0,0,0], [0,0,0,0,0], [0,0,0,0,0] -], [ [0,0,0,0,0], [0,0,0,0,0],
[0,0,0,0,0] -] -]
print mask.getName()
#maskname
```

Example 2: Set/Unset mask (continued from the previous example)

```
#One mask data in the array can be set by
print mask.set([1,2,3]) # The [1,2] mask data is set by bit offset 3
#[ [ [0,0,0,0,0], [0,0,0,0,0], [0,0,0,0,0] -], [ [0,0,0,0,0], [0,0,0,0,0],
[8,0,0,0,0] -] -]
#and unset by
print mask.unset([1,2,3])
#[ [ [0,0,0,0,0], [0,0,0,0,0], [0,0,0,0,0] -], [ [0,0,0,0,0], [0,0,0,0,0],
[0,0,0,0,0] -] -]
```

Example 3: Check mask (continued from the previous example)

```
#The mask data can be checked by
print mask.isSet([1,2,3])
#1
print mask.isSet([1,1,2])
#0
```

Example 4: How to set other integers (continued from the previous example)

```
# The bit offset 0-31 to set the first integer, 32-63 the second, -....
print mask.set([1,0,65]) #FixedMask can only set 64 bits.
#[ [ [0,0,0,0,0], [0,0,0,0,0], [0,0,0,0,0] -], [ [0,0,2,0,0], [0,0,0,0,0],
[0,0,0,0,0] -] -]
print mask.set([0,1,157])
```

Example 4: How to set other integers (continued from the previous example)

```
#[ [ [0,0,0,0,0], [0,0,0,0,536870912], [0,0,0,0,0] -], [ [0,0,2,0,0],
[0,0,0,0,0], [0,0,0,0,0] -] -]
```

Example 5: High dimension mask array, up to 5d:

```
mask1=PackedMask("m1", [33])
print mask1.getMask()
#[0,0]
print mask1.set([3])
#[8,0]
mask2=PackedMask("m2", [2,100])
print mask2.getMask()
#[ [0,0,0,0], [0,0,0,0] -]
print mask2.set([1,4])
#[ [0,0,0,0], [16,0,0,0] -]
mask3=PackedMask("m3", [2,3,64])
print mask3.getMask()
#[ [ [0,0], [0,0], [0,0] -], [ [0,0], [0,0], [0,0] -] -]
print mask3.set([0,1,5])
#[ [ [0,0], [32,0], [0,0] -], [ [0,0], [0,0], [0,0] -] -]
mask4=PackedMask("m4", [2,3,4,129])
print mask4.getMask()
print mask4.set([1,2,3,150])
mask5=PackedMask("m5", [2,3,4,5,129])
print mask5.getMask()
print mask5.set([1,2,3,4,150])
```


See also

- [FixedMask](#)

History

- July 2007 Original, based on MW code
- June 2009 SPR HCSS-6831. Set wrong mask data.
- Sept 2009 SCR HCSS-6832 and 6834. UM and URM.

2.284. pacs2HipeDouble3d

Full Name:	herschel.ia.toolbox.cube.pacs2HipeDouble3d
Type:	Java Task - 
Import:	from herschel.ia.toolbox.cube import pacs2HipeDouble3d

Description

A Task which reshape PACS 3 dimensional data

Example

Example 1: Rotate a cube pacs3d[Depth,Height,Width] in a cube hipe3d[Depth,Height,Width]

```
importCube(simplecube = cube, filename = -"myFile.fits")
```

API Summary


Properties
Cube simplecube [IO, MANDATORY, default=No default value]
string filename [INPUT, MANDATORY, default=no default value]

API details

Properties

Cube simplecube [IO, MANDATORY, default=No default value]
The input Cube to load
string filename [INPUT, MANDATORY, default=no default value]
The file to import

2.285. PadeModel

Full Name:	herschel.ia.numeric.toolbox.fit.PadeModel
Type:	Java Class - 
Import:	from herschel.ia.numeric.toolbox.fit import PadeModel

Description

General Pade model of arbitrary degrees in numerator and denominator.

$$f(x;p) = \sum p_n * x^n / (1 + \sum p_{num+k} * x^k)$$

where the sum in the numerator is over n running from 0 to num (inclusive) and the sum in the denominator is over k running from 1 to den (inclusive)

All parameters are initialized at 1. It is a non-linear model.


Beware of the poles where the denominator equals zero.

See [example](#)

Example

Example 1: PadeModel	
<pre>pade = PadeModel(3, 1 -) # 3rd degree polynomial print pade.getNumberOfParameters() # 5</pre>	

2.286. pause

Full Name:	herschel.ia.toolbox.util.PauseTask
Alias:	pause
Type:	Java Task - 
Import:	from herschel.ia.toolbox.util import PauseTask
Category:	task

Description

Pause the execution of jython code

The pause task is used to pause the execution of jython. When executed a debug dialog appears displaying the current local namespace in a inspector window and a new console for executing statements into the localized namespace.

Users can add/alter the values in the namespace, the values affects the code still to be executed.

Examples

Example 1: Pause the execution of jython code

```
pause()
```

Example 2: Pause the execution of my_function between the assignment of x and its printing

```
#Assignments done to x from the debugger window are reflected by the print
def my_function():
    x=100
    pause()
    print x
my_function()
```

Limitations

no Limitation

Miscellaneous

No miscellaneous


See also

- [pause](#)

History

- 2007-10-11 - NdC: first release.

2.287. pointHistoryDisplay

Full Name:	herschel.ia.toolbox.pointing.PointHistoryDisplayTask
Alias:	pointHistoryDisplay
Type:	Java Task - 
Import:	from herschel.ia.toolbox.pointing import PointHistoryDisplayTask
Category:	utility task

Description

Task for displaying graphs of pointing information

This takes a PointingProduct and produces graphs of the data contained in the product.

Example

Example 1: PointHistoryDisplayTask

```
<pre>
...
pp = pool.getProduct(urn)
phdt = PointHistoryDisplayTask()
graphs = StringId(["RAC-Time", -"RAG-Time"])
phdt(pp, graphs, -"mosaic")
...
</pre>
```

API Summary

Jython Syntax

```
PointHistoryDisplayTask()(pp, graphs, mosaic)
see examples
```

Properties

[PointingProduct](#) [PointingProduct](#) [INPUT, true, default=null]

[StringId](#) [plotPairs](#) [INPUT, true, default=null]

[String](#) [layout](#) [INPUT, true, default="mosaic"]

API details

Properties

PointingProduct [PointingProduct](#) [INPUT, true, default=null]

PointingProduct

StringId [plotPairs](#) [INPUT, true, default=null]

StringId containing strings specifying parameter pairs to be plotted. These are pairs of Strings separated by a hyphen.

Allowed Strings are


StringId plotPairs [INPUT, true, default=null]

- "Time"
- "RAC" (Commanded RA)
- "RAG" (Gyro-propagated RA)
- "RAF" (Filtered RA)
- "DecC" (Commanded Dec)
- "DecG" (Gyro-propagated RA)
- "DecF" (Filtered RA)
- "PAC" (Commanded Position angle)
- "PAG" (Gyro-propagated position angle)
- "PAF" (Filtered position angle)
- "AngVel1" (first angular velocity value)
- "AngVel2" (second angular velocity value)
- "AngVel3" (third angular velocity value)

String layout [INPUT, true, default="mosaic"]

String with value "mosaic" or "overlay" to specify how multiple plots should be organised.

2.288. Polygon

Full Name:	herschel.ia.gui.image.Polygon
Type:	Java Class - 
Import:	from herschel.ia.gui.image import Polygon

Description

A polygon shape.

A polygon shape.

API Summary

Constructors
<p>Polygon()</p> <p>The construction of a new Polygon without coordinates.</p>
<p>Polygon(double[] coords)</p> <p>The construction of a new Polygon with the given vertices</p>
<p>Polygon(double x, double y)</p> <p>The construction of a new Polygon with a single starting point.</p>
<p>Polygon(int size)</p> <p>The construction of a new Polygon with space for the given</p>
Method
<p>boolean contains(double x, double y)</p> <p>Checks whether the point with the given pixel coordinates is inside this Polygon.</p>

Miscellaneous

This class extends the existing `diva.util.java2d.Polygon2D.Double`, because the `contains()` method was implemented incorrectly there.

API details

Constructors

Polygon()
The construction of a new Polygon without coordinates.
The construction of a new Polygon without coordinates.
Polygon(double[] coords)
The construction of a new Polygon with the given vertices
in the format [x0, y0, x1, y1,...].
Argument
double[] coords [INPUT, MANDATORY]

Polygon(double x, double y)

The construction of a new Polygon with a single starting point.

Arguments

double **x** [INPUT, MANDATORY]

double **y** [INPUT, MANDATORY]

Polygon(int size)

The construction of a new Polygon with space for the given number of vertices.

Argument

int **size** [INPUT, MANDATORY]

Method

boolean contains(double x, double y)

Checks whether the point with the given pixel coordinates is inside this Polygon.

Arguments

double **x** [INPUT, MANDATORY]


double **y** [INPUT, MANDATORY]

Return

boolean

Return true if the point with the given pixel coordinates is inside this Polygon; false otherwise.

2.289. PolygonHistogramExplorer

Full Name:	herschel.ia.gui.image.PolygonHistogramExplorer
Type:	Java Class - 
Import:	from herschel.ia.gui.image import PolygonHistogramExplorer

Description

An explorer for PolygonHistogramProducts.

An explorer for PolygonHistogramProducts.

API Summary

Constructors
PolygonHistogramExplorer() The construction of a new PolygonHistogramExplorer.
PolygonHistogramExplorer(Object object) The construction of a new PolygonHistogramExplorer associated with the given object.
Methods
String getName() Returns the name.
String getDescription() Returns the description.
boolean canHandle(Class className) Checks whether this PolygonHistogramExplorer can handle objects of the given class.
setObject(Object object) Sets the object.
PolygonHistogramProduct getObject() Returns the object.
Class getVariableType() Returns the expected variable type.
JScrollPane getParameterTable() Returns the parameter table.

API details

Constructors

PolygonHistogramExplorer() The construction of a new PolygonHistogramExplorer. The construction of a new PolygonHistogramExplorer.
PolygonHistogramExplorer(Object object) The construction of a new PolygonHistogramExplorer associated with the given object.

PolygonHistogramExplorer([Object](#) object)

The construction of a new PolygonHistogramExplorer associated with the given object.

Argument

[Object](#) object [INPUT, MANDATORY]

Methods

[String](#) getName()

Returns the name.

Returns the name for this PolygonHistogramExplorer.

Return

[String](#)

Returns the name for this PolygonHistogramExplorer.

[String](#) getDescription()

Returns the description.

Returns the description for this PolygonHistogramExplorer.

Return

[String](#)

Returns the description for this PolygonHistogramExplorer.

boolean canHandle([Class](#) className)

Checks whether this PolygonHistogramExplorer can handle objects of the given class.

Returns true if this PolygonHistogramExplorer can handle objects of the given class; false otherwise.

Argument

[Class](#) className [INPUT, MANDATORY]

Return

boolean

Returns true if this PolygonHistogramExplorer can handle objects of the given class; false otherwise.

setObject([Object](#) object)

Sets the object.

Sets the object for this PolygonHistogramExplorer to the given object.

Argument

[Object](#) object [INPUT, MANDATORY]

[PolygonHistogramProduct](#) getObject()

Returns the object.

[PolygonHistogramProduct](#) getObject()

Returns the object for this PolygonHistogramExplorer.

Return

[PolygonHistogramProduct](#)

Returns the object for this PolygonHistogramExplorer.

[Class](#) getVariableType()

Returns the expected variable type.

Returns the expected variable type for this PolygonHistogramExplorer.

Return

[Class](#)

Returns the expected variable type for this PolygonHistogramExplorer.

[JScrollPane](#) getParameterTable()

Returns the parameter table.

Returns the parameter table for this EllipseHistogramExplorer.

Return

[JScrollPane](#)

Returns the parameter table for this EllipseHistogramExplorer.

2.290. PolygonHistogramPanel

Full Name:	herschel.ia.toolbox.image.gui.PolygonHistogramPanel
Type:	Java Class - 
Import:	from herschel.ia.toolbox.image.gui import PolygonHistogramPanel

Description

A panel for the PolygonHistogramTask.

A panel to serve as GUI for the PolygonHistogramTask.

API Summary

Constructor
PolygonHistogramPanel() The construction of a new PolygonHistogramPanel.
Methods
drawFigure() Drawing the rectangle on the associated image.
updateFigure() Updates the associated polygon.
trigger() Trigger the execution of the associated task.
updateHistogram() Updates the associated histogram.

API details

Constructor

<code>PolygonHistogramPanel()</code>
The construction of a new PolygonHistogramPanel.
The construction of a new PolygonHistogramPanel.


Methods

<code>drawFigure()</code>
Drawing the rectangle on the associated image.
Draws the rectangle on the image associated with this RectangleHistogramPanel.
<code>updateFigure()</code>
Updates the associated polygon.
Updates the polygon associated with this PolygonHistogramPanel.

trigger()
Trigger the execution of the associated task.
Triggers the execution of the task associated with this PolygonHistogramPanel.

updateHistogram()
Updates the associated histogram.
Updates the histogram associated with this PolygonHistogramPanel.

2.291. PolygonHistogramProduct

Full Name:	herschel.ia.dataset.image.PolygonHistogramProduct
Type:	Java Class - 
Import:	from herschel.ia.dataset.image import PolygonHistogramProduct

Description

A class to deal with the results of a polygon histogram.

API Summary

Constructor
PolygonHistogramProduct() The constructor of a new PolygonHistogramProduct.
Methods
setEdges(Double1d edgesPixels) Sets the edges for this PolygonHistogramProduct to the given
setEdges(Double1d edgesPixels, String1d edgesSky) Sets the edges for this PolygonHistogramProduct to the given list
CompositeDataset getEdges() Returns the edges for this PolygonHistogramProduct.
int getNbOfEdges() Returns the number of edges for this
TableDataset getEdgesPixelCoordinates() Returns the edges for this PolygonHistogramProduct in
Double2d getEdgesPixelCoordinatesDouble2d() Returns the pixel coordinates of the edges as Double2d.
TableDataset getEdgesSkyCoordinates() Returns the edges for this PolygonHistogramProduct in

API details

Constructor

PolygonHistogramProduct()
The constructor of a new PolygonHistogramProduct.

Methods

setEdges(Double1d edgesPixels)
Sets the edges for this PolygonHistogramProduct to the given pixel coordinates.
Argument

setEdges([Double1d](#) edgesPixels)

[Double1d](#) edgesPixels [INPUT, MANDATORY]

setEdges([Double1d](#) edgesPixels, [String1d](#) edgesSky)

Sets the edges for this PolygonHistogramProduct to the given list

of pixel and sky coordinates.

Arguments

[Double1d](#) edgesPixels [INPUT, MANDATORY]

[String1d](#) edgesSky [INPUT, MANDATORY]

CompositeDataset getEdges()

Returns the edges for this PolygonHistogramProduct.

Return

CompositeDataset

Returns the edges for this PolygonHistogramProduct.

int getNbOfEdges()

Returns the number of edges for this

PolygonHistogramProduct.

Return

int

Returns the number of edges for this PolygonHistogramTask.

TableDataset getEdgesPixelCoordinates()

Returns the edges for this PolygonHistogramProduct in

pixel coordinates.

Return

TableDataset

Returns the edges for this PolygonHistogramProduct in pixel coordinates.

[Double2d](#) getEdgesPixelCoordinatesDouble2d()

Returns the pixel coordinates of the edges as Double2d.

Return

[Double2d](#)

Returns the pixel coordinates of the edges as a Double2d.

TableDataset getEdgesSkyCoordinates()

Returns the edges for this PolygonHistogramProduct in

sky coordinates.


TableDataset `getEdgesSkyCoordinates()`

Return

TableDataset

Returns the edges for this Polygon in sky coordinates.

2.292. PolygonHistogramTask

Full Name:	herschel.ia.toolbox.image.PolygonHistogramTask
Type:	Java Task - 
Import:	from herschel.ia.toolbox.image import PolygonHistogramTask

Description

A Task to make a histogram within a polygon.

A Task to make a histogram of a region of interest, which is bounded by a polygon.

API Summary

Properties
Image image [INPUT, MANDATORY, default=No default value]
Double lowCut [INPUT, OPTIONAL, default=No default value]
Double highCut [INPUT, OPTIONAL, default=No default value]
Integer bins [INPUT, MANDATORY, default=No default values]
PolygonHistogramProduct histogram [OUTPUT, MANDATORY, default=No default value]
DoubleIcd frequencies [OUTPUT, MANDATORY, default=No default value]
DoubleIcd edgesPixel [INPUT, OPTIONAL, default=No default value]
StringIcd edgesSky [INPUT, OPTIONAL, default=No default value]

API details

Properties

Image image [INPUT, MANDATORY, default=No default value]
The image.
Double lowCut [INPUT, OPTIONAL, default=No default value]
The lower cut level.
Double highCut [INPUT, OPTIONAL, default=No default value]
The upper cut level.
Integer bins [INPUT, MANDATORY, default=No default values]
The number of bins.
PolygonHistogramProduct histogram [OUTPUT, MANDATORY, default=No default value]
The histogram.
DoubleIcd frequencies [OUTPUT, MANDATORY, default=No default value]
The frequencies.


[DoubleId](#) edgesPixel [INPUT, OPTIONAL, default=No default value]

The edges of the polygon in pixel coordinates.

[StringId](#) edgesSky [INPUT, OPTIONAL, default=No default value]

The edges of the polygon in sky coordinates.

2.293. PolygonIntersect

Full Name:	herschel.ia.toolbox.image.PolygonIntersect
Type:	Java Class - 
Import:	from herschel.ia.toolbox.image import PolygonIntersect

Description

Area of intersection of polygons.

Area of intersection of polygons.

API Summary

Method
double <code>getIntersectionArea</code>(Point2D[] polygon1Points, Point2D[] polygon2Points) Returns the area of intersection of two polygons.

Miscellaneous


Algorithm based on <http://cap-lore.com/MathPhys/IP/> Adapted 9-May-2006 by Lagado.

API details

Method

<pre>double getIntersectionArea(Point2D[] polygon1Points, Point2D[] polygon2Points)</pre>
Returns the area of intersection of two polygons.
Returns the area of intersection of two given polygons.
Arguments
<pre>Point2D[] polygon1Points [INPUT, MANDATORY]</pre>
<pre>Point2D[] polygon2Points [INPUT, MANDATORY]</pre>
Return
<pre>double</pre>
Returns the area of intersection of two given polygons.

2.294. Polynomial

Full Name:	herschel.ia.numeric.toolbox.basic.Polynomial
Type:	Java Class - 
Import:	from herschel.ia.numeric.toolbox.basic import Polynomial

Description

Calculates the y points of a polynomial for a given input array.

The input array is an integral or floating-point numeric array (e.g. Double5d) and the coefficients is a double array.

Example

Example 1: Apply Polynomial on a Int1d

```
print Polynomial(Double1d([1]))(Double1d.range(5)) # [1.0,1.0,1.0,1.0,1.0]
print Polynomial(Double1d([0,1]))(Double1d.range(5)) # [0.0,1.0,2.0,3.0,4.0]
print Polynomial(Double1d([0,0,1]))(Double1d.range(5)) # [0.0,1.0,4.0,9.0,16.0]
#or
print Double1d.range(5).apply(Polynomial(Double1d([1]))) #
[1.0,1.0,1.0,1.0,1.0]
print Double1d.range(5).apply(Polynomial(Double1d([0,1]))) #
[0.0,1.0,2.0,3.0,4.0]
print Double1d.range(5).apply(Polynomial(Double1d([0,0,1]))) #
[0.0,1.0,4.0,9.0,16.0]
```

API Summary

Jython Syntax

```
<y>=Polynomial(<coefficients>)(<x>)
```

Properties

[an integral or floating-point numeric array **x** \[INPUT, MANDATORY, default=no default value\]](#)

[a double array **coefficients** \[INPUT, MANDATORY, default=no default value\]](#)

[array **y** \[OUTPUT, MANDATORY, default=no default value\]](#)

API details

Properties

an integral or floating-point numeric array **x [INPUT, MANDATORY, default=no default value]**

The input array is an integral or floating-point numeric array.


a double array **coefficients [INPUT, MANDATORY, default=no default value]**

The number of element to shift

<code>array y [OUTPUT, MANDATORY, default=no default value]</code>
--

Returns an array.

2.295. PolynomialModel

Full Name:	herschel.ia.numeric.toolbox.fit.PolynomialModel
Type:	Java Class - 
Import:	from herschel.ia.numeric.toolbox.fit import PolynomialModel

Description

General polynomial model of arbitrary degree.

$$f(x;p) = \sum p_k * x^k$$

where the sum is over k running from 0 to degree (inclusive).

It is a linear model.


See [example](#)

Example

Example 1: PolynomialModel

```
poly = PolynomialModel( 3 -)           # 3rd degree polynomial
print poly.getNumberOfParameters()    # 4
```

2.296. PolySurfaceModel

Full Name:	herschel.ia.numeric.toolbox.fit.PolySurfaceModel
Type:	Java Class - 
Import:	from herschel.ia.numeric.toolbox.fit import PolySurfaceModel

Description

General polynomial surface model of arbitrary degree.

$$f(x,y;p) = \sum_d \sum_k p_n * x^k * y^{\text{degree} - k}$$


where the first sum is over d running from 0 to degree (inclusive) and the second sum is over k running from 0 to d (inclusive). The index n is just incrementing, making all p's different.

It is a 2-dimensional linear model.

Example

Example 1: PolySurfaceModel	
<pre>poly = PolySurfaceModel(3 -) # 3rd degree polynomial print poly.getNumberOfParameters() # 10</pre>	

2.297. poolDataReader

Full Name:	herschel.ia.toolbox.util.PoolDataReaderTask
Alias:	poolDataReader
Type:	Java Task - 
Import:	from herschel.ia.toolbox.util import PoolDataReaderTask
Category:	utility task

Description

Task for reading a product from a pool.

Data will be readed through a ProductStorage by default. If no storage is specified, a new one will be created (unless the parameter 'useStorage' is false). Only the product identifier is required (an urn or a tag).

Examples

Example 1: Read by urn

```
pool = PoolManager.getInstance().get('myPool')
store = ProductStorage(pool)
store.save(Product())
reader = PoolDataReaderTask()
product = reader(id='urn:myPool:herschel.ia.dataset.Product:0',poolid='myPool')
print product
```

Example 2: Read by tag

```
pool = PoolManager.getInstance().get('myPool')
store = ProductStorage(pool)
store.saveAs(Product(), 'myTag')
reader = PoolDataReaderTask()
product = reader(id='myTag',poolid='myPool')
print product
```

Example 3: Read two products using the same storage

```
pool = PoolManager.getInstance().get('myPool')
store = ProductStorage(pool)
store.saveAs(Product(), -'myTag1')
store.saveAs(Product(), -'myTag2')
ps = ProductStorage()
pool = PoolManager.getPool("myPool")
ps.register(pool)
reader = PoolDataReaderTask()
product1 = reader(id='myTag1',storage=ps)
print product1
product2 = reader(id='myTag2',storage=ps)
print product2
```

Example 4: Read two products using the same default storage

```
pool = PoolManager.getInstance().get('standard')
store = ProductStorage(pool)
store.saveAs(Product(), -'myTag1')
store.saveAs(Product(), -'myTag2')
pdr = PoolDataReaderTask()
product1 = pdr(id='myTag1')
```

Example 4: Read two products using the same default storage

```
print product1
ps = pdrt.storage
product2 = pdrt(id='myTag2',storage=ps)
print product2
```

API Summary

Jython Syntax

```
product=PoolDataReaderTask()(urn='urn:poolid')
see examples
```

Properties

[String id](#) [INPUT, MANDATORY, default=null]

[ProductStorage storage](#) [INPUT, OPTIONAL, default=new ProductStorage]

[ProductPool pool](#) [INPUT, OPTIONAL, default=null (deprecated) PAL default pool]

[String poolid](#) [INPUT, OPTIONAL, default=null]

[Boolean useStorage](#) [INPUT, OPTIONAL, default=true]

[Product product](#) [OUTPUT, MANDATORY, default=null]

API details

Properties

[String id](#) [INPUT, MANDATORY, default=null]

Product identifier, either an urn or a tag. A Tag can be used when reading through storage only.

[ProductStorage storage](#) [INPUT, OPTIONAL, default=new ProductStorage]

If storage is specified, pool and poolid parameters are not allowed. If storage is specified and useStorage parameter is false, an exception will be raised. If no storage is specified, pool or poolid parameters will be used and a new storage will be created (unless useStorage is false) and will be available as an output parameter.

[ProductPool pool](#) [INPUT, OPTIONAL, default=null (deprecated) PAL default pool]

If storage is specified, the pool parameter is not allowed. If a pool is specified, the poolid parameter is not allowed. DEPRECATED: If no pool is specified, the default pool will be used. (see {@link herschel.ia.pal}) A product storage will be created (unless useStorage is false) and that storage will be available as an output parameter

[String poolid](#) [INPUT, OPTIONAL, default=null]

Pool identifier. {@link herschel.ia.pal.PoolManager} will be used to find the pool. If storage is specified, the poolid parameter is not allowed. If a poolid is specified, the pool parameter is not allowed. DEPRECATED: If no poolid is specified, the default pool will be used. (see {@link herschel.ia.pal}) A product storage will be created (unless useStorage is false) and that storage will be available as an output parameter.

Boolean useStorage [INPUT, OPTIONAL, default=true]

Enable/Disable storage usage. If this parameter is false and storage is specified, an exception will be raised. If this parameter is true and no storage is specified, a new { @link herschel.ia.pal.ProductStorage } will be created and will be available as an output parameter.


Product product [OUTPUT, MANDATORY, default=null]
--

The loaded product.

History

- 22-11-2007 JCS
- 2009-02-17 - JDS: (default pool, SPR-6006)
- 2009-07-07 - JDS: Using PoolManager.getPool("standard")

2.298. poolDataWriter

Full Name:	herschel.ia.toolbox.util.PoolDataWriterTask
Alias:	poolDataWriter
Type:	Java Task - 
Import:	from herschel.ia.toolbox.util import PoolDataWriterTask
Category:	utility task

Description

Task for writing a product into a pool.

Data will be saved through a ProductStorage by default. If no storage is specified, a new one will be created and available (unless the parameter 'useStorage' is false).

Examples

Example 1: write product

```
p = Product()
pdwt = PoolDataWriterTask()
urn = pdwt(product=p)
print urn
```

Example 2: write product with a tag and read it later

```
p = Product()
pdwt = PoolDataWriterTask()
urn = pdwt(product=p, tag='myTag', poolid='myPool')
print urn
# ...do some work...
pdrt = PoolDataReaderTask()
product = pdrt(id='myTag', poolid='myPool')
print product
```

Example 3: write two products using the same storage

```
ps = ProductStorage()
pool = PoolManager.getPool("standard")
ps.register(pool)
pdwt = PoolDataWriterTask()
urn1 = pdwt(product=Product(), tag='myTag1', storage=ps)
print urn1
urn2 = pdwt(product=Product(), tag='myTag2', storage=ps)
print urn2
```

Example 4: write two products using the same default storage

```
pdwt = PoolDataWriterTask()
urn1 = pdwt(product=Product(), tag='myTag1')
print urn1
ps = pdwt.storage
urn2 = pdwt(product=Product(), tag='myTag2', storage=ps)
print urn2
```

API Summary

Jython Syntax

```
urn=PoolDataWriterTask()(product=Product())
```

Jython Syntax
See examples.
Properties
Product product [INPUT, MANDATORY, default=null]
String tag [INPUT, OPTIONAL, default=null]
ProductStorage storage [INPUT, OPTIONAL, default=new ProductStorage]
ProductPool pool [INPUT, OPTIONAL, default=PAL default pool]
String poolid [INPUT, OPTIONAL, default=null]
String urn [OUTPUT, MANDATORY, default=null]
Boolean useStorage [INPUT, OPTIONAL, default=true]

API details

Properties

Product product [INPUT, MANDATORY, default=null]
The product to be saved.
String tag [INPUT, OPTIONAL, default=null]
Product tag. A Tag can be used when saving through storage only.
ProductStorage storage [INPUT, OPTIONAL, default=new ProductStorage]
If storage is specified, pool and poolid parameters are not allowed. If storage is specified and useStorage parameter is false, an exception will be raised. If no storage is specified, pool or poolid parameters will be used and a new storage will be created (unless useStorage is false) and will be available as an output parameter.
ProductPool pool [INPUT, OPTIONAL, default=PAL default pool]
If storage is specified, the pool parameter is not allowed. If a pool is specified, the poolid parameter is not allowed. DEPRECATED: If no pool is specified, the default pool will be used. (see {@link herschel.ia.pal}) A product storage will be created (unless useStorage is false) and that storage will be available as an output parameter
String poolid [INPUT, OPTIONAL, default=null]
Pool identifier. {@link herschel.ia.pal.PoolManager} will be used to find the pool. If storage is specified, the poolid parameter is not allowed. If a poolid is specified, the pool parameter is not allowed. DEPRECATED: If no poolid is specified, the default pool will be used. (see {@link herschel.ia.pal}) A product storage will be created (unless useStorage is false) and that storage will be available as an output parameter.
String urn [OUTPUT, MANDATORY, default=null]
The urn of the product just written.
Boolean useStorage [INPUT, OPTIONAL, default=true]
Enable/Disable storage usage. If this parameter is false and storage is specified, an exception will be raised. If this parameter is true and no storage is specified, a new {@link herschel.ia.pal.ProductStorage} will be created and will be available as an output parameter.

History

- 22-11-2007 JCS
- 2008-12-15 - JDS: (added deprecation marks to jtags)
- 2009-02-17 - JDS: (default pool, SPR-6006)
- 2009-07-07 - JDS: Using PoolManager.getPool("standard")

2.299. PositionList

Full Name:	herschel.ia.dataset.image.PositionList
Type:	Java Class - 
Import:	from herschel.ia.dataset.image import PositionList

Description

Position list.

A list of positions.

API Summary

Methods
double getRa(int row) Returns the RA at a given row.
double getDec(int row) Returns the Dec at a given row.
int getSize() Returns the number of positions.

API details

Methods

double getRa(int row) Returns the RA at a given row. Returns the RA for the position for this PositionList at the given row in the list.
Argument int row [INPUT, MANDATORY]
Return double Returns the RA for the position for this PositionList at the given row in the list.
double getDec(int row) Returns the Dec at a given row. Returns the Dec for the position for this PositionList at the given row in the list.
Argument int row [INPUT, MANDATORY]
Return double Returns the Dec for the position for this PositionList at the given row in the list.

int `getSize()`

Returns the number of positions.


Returns the number of positions for this PositionList.

Return

int

Returns the number of positions for this PositionList.

2.300. PowerLawModel

Full Name:	herschel.ia.numeric.toolbox.fit.PowerLawModel
Type:	Java Class - 
Import:	from herschel.ia.numeric.toolbox.fit import PowerLawModel

Description

General powerlaw model of arbitrary degree.

$$f(x;p) = p_0 * (x - p_1)^{p_2}$$


p_0 = amplitude p_1 = x-shift p_2 = power

The parameters are initialized at {1.0, 0.0, 1.0}.

Example

Example 1: PowerLawModel
<pre>pl = PowerLawModel() print pl.getNumberOfParameters() # 3</pre>

2.301. PowerModel

Full Name:	herschel.ia.numeric.toolbox.fit.PowerModel
Type:	Java Class - 
Import:	from herschel.ia.numeric.toolbox.fit import PowerModel

Description

General power model of arbitrary degree.


$$f(x;p) = p * x^k$$

k is an integer (positive or negative).

Example

Example 1: PowerModel
<pre>pwr = PowerModel(--1 -) print pwr.getNumberOfParameters() # 1</pre>

2.302. PowerSpectrum

Full Name:	herschel.ia.gui.explorer.table.PowerSpectrum
Type:	Java Class - 
Import:	from herschel.ia.gui.explorer.table import PowerSpectrum

Description

PowerSpectrum Generator OverView

The PowerSpectrum generator is a GUI tool which allows users to generate PowerSpectrum inactively. The tool allows users to specify the time data and unit. By default, the tool will look for the time data in the table. If there are several time column data, the lowest column index will be used as a time data to calculate frequency.

API Summary

Constructor
PowerSpectrum(TableDataset table)

Methods
setObject(TableDataset data)
JComponent getComponent()

API details

Constructor

PowerSpectrum(TableDataset table)
Argument
TableDataset table [INPUT, MANDATORY, default=no default value] The input table must contain one or more time column data.


Methods

setObject(TableDataset data)
Argument
TableDataset data [INPUT, MANDATORY, default=no default value] The input table must contain at least one time column data.
JComponent getComponent()
Return
JComponent - this GUI component

History

- 2007-12-02 - first: version
- 2008-04-03 - Move: the code to calculate PowerSpectrum to ia_numeric_toolbox_xform
- 2008-11-21 - Major: GUI change to allow users to choose time data and its unit

2.303. PowerSpectrum

Full Name:	herschel.ia.toolbox.astro.PowerSpectrum
Type:	Java Class - 
Import:	from herschel.ia.toolbox.astro import PowerSpectrum

Examples

Example 1: java example import herschel.ia.numeric.toolbox.xform. *;

```
TableDataset table; //defined somewhere double flimit = 0.1; double
sigma = 0.4; boolean deglitch = true;
TableDataset pw_table = PowerSpectrum.getPowerSpectrum(flmit,
sigma, deglitch, table);
```

Example 2: jython example from herschel.ia.numeric.toolbox.xform import *

```
flimt = 0.1 sima = 0.4 deglitch= 1 pw_table =
PowerSpectrum.powerSpectrum(flmit, sigma, deglitch, table)
```


See also

- [???](#)

History

- Feb 22, 2008 - first version

2.304. Pow

Full Name:	herschel.ia.numeric.toolbox.basic.Pow
Type:	Java Class - 
Import:	from herschel.ia.numeric.toolbox.basic import Pow

Description

Gives the array raised to the specified power: $y=(x)^{\text{power}}$

Example

Example 1: Apply Pow on a Int1d
<pre>x=Double1d([1,2,3,4]) print Pow(2)(x) # [1.0,4.0,9.0,16.0] print x.apply(Pow(2)) # [1.0,4.0,9.0,16.0]</pre>

API Summary

Jython Syntax
<code><y>=Pow(<power>)(<x>)</code>
Properties
any array or number x [INPUT, MANDATORY, default=no default value]
a number power [INPUT, MANDATORY, default=no default value]
a number or an array y [OUTPUT, MANDATORY, default=no default value]

API details


Properties

any array or number x [INPUT, MANDATORY, default=no default value]
An array or a number
a number power [INPUT, MANDATORY, default=no default value]
The power
a number or an array y [OUTPUT, MANDATORY, default=no default value]
The result is a number or an array depending on the input

See also

- [Pow](#)

2.305. PreviousInterpolator

Full Name:	herschel.ia.numeric.toolbox.interp.PreviousInterpolator
Alias:	PreviousInterpolator
Type:	Java Class - 
Import:	from herschel.ia.numeric.toolbox.interp import PreviousInterpolator

Description

Creates an linear interpolation function from a set of knots (x,y),
that can be applied to numeric arrays of rank 1.

Example

Example 1: Create and apply a PreviousInterpolator on a Double1d
<pre># create some knots x=Double1d.range(10) # [0.0,1.0,2.0,3.0,4.0,5.0,6.0,7.0,8.0,9.0] f=PreviousInterpolator(x,SQUARE(x)) u=Double1d([1,1.1,2,2.6,3,3.9]) print f(u) # [1.0,1.0,4.0,4.0,9.0,9.0]</pre>

API Summary

Jython Syntax
<f>=PreviousInterpolator(<x>,<y>[,allowExtrapolation])
Properties
Double1d x [INPUT, MANDATORY, default=no default value]
Double1d y [INPUT, NOT_MANDATORY, default=false]
boolean allowExtrapolation [INPUT, NOT_MANDATORY, default=false]

API details

Properties


Double1d x [INPUT, MANDATORY, default=no default value]
The knots are Double1d
Double1d y [INPUT, NOT_MANDATORY, default=false]
The knots are Double1d
boolean allowExtrapolation [INPUT, NOT_MANDATORY, default=false]
Extrapolation is not allowed by default. Boolean.TRUE allows extrapolation.

See also

- [CubicSplineInterpolator](#)

- [LinearInterpolator](#)

2.306. PrfGaussian

Full Name:	herschel.ia.toolbox.srcext.PrfGaussian
Type:	Java Class - 
Import:	from herschel.ia.toolbox.srcext import PrfGaussian

Description

A representation of a Gaussian point response function as an Image.

The value in each pixel is the average of a two-dimensional Gaussian over that region.

More detail in the Developer's Reference Manual.

Examples


Example 1: Create image with point sources at pixel positions

```
# FWHM = 1.0 pixels
prf = PrfGaussian([100, 100], 1.0)
#
# Source at (10, 20) with peak of 2.0
sourceImage = prf.of(10, 20, 2.0)
```

Example 2: Create image with point sources at world coordinates

```
# image = SimpleImage with WCS defined
# FWHM = 1.0 arcsec
prf = PrfGaussian(image, 1.0)
#
# Source at (ra, dec) = (180, 30) degrees with peak flux 0.1
sourceImage = prf.of(180, 30, 0.1)
```

2.307. ProcessDistributor

Full Name:	herschel.ia.dataflow.ProcessDistributor
Type:	Java Class - 
Import:	from herschel.ia.dataflow import ProcessDistributor

Description

Decouples the chain of execution of processes, i.e, makes a event-based process to behave as thread-based one.

Helper class that allows one or more Event-based processes (or connectables in general) to be run in a different thread.

This is intended to be used as is.

As typical example, a dataflow containing three Event-based processes A, B, C, whose flow is:

```
A----B
|
---C
```

The user (or developer) sees that he needs better performance running B process in another thread, so a PD (product-distributor) process is added to the dataflow this way:

```
A----PD----B
|
---C
```

In case B and C must run in the same thread, the dataflow can be modified like this:

```
A----PD----B
|
---C
```

The ProcessProductDistributor has an input called "input" and a output called "output".

Example

Example 1: how to use a process distributor.

```
<pre>
from herschel.ia.dataflow import *
from myProcesses import *
df = DataFlow("mydf")
df.createProcess("avg", AverageProcess)
df.createProcess("viewer", ViewerProcess)
p = ProcessDistributor("distrib", DoubleId)
df.addProcess(p)
df.connect("avg.output", -"distrib.input")
df.connect("distrib.output", -"viewer.input")
df.start()
</pre>
```

Limitations

no limitation


See also

- [reference](#)

History

- 26-5-2005 jcg: change javadoc format for help support.

2.308. PRODUCT

Full Name:	herschel.ia.numeric.toolbox.basic.Product
Alias:	PRODUCT
Type:	Java Class - 
Import:	from herschel.ia.numeric.toolbox.basic import Product

Description

Yields the product of all elements in the input array.

Returns a scalar of the same type as the type of the input array.

Example

Example 1: Apply PRODUCT on a Int2d
<pre>x=Int2d([[1,2], [-1,3] -]) print PRODUCT(x) # --6 print PRODUCT(x, 0) # [-1, 6] print PRODUCT(x, 1) # [2,-3]</pre>

API Summary

Jython Syntax
<code><y>=PRODUCT(<x>, [, <dim>])</code>
Properties
any scalar array x [INPUT, MANDATORY, default=no default value]
integer dim [INPUT, MANDATORY, default=no default value]
float or double array y [OUTPUT, MANDATORY, default=no default value]

API details


Properties

any scalar array x [INPUT, MANDATORY, default=no default value]
The input array can be any scalar array.
integer dim [INPUT, MANDATORY, default=no default value]
The dimension to compute the calculation along
float or double array y [OUTPUT, MANDATORY, default=no default value]
Returns a scalar of the same type as the type of the input array.

See also

- [SUM](#)

2.309. ProductRef

Full Name:	herschel.ia.pal.ProductRef
Type:	Java Class - 
Import:	from herschel.ia.pal import ProductRef

Description

A ProductRef provides a reference to a product that is held in a

product storage or to a product in memory. Typically a ProductRef is returned by the load, save and select methods of a ProductStorage. A Product reference is providing a mechanism to inspect the meta data of a stored product without loading the complete product into memory.

Example

Example 1: Usage of a product reference p=Product(creator="me")

```
ref=storage.save(p)
print ref.type # herschel.ia.dataset.Product print ref.urn #
urn:simple.default:herschel.ia.dataset.Product:23674 print
ref.meta['creator'] # me print ref.product.creator # me (product
loaded into memory!)
```

API Summary

Methods
getProduct() Returns the Product class to which this Product reference is
getType() Returns the Product class to which this Product reference is

API details


Methods

getProduct() Returns the Product class to which this Product reference is pointing to.
getType() Returns the Product class to which this Product reference is pointing to.

See also

- [???](#)

2.310. ProductStorage

Full Name:	herschel.ia.pal.ProductStorage
Type:	Java Class - 
Import:	from herschel.ia.pal import ProductStorage

Description

The ProductStorage is a storage mechanism to provide read, write and query on Products stored in registered Product Pools.

Examples

Example 1: creation and registering myStore=ProductStorage()

```
myStore.register(LocalStoreFactory.getStore("foo")) # the foo
lstore pool
```

Example 2: loading a Product from this store

```
ref=myStore.load("urn:foo:herschel.ia.dataset.Product:123" -)
```

Example 3: Untitled

```
saving a Product product=Product(...) ref=myStorage.save(product)
```

Example 4: select Products based on a query query=AttribQuery(...)

```
results=myStorage.select(query)
```

API Summary

Constructors
ProductStorage() Creates an empty ProductStorage with no pools registered.
ProductStorage(ProductPool pool) Creates a ProductStorage with the given pool registered, which
ProductStorage(ProductPool[] pools) Creates a ProductStorage with the given pools registered.
ProductStorage(String poolName) Creates a ProductStorage with the the pools corresponding to the
ProductStorage(String[] poolNames) Creates a ProductStorage with the the pools corresponding to the
Methods
setSharedMode(boolean sharedMode)

Methods	
	Sets whether the last version info is to be updated before each
register(ProductPool pool)	Registers a ProductPool to the ProductStorage. The first
register(ProductStorage[] stores)	Chain storage registrations, register storages in storages. The
register(ProductStorage. The product pools that store)	Chain storage registrations, register pools in another storage.
register(ProductPool array pools)	Registers ProductPools to the ProductStorage. The first
getPools()	Provides the set of ProductPools registered, in order in which
getWritablePool()	Returns the writable pool, which is the first registered pool.
Set getProductClasses()	Provides all Product class definitions found in this pool.
remove(String urn)	Removes a product of given URN from the storage.
ProductRef load(String id)	Loads a Product from this store.
ProductRef save(Product product)	Saves a Product to this store. If the product is a context that
ProductRef saveHeadOnly(Product product)	Saves a Product to this store. If the product is a context that
Set select(StorageQuery query)	Returns a set of references to products that match the specified
Set select(StorageQuery query, Set previous)	Returns a set of references to products that match the specified

API details

Constructors

ProductStorage()
Creates an empty ProductStorage with no pools registered.
ProductStorage(ProductPool pool)
Creates a ProductStorage with the given pool registered, which happens to be the writable one.
Argument
ProductPool pool [INPUT, MANDATORY]
Example
Creating a storage initialized with a pool.

ProductStorage(ProductPool pool)

```
storage = ProductStorage(pool) # pool is the writable one
```

ProductStorage(ProductPool[] pools)

Creates a ProductStorage with the given pools registered.

The first provided pool is the writable one.

Argument

ProductPool[] **pools** [INPUT, MANDATORY]

Example

Creating a storage initialized with pools.

```
pool1 = PoolManager.getPool("pool1")
pool2 = PoolManager.getPool("pool2")
pool3 = PoolManager.getPool("pool3")
storage = ProductStorage([pool1, pool2, pool3]) # pool1 is the
writable one
```

ProductStorage(String poolName)

Creates a ProductStorage with the the pools corresponding to the given name. This pool is the writable one.

Argument

String **poolName** [INPUT, MANDATORY]

Example

Creating a storage initialized with pools. # the following line

```
storage = ProductStorage("pool") # is equivalent to
pool = PoolManager.getPool("pool")
storage = ProductStorage(pool)
```

ProductStorage(String[] poolNames)

Creates a ProductStorage with the the pools corresponding to the given names registered. The first provided pool is the writable one.

Argument

String[] **poolNames** [INPUT, MANDATORY]

Example

Creating a storage initialized with pools. # the following line

```
storage = ProductStorage(["pool1", "-pool2", "-pool3"]) # is
equivalent to
pool1 = PoolManager.getPool("pool1")
pool2 = PoolManager.getPool("pool2")
pool3 = PoolManager.getPool("pool3")
storage = ProductStorage([pool1, pool2, pool3])
```

Methods

setSharedMode(boolean sharedMode)

Sets whether the last version info is to be updated before each

setSharedMode(boolean sharedMode)

operation on the storage. Set it to `true` if you expect any other `ProductStorage` may update the products from the outside.

This can be set also in the initialization of the application, by the property `hcss.ia.pal.sharedMode`.

Argument

boolean **sharedMode** [INPUT, MANDATORY, default=Whether the storage is]

accessed for writing by several processes at the same time.

register(ProductPool pool)

Registers a `ProductPool` to the `ProductStorage`. The first

`ProductPool` will be the one for which save operations would write products to. The remaining pools registered will be read-only.

Argument

`ProductPool` **pool** [INPUT, MANDATORY, default=The product pool that]

will be added to this product storage.

Example

How to register a product pool

```
storage.register(pool)
```

register([ProductStorage\[\]](#) stores)

Chain storage registrations, register storages in storages. The

first `ProductPool` that registered will be the one for which save operations would write products to. The remaining pools registered will be read-only.

Argument

[ProductStorage\[\]](#) **stores** [INPUT, MANDATORY]

Example

How to register a product storage

```
storage.register(store)
```

register(ProductStorage. The product pools that store)

Chain storage registrations, register pools in another storage.

The first `ProductPool` that registered will be the one for which save operations would write products to. The remaining pools registered will be read-only.

Argument

`ProductStorage. The product pools that` **store** [INPUT, MANDATORY, default=no default value]

registered in that storage will be added to this product storage.

Example

How to register a product storage

register(ProductStorage. The product pools that store)

```
storage.register(store)
```

register(ProductPool array pools)

Registers ProductPools to the ProductStorage. The first

ProductPool will be the one for which save operations would write products to. The remaining pools registered will be read-only.

Argument

ProductPool array **pools** [INPUT, MANDATORY, default=The product pools]

that will be added to this product storage.

Example

How to register many pools at the same time

```
storage.register([pool1, pool2, pool3])
```

getPools()

Provides the set of ProductPools registered, in order in which they were initially registered.

getWritablePool()

Returns the writable pool, which is the first registered pool.

[Set](#) getProductClasses()

Provides all Product class definitions found in this pool.

Return

[Set](#)

a collection of Product classes

remove([String](#) urn)

Removes a product of given URN from the storage.

Argument

[String](#) **urn** [INPUT, MANDATORY, default=The URN to the Product that]

should be removed.

[ProductRef](#) load([String](#) id)

Loads a Product from this store.

Argument

[String](#) **id** [INPUT, MANDATORY, default=The URN or tag to the Product]

Return

[ProductRef](#)

A reference to the Product corresponding to the input URN or tag.

[ProductRef](#) save(Product product)

Saves a Product to this store. If the product is a context that

has descendents, all those descendents will be copied if they do not already exist in the destination storage (this is a deep-copy or cloning operation).

Argument

Product **product** [INPUT, MANDATORY, default=The product that should be]
saved.

Return

[ProductRef](#)

A reference to the Product saved into the store.

[ProductRef](#) saveHeadOnly(Product product)

Saves a Product to this store. If the product is a context that

has descendents, all those descendents will be copied if they do not already exist in the destination storage (this is a deep-copy or cloning operation).

Argument

Product **product** [INPUT, MANDATORY, default=The product that should be]
saved.

Return

[ProductRef](#)

A reference to the Product saved into the store.

[Set](#) select(StorageQuery query)

Returns a set of references to products that match the specified

query.

Argument

StorageQuery **query** [INPUT, MANDATORY, default=The query applied to]
this store.

Return

[Set](#)

A set of references to Products selected by the query

[Set](#) select(StorageQuery query, [Set](#) previous)

Returns a set of references to products that match the specified

query.

Arguments

StorageQuery **query** [INPUT, MANDATORY, default=The query applied to]

[Set](#) `select(StorageQuery query, Set previous)`

this store.

[Set](#) `previous` [INPUT, MANDATORY]

Return


[Set](#)

A set of references to Products selected by the query

See also

- [???](#)

2.311. ProfileExplorer

Full Name:	herschel.ia.gui.image.ProfileExplorer
Type:	Java Class - 
Import:	from herschel.ia.gui.image import ProfileExplorer

Description

An explorer for Profiles.

An explorer for Profiles.

API Summary

Constructors
ProfileExplorer() The construction of a new ProfileExplorer.
ProfileExplorer(Object object) The construction of a new ProfileExplorer associated with the given object
Methods
String getName() Returns the name.
String getDescription() Returns the description.
boolean canHandle(Class className) Checks whether this ProfileExplorer can handle objects of the given class.
setObject(Object object) Setting the object.
Profile getObject() Returns the object.
Class getVariableType() Returns the expected variable type.
JComponent getComponent() Returns the component that is responsible for displaying the data object.
JPanel getCoordinatePanel() Returns the coordinate table.
JPanel getProfilePlotPanel() Returns the profile plot component.
addDataObjectListener(DataObjectListener listener) Adds the given listener.
removeDataObjectListener(DataObjectListener listener) Removes the given listener.

API details

Constructors

ProfileExplorer()
The construction of a new ProfileExplorer.
The construction of a new ProfileExplorer.

ProfileExplorer(Object object)
The construction of a new ProfileExplorer associated with the given object
The construction of a new ProfileExplorer associated with the given object.
Argument
<code>Object object</code> [INPUT, MANDATORY]

Methods

String getName()
Returns the name.
Returns the name for this ProfileExplorer.
Return
<code>String</code>
Returns the name for this ProfileExplorer.

String getDescription()
Returns the description.
Returns the description for this ProfileExplorer.
Return
<code>String</code>
Returns the description for this ProfileExplorer.

boolean canHandle(Class className)
Checks whether this ProfileExplorer can handle objects of the given class.
Returns true if this ProfileExplorer can handle object of the given class; false otherwise.
Argument
<code>Class className</code> [INPUT, MANDATORY]
Return
<code>boolean</code>
Returns true of this ProfileExplorer can handle objects of the given class; false otherwise.

setObject([Object](#) object)

Setting the object.

Sets the object for this ProfileExplorer to the given object.

Argument

[Object](#) **object** [INPUT, MANDATORY]

[Profile](#) getObject()

Returns the object.

Returns the object for this ProfileExplorer.

Return

[Profile](#)

Returns the object for this ProfileExplorer.

[Class](#) getVariableType()

Returns the expected variable type.

Returns the expected variable type for this ProfileExplorer.

Return

[Class](#)

Returns the expected variable type for this ProfileExplorer.

[JComponent](#) getComponent()

Returns the component that is responsible for displaying the data object.

Returns the component that is responsible for displaying the data object for this ProfileExplorer.

Return

[JComponent](#)

Returns the component that is responsible for displaying the data object for this ProfileExplorer.

[JPanel](#) getCoordinatePanel()

Returns the coordinate table.

Constructs and returns the coordinate table for this ProfileExplorer.

Return

[JPanel](#)

Returns the coordinate table for this ProfileExplorer.

[JPanel](#) getProfilePlotPanel()

Returns the profile plot component.

Returns the profile plot component for this ProfileExplorer.

[JPanel](#) `getProfilePlotPanel()`

Return

[JPanel](#)

Returns the profile plot component for this ProfileExplorer.

`addDataObjectListener(DataObjectListener listener)`

Adds the given listener.

Add the given listener to this ProfileExplorer to receive data object events from it.

Argument

DataObjectListener **listener** [INPUT, MANDATORY]

`removeDataObjectListener(DataObjectListener listener)`


Removes the given listener.

Removes the given listener for this ProfileExplorer, so that it no longer receives data object events by this explorer.

Argument

DataObjectListener **listener** [INPUT, MANDATORY]

2.312. Profile

Full Name:	herschel.ia.dataset.image.Profile
Type:	Java Class - 
Import:	from herschel.ia.dataset.image import Profile

Description

A class to deal with the results of profile plotting.

API Summary

Constructor
<p>Profile()</p> <p>The constructor of a new Profile.</p>
Methods
<p>setBegin(double beginX, double beginY)</p> <p>Sets the begin to the given pixel coordinates.</p>
<p>setBegin(double beginX, double beginY, String beginRA, String beginDec)</p> <p>Sets the begin to the given pixel and sky</p>
<p>setEnd(double endX, double endY)</p> <p>Sets the end to the given pixel coordinates.</p>
<p>setEnd(double endX, double endY, String endRA, String endDec)</p> <p>Sets the end to the given pixel and sky coordinates.</p>
<p>setProfile(DoubleId profile)</p> <p>Sets the profile for this Profile to the given</p>
<p>setUnit(Unit unit)</p> <p>Sets the unit for this Profile.</p>
<p>DoubleId getBeginPixelCoordinates()</p> <p>Returns the begin for this Profile in pixel coordinates.</p>
<p>StringId getBeginSkyCoordinates()</p> <p>Returns the begin for this Profile in sky coordinates.</p>
<p>DoubleId getEndPixelCoordinates()</p> <p>Returns the end for this Profile in pixel coordinates.</p>
<p>StringId getEndSkyCoordinates()</p> <p>Returns the end for this Profile in sky coordinates.</p>
<p>DoubleId getProfile()</p> <p>Returns the profile for this Profile.</p>
<p>String getUnit()</p> <p>Returns the unit for this Profile.</p>

API details

Constructor

Profile()
The constructor of a new Profile.

Methods

setBegin(double beginX, double beginY)
Sets the begin to the given pixel coordinates.
Arguments
double beginX [INPUT, MANDATORY]
double beginY [INPUT, MANDATORY]

setBegin(double beginX, double beginY, String beginRA, String beginDec)
Sets the begin to the given pixel and sky coordinates.
Arguments
double beginX [INPUT, MANDATORY]
double beginY [INPUT, MANDATORY]
String beginRA [INPUT, MANDATORY]
String beginDec [INPUT, MANDATORY]

setEnd(double endX, double endY)
Sets the end to the given pixel coordinates.
Arguments
double endX [INPUT, MANDATORY]
double endY [INPUT, MANDATORY]

setEnd(double endX, double endY, String endRA, String endDec)
Sets the end to the given pixel and sky coordinates.
Arguments
double endX [INPUT, MANDATORY]
double endY [INPUT, MANDATORY]
String endRA [INPUT, MANDATORY]
String endDec [INPUT, MANDATORY]

setProfile(DoubleId profile)
Sets the profile for this Profile to the given profile.
Argument
DoubleId profile [INPUT, MANDATORY]

setUnit(Unit unit)

Sets the unit for this Profile.

ArgumentUnit **unit** [INPUT, MANDATORY]**DoubleId** **getBeginPixelCoordinates()**

Returns the begin for this Profile in pixel coordinates.

Return[DoubleId](#)

Returns the begin for this Profile in pixel coordinates.

StringId **getBeginSkyCoordinates()**

Returns the begin for this Profile in sky coordinates.

Return[StringId](#)

Returns the begin for this Profile in sky coordinates.

DoubleId **getEndPixelCoordinates()**

Returns the end for this Profile in pixel coordinates.

Return[DoubleId](#)

Returns the end for this Profile in pixel coordinates.

StringId **getEndSkyCoordinates()**

Returns the end for this Profile in sky coordinates.

Return[StringId](#)

Returns the end for this Profile in sky coordinates.

DoubleId **getProfile()**

Returns the profile for this Profile.

Return[DoubleId](#)

Returns the profile for this Profile.

String **getUnit()**

Returns the unit for this Profile.

Return[String](#)

Returns the unit for this Profile.

2.313. ProfilePanel

Full Name:	herschel.ia.toolbox.image.gui.ProfilePanel
Type:	Java Class - 
Import:	from herschel.ia.toolbox.image.gui import ProfilePanel

Description

A panel for the ProfileTask.

A panel to serve as GUI for the ProfileTask.

API Summary

Constructor	
ProfilePanel()	The construction of a new ProfilePanel.
Methods	
JPanel getImagePanel()	Returns a panel that displays the image.
JPanel getButtonPanel()	Returns the button panel.
Action getRunAction()	Returns the run action for this ProfilePanel.
Action getResetAction()	Returns the reset action for this ProfilePanel.
setSiteEventHandler(SiteEventHandler handler)	Setting the site event handler
setTask(TaskApi task)	Associating the task.
setVariableSelection(VariableSelection selection)	Setting the variable selection.
Display getDisplay()	Returns the display.
Image getImage()	Returns the associated image.
ImageFigure getLine()	Returns the straight line.
Double getBegin()	Returns the begin of the straight line in pixel coordinates.
Double getEnd()	Returns the end of the straight line in pixel coordinates.
TaskApi getTask()	Returns the associated task.

Methods
Map <code>getMap()</code> Returns the associated map.
SiteEventHandler <code>getHandler()</code> Returns the associated site event handler.
PlotXY <code>getPlot()</code> Returns the associated plot.
int <code>getNbOfClicks()</code> Returns the number of clicks made on the image.
boolean <code>isInImage(MouseEvent me)</code> Checks whether the given mouse event occurred inside the image.
boolean <code>isInImage(double pixX, double pixY)</code> Checks whether the given pixel coordinates lie inside the image.

API details

Constructor

<code>ProfilePanel()</code>
The construction of a new ProfilePanel.
The construction of a new ProfilePanel.

Methods

JPanel <code>getImagePanel()</code>
Returns a panel that displays the image.
Returns a panel that displays the image for this ProfilePanel.
Return
JPanel
Returns a panel that displays the image for this ProfilePanel.

JPanel <code>getButtonPanel()</code>
Returns the button panel.
Returns the button panel for this ProfilePanel.
Return
JPanel
Returns the button panel for this ProfilePanel.

Action <code>getRunAction()</code>
Returns the run action for this ProfilePanel.
Returns the run action for this ProfilePanel.

Action <code>getRunAction()</code>
Return Action Returns the run action for this ProfilePanel.
Action <code>getResetAction()</code>
Returns the reset action for this ProfilePanel. Returns the reset action for this ProfilePanel. Return Action Returns the reset action for this ProfilePanel.
setSiteEventHandler(SiteEventHandler handler)
Setting the site event handler Sets the site event handler for this ProfilePanel to the given site event handler. Argument SiteEventHandler handler [INPUT, MANDATORY]
setTask(TaskApi task)
Associating the task. Associates the given task with this ProfilePanel. Argument TaskApi task [INPUT, MANDATORY]
setVariableSelection(VariableSelection selection)
Setting the variable selection. Sets the variable selection for this ProfilePanel to the given variable selection. Argument VariableSelection selection [INPUT, MANDATORY]
Display <code>getDisplay()</code>
Returns the display. Returns the display for this ProfilePanel. Return Display Returns the display for this ProfilePanel.
Image <code>getImage()</code>
Returns the associated image.

Image `getImage()`

Returns the image associated with this ProfilePanel.

Return

Image

Returns the image associated with this ProfilePanel.

ImageFigure `getLine()`

Returns the straight line.

Returns the straight line for this ProfilePanel.

Return

ImageFigure

Returns the straight line for this ProfilePanel.

Double `getBegin()`

Returns the begin of the straight line in pixel coordinates.

Returns the begin of the straight line associated with this ProfilePanel in pixel coordinates.

Return

Double

Returns the begin of the straight line associated with this ProfilePanel in pixel coordinates.

Double `getEnd()`

Returns the end of the straight line in pixel coordinates.

Returns the end of the straight line associated with this ProfilePanel in pixel coordinates.

Return

Double

Returns the end of the straight line associated with this ProfilePanel in pixel coordinates.

TaskApi `getTask()`

Returns the associated task.

Returns the task associated with this ProfilePanel.

Return

TaskApi

Returns the task associated with this ProfilePanel.

Map `getMap()`

Returns the associated map.

Returns the map associated with this ProfilePanel.

Map `getMap()`**Return**[Map](#)

Returns the map associated with this ProfilePanel.

SiteEventHandler `getHandler()`

Returns the associated site event handler.

Returns the site event handler associated with this ProfilePanel.

Return**SiteEventHandler**

Returns the site event handler associated with this ProfilePanel.

PlotXY `getPlot()`

Returns the associated plot.

Returns the plot associated with this ProfilePanel.

Return**PlotXY**

Returns the plot associated with this ProfilePanel.

int `getNbOfClicks()`

Returns the number of clicks made on the image.

Returns the number of clicks made on the image for this ProfilePanel.

Return**int**

Returns the number of clicks made on the image for this ProfilePanel.

boolean `isInImage(MouseEvent me)`

Checks whether the given mouse event occurred inside the image.

Returns true if the given mouse event occurred inside the image for this ProfilePanel; false otherwise.

Argument[MouseEvent](#) **me** [INPUT, MANDATORY]**Return****boolean**

Returns true if the given mouse event occurred inside the image for this ProfilePanel; false otherwise.

boolean `isInImage(double pixX, double pixY)`

Checks whether the given pixel coordinates lie inside the image.

boolean isInImage(double pixX, double pixY)

Returns true if the given pixel coordinates lie inside the image for this ProfilePanel; false otherwise.

Arguments

double **pixX** [INPUT, MANDATORY]


double **pixY** [INPUT, MANDATORY]

Return

boolean

Returns true if the given pixel coordinates lie inside the image for this ProfilePanel; false otherwise.

2.314. ProfilePlotting

Full Name:	herschel.ia.toolbox.image.gui.ProfilePlotting
Type:	Java Class - 
Import:	from herschel.ia.toolbox.image.gui import ProfilePlotting

Description

An implementation of profile plotting. In a separate window, you can plot the

intensity along a straight line over an image. There is also shown from which sky coordinate to which sky coordinate (if available) and from which pixel coordinate to which pixel coordinate, this straight line is drawn.

API Summary

Constructors	
ProfilePlotting(ImageAnalysisToolbox toolbox)	The standard constructor for ProfilePlotting. A new window, associated
ProfilePlotting(boolean useAsComponent, ImageAnalysisToolbox toolbox)	Constructor for ProfilePlotting. When the new ProfilePlotting is
Methods	
setBegin(Double begin)	Sets the begin of the straight line associated with this
setEnd(Double end)	Sets the end of the straight line, associated with this
PlotXY getPlot()	Returns the PlotXY shown in this ProfilePlotting.
ImageFigure getLine()	Returns the straight line along which we wish to plot
Double getBegin()	Returns the begin of the drawn line in pixel coordinates.
Double getEnd()	Returns the end of the drawn line in pixel coordinates.
int getClicks()	Returns the number of valid clicks (i.e. in the image) you made.
String getSkyCoordinates()	Returns the String version of the sky coordinates
String getPixelCoordinates()	Returns the String version of the pixel coordinates of the begin.
boolean isInImage(double pixX, double pixY)	Checks whether the given pixel coordinates lies
DoubleId getPlotPoints()	

Methods
Returns the intensity of the pixels along the straight line, ArrayList <code>getImageFigures()</code>
Returns all ImageFigures used for this ProfilePlotting (the

API details

Constructors

<code>ProfilePlotting(ImageAnalysisToolbox toolbox)</code>
The standard constructor for ProfilePlotting. A new window, associated with the given ImageAnalysisToolbox is opened. It is on the image, analyzed with the given ImageAnalysisToolbox, we wish to draw a straight line, and plot the intensity along that straight line in the new ProfilePlotting window.
Argument ImageAnalysisToolbox <code>toolbox</code> [INPUT, MANDATORY]

<code>ProfilePlotting(boolean useAsComponent, ImageAnalysisToolbox toolbox)</code>
Constructor for ProfilePlotting. When the new ProfilePlotting is component-based, a window for plotting the intensity is opened; otherwise nothing will be shown.
Arguments boolean <code>useAsComponent</code> [INPUT, MANDATORY] ImageAnalysisToolbox <code>toolbox</code> [INPUT, MANDATORY]

Methods

<code>setBegin(Double begin)</code>
Sets the begin of the straight line associated with this ProfilePlotting to the given point (in mouse coordinates).
Argument Double <code>begin</code> [INPUT, MANDATORY]

<code>setEnd(Double end)</code>
Sets the end of the straight line, associated with this ProfilePlotting to the given point (in mouse coordinates).
Argument Double <code>end</code> [INPUT, MANDATORY]

<code>PlotXY getPlot()</code>
Returns the PlotXY shown in this ProfilePlotting.
Return PlotXY

PlotXY `getPlot()`

Returns the PlotXY shown in this ProfilePlotting.

ImageFigure `getLine()`

Returns the straight line along which we wish to plot the intensity in this ProfilePlotting.

Return

[ImageFigure](#)

The straight line along which we wish to plot the intensity in this ProfilePlotting.

Double `getBegin()`

Returns the begin of the drawn line in pixel coordinates.

Return

[Double](#)

Returns the begin of the drawn line in pixel coordinates.

Double `getEnd()`

Returns the end of the drawn line in pixel coordinates.

Return

[Double](#)

Returns the end of the drawn line in pixel coordinates.

int `getClicks()`

Returns the number of valid clicks (i.e. in the image) you made.

Return

int

Returns the number of valid clicks (i.e. in the image) you made.

String `getSkyCoordinates()`

Returns the String version of the sky coordinates of the begin and end of the drawn straight line.

Return

[String](#)

Returns the String version of the sky coordinates of the begin and end of the drawn straight line.

String `getPixelCoordinates()`

Returns the String version of the pixel coordinates of the begin.

Return

[String](#)

[String](#) getPixelCoordinates()

Returns the String version of the pixel coordinates of the begin and end of the drawn line.

boolean isInImage(double pixX, double pixY)

Checks whether the given pixel coordinates lies inside the image for this ProfilePlotting.

Arguments

double **pixX** [INPUT, MANDATORY]

double **pixY** [INPUT, MANDATORY]

Return

boolean

Returns true if the given pixel coordinates lie inside the image for this ProfilePlotting; false otherwise.

[DoubleId](#) getPlotPoints()

Returns the intensity of the pixels along the straight line, associated with this ProfilePlotting.

Return

[DoubleId](#)

Returns the intensity of the pixels along the drawn straight line associated with this ProfilePlotting.

[ArrayList](#) getImageFigures()


Returns all ImageFigures used for this ProfilePlotting (the straight line on the image).

Return

[ArrayList](#)

Returns all ImageFigures used for this ProfilePlotting (the straight line on the image).

2.315. ProfileTask

Full Name:	herschel.ia.toolbox.image.ProfileTask
Type:	Java Task - 
Import:	from herschel.ia.toolbox.image import ProfileTask

Description

A Task to make intensity plots along a straight line.

A Task which determines the intensity of the pixels along a straight line on an image.

API Summary

Properties
Image image [INPUT, MANDATORY, default=No default value]
Double beginX [INPUT, OPTIONAL, default=Default value : NaN]
Double beginY [INPUT, OPTIONAL, default=Default value : NaN]
Double endX [INPUT, OPTIONAL, default=Default value : NaN]
Double endY [INPUT, OPTIONAL, default=Default value : NaN]
String beginRA [INPUT, OPTIONAL, default=Default value : "begin-RA"]
String beginDec [INPUT, OPTIONAL, default=Default value : "begin-Dec"]
String endRA [INPUT, OPTIONAL, default=Default value : "endRA"]
String endDec [INPUT, OPTIONAL, default=Default value : "endDec"]
Profile profile [OUTPUT, MANDATORY, default=Default value : Profile()]

API details

Properties

Image image [INPUT, MANDATORY, default=No default value]
The image.
Double beginX [INPUT, OPTIONAL, default=Default value : NaN]
The x-pixel-coordinate of the begin of the straight line.
Double beginY [INPUT, OPTIONAL, default=Default value : NaN]
The y-pixel-coordinate of the begin of the straight line.
Double endX [INPUT, OPTIONAL, default=Default value : NaN]
The x-pixel-coordinate of the end of the straight line.
Double endY [INPUT, OPTIONAL, default=Default value : NaN]
The y-pixel-coordinate of the end of the straight line.

```
String beginRA [INPUT, OPTIONAL, default=Default value : "begin-RA"]
```

The right ascension of the begin of the straight line.

```
String beginDec [INPUT, OPTIONAL, default=Default value : "begin-Dec"]
```

The declination of the begin of the straight line.

```
String endRA [INPUT, OPTIONAL, default=Default value : "endRA"]
```

The right ascension of the end of the straight line.


```
String endDec [INPUT, OPTIONAL, default=Default value : "end-Dec"]
```

The declination of the end of the straight line.

```
Profile profile [OUTPUT, MANDATORY, default=Default value : Profile()]
```

The profile.

2.316. QRDecomposition

Full Name:	herschel.ia.numeric.toolbox.matrix.QRDecomposition
Alias:	QRDecomposition
Type:	Java Class - 
Import:	from herschel.ia.numeric.toolbox.matrix import QRDecomposition

Example

<p>Example 1: A = Double2d([[2.0, 1.0, 1.0],[4.0, -6.0, 0.0],[-2.0, 7.0, 2.0]])</p> <pre> B=Double2d([[3.0],[-8.0],[10.0] -] -) res=B.apply(QRDecomposition(A)) print res [[0.9999999999999971], [1.9999999999999978], [-0.9999999999999969]] QRD = QRDecomposition(A) leastSquaresSolutionAXeqB = B.apply(QRD) print leastSquaresSolutionAXeqB [[0.9999999999999971], [1.9999999999999978], [-0.9999999999999969]] isFullRank = QRD.isFullRank print isFullRank 1 HouseholderVectors = QRD.h print HouseholderVectors [[1.4082482904638631,0.0,0.0], [0.8164965809277261,1.4099776105529318,0.0], [-0.4082482904638631,-0.9120955864630134,2.0]] rightTriangularFactor = QRD.r print rightTriangularFactor [[-4.898979485566356,7.3484692283495345,0.408248290463863], [0.0,5.656854249492381,2.1213203435596424], [0.0,0.0,-0.5773502691896262]] orthFactor = QRD.qs print orthFactor [[-0.40824829046386313,0.7071067811865475,0.5773502691896257], [-0.8164965809277261,5.551115123125783E-17,-0.5773502691896256], [0.4082482904638631,0.7071067811865475,-0.577350269189626]] </pre>

API Summary

<p>Jython Syntax</p> <pre> A=Double2d() B=Double2d() res=B.apply(QRDecomposition(A)) QRD = QRDecomposition(A) leastSquaresSolutionAXeqB = B.apply(QRD) </pre>

Jython Syntax

```
isFullRank = QRD.isFullRank
HouseholderVectors = QRD.h
rightTriangularFactor = QRD.r
orthFactor = QRD.qs
```

Property

[Double2d](#) **A** [INPUT, MANDATORY, default=no default value]


API details

Property

[Double2d](#) **A** [INPUT, MANDATORY, default=no default value]

Input must be a Double2d or Float2d array.


2.317. Query

Full Name:	herschel.ia.pal.query.Query
Type:	Java Class - 
Import:	from herschel.ia.pal.query import Query

Example

Example 1: Example of an query: q=Query('type=='AbcProduct' and
<pre>creator=='Scott') q=Query("foo.Product", "type=='AbcProduct' and creator=='Scott'")</pre>

2.318. RadialVelocityTask

Full Name:	herschel.ia.toolbox.astro.RadialVelocityTask
Type:	Java Task - 
Import:	from herschel.ia.toolbox.astro import RadialVelocityTask
Category:	toolbox

Description

The RadialVelocity Task.

Calculates S/C radial velocity projection in the direction of the pointing, optionally including Sun to LSR and Earth to Sun velocities to help in Doppler correction of observed frequencies.

Pointing and Orbit products should be contained in an observation context processed by the SPG. The corrections applied from the data obtained from products can include:

- RadialVelocityTask.Correction.NONE
- RadialVelocityTask.Correction.SUNTOLSR
- RadialVelocityTask.Correction.EARTHOTOSUN (approx or fine (using Ephemeris))
- RadialVelocityTask.Correction.SUNANDEARTH (default)

The sign of the radial velocity is positive towards the target and negative otherwise.

Examples

Example 1: Providing an Ephemeris instance

```
from herschel.share.fltdyn.time import FineTime
from herschel.share.fltdyn.ephem import Ephemerides
orbitFile="/home/jadiaz/workspace/ia_toolbox_astro/H20090608_0001.LOE" #long
term, 2month span
ephemFile="/home/jadiaz/workspace/ia_toolbox_astro/ascp2007-2020.405" #de405,
weekly, 2month span
ephem=Ephemerides(orbitFile, ephemFile) #should be singleton, it is better to
specify a time interval
t = FineTime(pointing.startTime)
orbit = observation.getAuxiliary().getOrbitEphemeris()
pointing = observation.getPointing().getPointing()
v =
radialVelocity(pointing=pointing,orbit=orbit,time=t,ephemerides=ephem,corrections=RadialVelocit
```

Example 2: Without an Ephemeris instance, all corrections, getting value and direction

```
from herschel.share.fltdyn.time import FineTime
t = FineTime(pointing.startTime)
v = radialVelocity(pointing=pointing,orbit=orbit,time=t)
ra = radialVelocity.ra
dec = radialVelocity.dec
```

Example 3: Comparing uncorrected with corrected data

```
steps = 100
start = pointing.startTime
span= pointing.endTime -- start
step= span -/ steps;
vos=DoubleIeld() #uncorrected
ves=DoubleIeld() #corrected
ts=DoubleIeld() #time
for OD in range(steps + 1):
```

Example 3: Comparing uncorrected with corrected data

```

t = FineTime(start)
ts.append(start)
v = radialVelocity(pointing=pointing,orbit=orbit,time=t)
vos.append(v)
v =
radialVelocity(pointing=pointing,orbit=orbit,time=t,ephemerides=ephem,corrections=RadialVelocityTask.CORRECTION_SUNANDEARTH)
ves.append(v)
start = start + step
myPlot=PlotXY(ts, vos)
myPlot=PlotXY(ts, ves)

```

API Summary

Jython Syntax

```

v = radialVelocity((<storage> | <context> | <pointing>, <orbit>),
<time> [, <ephemerides>, <corrections>])

```

```

ra = radialVelocity.ra

```

```

dec = radialVelocity.dec

```

You must use either storage or context or pointing and orbit as the S/C data source.

```

Corrections: RadialVelocityTask.CORRECTION_NONE,
RadialVelocityTask.CORRECTION_SUNTOLSR,
RadialVelocityTask.CORRECTION_EARTH_TOSUN,
RadialVelocityTask.CORRECTION_SUNANDEARTH (default)

```

Properties

AuxiliaryContext storage [INPUT, OPTIONAL, default=null]
AuxiliaryContext context [INPUT, OPTIONAL, default=null]
ProductStorage pointing [INPUT, OPTIONAL, default=null]
OrbitEphemerisProduct orbit [INPUT, OPTIONAL, default=null]
FineTime time [INPUT, MANDATORY, default=null]
Ephem ephemerides [INPUT, OPTIONAL, default=null]
RadialVelocityTask.CORRECTION_TYPE corrections [INPUT, MANDATORY, default=SUNANDEARTH]
Double velocity [OUTPUT, MANDATORY, default=null]
Double ra [OUTPUT, OPTIONAL, default=null]
Double dec [OUTPUT, OPTIONAL, default=null]

Limitations

If you do not provide an Ephemerides object, EARTH_TOSUN correction will be approximate. The error without an Ephemerides object is up to +/- 40 cm/s

API details

Properties

AuxiliaryContext storage [INPUT, OPTIONAL, default=null]

The ProductStorage applicable to the observation we are working with. When storage parameter is used context, pointing and orbit cannot be used.

AuxiliaryContext context [INPUT, OPTIONAL, default=null]
The AuxiliaryContext applicable to the observation we are working with. When context parameter is used storage, pointing and orbit cannot be used.
ProductStorage pointing [INPUT, OPTIONAL, default=null]
The PointingProduct applicable to the observation we are working with. When pointing is used, orbit must be used too; storage and context cannot be used.
OrbitEphemerisProduct orbit [INPUT, OPTIONAL, default=null]
The OrbitEphemerisProduct applicable to the observation we are working with. When orbit is used, pointing must be used too: storage and context cannot be used.
FineTime time [INPUT, MANDATORY, default=null]
The time for which we want to get the radial velocity.
Ephem ephemerides [INPUT, OPTIONAL, default=null]
The Ephemerides object used in the corrections. If not used, corrections will be approximated. The error without an Ephemerides object is up to +- 40 cm/s
RadialvelocityTask.CorrectionType corrections [INPUT, MANDATORY, default=SUNANDEARTH]
The corrections flags to be applied to the velocity. Possible values are: NONE, SUNTOLSR, EARTHTOSUN, SUNANDEARTH
Double velocity [OUTPUT, MANDATORY, default=null]
The (signed) velocity projection in the pointing direction (radial velocity), in km/s Positive towards the target, negative otherwise. This is in J2000.
Double ra [OUTPUT, OPTIONAL, default=null]
RA of the pointing at t. Range = 0 .. 2*PI
Double dec [OUTPUT, OPTIONAL, default=null]
DEC of the pointing at t. Range - PI/2 .. PI/2

See also


- [RadialVelocityTask](#)
- [???](#)
- [???](#)
- [???](#)
- [???](#)

History

- 2008-10-10 - JDS: first interface candidate release
- 2008-12-12 - JDS: added precession algorithm
- 2009-02-02 - JDS: fixed degrees and radians

- 2009-02-11 - JDS: added more documentation (units)
- 2009-05-15 - JDS: moved to using a constant Sun speed vector in J2000, include Earth velocity
- 2009-06-26 - JDS: Initial task version, with tests, using Ephemeris
- 2009-07-01 - JDS: Doc corrections
- 2009-07-14 - JDS: Removed old RadialVelocity class, added Java static function with arrays, improved Task GUI
- 2009-11-02 - JDS: Doc added about error
- 2009-11-23 - JDS: Added info about sign of the radial velocity in several places
- 2009-12-09 - JDS: Recovered old radialVelocity class. Cleaned up documentation
- 2010-02-01 - JDS: Using Stumpf's model (+- 40 cm/s)
- 2010-02-25 - JDS: The SSB is moving at 20.0 km/sec towards (J2000 18h03m50.29s, +30#00'16.8") wrt the LSR : HCSS-9441

2.319. RandomGauss

Full Name:	herschel.ia.numeric.toolbox.random.RandomGauss
Alias:	RandomGauss
Type:	Java Class - 
Import:	from herschel.ia.numeric.toolbox.random import RandomGauss

Description

Generates or populates an array with normally $N(0,1)$ distributed pseudo random numbers

Examples

Example 1: example usage of a random function

```
f=RandomGauss()
f=RandomGauss(seed=72654)
y=f(x) # creates a random array with same dimensions and type as x
y=x.apply(f) # like wise, but java style
x.perform(f) # changes the values in x
```

Example 2: To generate 100 Double values normally distributed with mean=0 and sigma = 1.0

```
f = RandomGauss()
y = Double1d(100).apply(f)
# To make it with mean=mean and sigma=s
ynew = mean + y*s
```

API Summary

Jython Syntax

```
f = RandomGauss([seed=<seed>])
```

Property


[OPTIONAL seed](#) [INPUT, the seed for the random generator, default=no default value]

API details

Property

`OPTIONAL seed` [INPUT, the seed for the random generator, default=no default value]

2.320. RandomPoisson

Full Name:	herschel.ia.numeric.toolbox.random.RandomPoisson
Alias:	RandomPoisson
Type:	Java Class - 
Import:	from herschel.ia.numeric.toolbox.random import RandomPoisson

Description

Generates Poisson random numbers sequence with given mean

Examples

Example 1: example usage of a random function

```
f=RandomPoisson(20)
f=RandomPoisson(15,seed=72654)
y=f(x) # creates a random array with same dimensions and type as x
y=x.apply(f) # like wise, but java style
x.perform(f) # changes the values in x
```

Example 2: To generate 100 Double values who follow Poisson distribution with mean 5

```
f = RandomPoisson(5.0)
y = LongId(100).apply(f)
```

API Summary

Jython Syntax

```
f = RandomPoisson(<mean>,seed=<seed>)
```

Properties

MANDATORY [mean](#) [INPUT, the Poisson expectation value (mean), default=no default value]

OPTIONAL [seed](#) [INPUT, the seed for the random generator, default=no default value]


API details

Properties

MANDATORY [mean](#) [INPUT, the Poisson expectation value (mean), default=no default value]

OPTIONAL [seed](#) [INPUT, the seed for the random generator, default=no default value]

2.321. RandomUniform

Full Name:	herschel.ia.numeric.toolbox.random.RandomUniform
Alias:	RandomUniform
Type:	Java Class - 
Import:	from herschel.ia.numeric.toolbox.random import RandomUniform

Description

Generates arrays with uniformly-distributed random numbers.

Generates or populates an array with uniformly-distributed pseudo-random numbers in the range $0 \leq x < \text{max}$.

Examples

Example 1: example usage of a random function

```
x=DoubleIcd(20)
f=RandomUniform(15,seed=72654)
y=f(x)          # creates a random array with same dimensions and type as x
y=x.apply(f)    # like wise, but java style
x.perform(f)    # changes the values in x
```

Example 2: To generate 100 Double values in [0,1)

```
f = RandomUniform()
y = DoubleIcd(100).apply(f)
# To generate 100 Integer values in [0,100)
fx = RandomUniform(100.0)
yint = IntIcd(100).apply(fx)
```

API Summary

Jython Syntax

```
f = RandomUniform([<max>],seed=<seed>) # between [0,max), default
max=1.
```

Properties

[OPTIONAL max](#) [INPUT, if set the random sequence is in [0, default=max) otherwise in [0]

[OPTIONAL seed](#) [INPUT, the seed for the random generator, default=no default value]


API details

Properties

[OPTIONAL max](#) [INPUT, if set the random sequence is in [0, default=max) otherwise in [0]

[OPTIONAL seed](#) [INPUT, the seed for the random generator, default=no default value]

2.322. RangeExtractionGui

Full Name:	herschel.ia.gui.cube.RangeExtractionGui
Type:	Java Class - 
Import:	from herschel.ia.gui.cube import RangeExtractionGui

Description

A class to extract a sub Range on a cube containing a large spectrum

API Summary


Method
setPlot() initiate the PlotXY, shown in this Range extraction GUI GUI which contain the global spectrum of the GUI.

API details

Method

<code>setPlot()</code>
initiate the PlotXY, shown in this Range extraction GUI GUI which contain the global spectrum of the GUI.

2.323. RangeExtractionTask

Full Name:	herschel.ia.toolbox.cube.RangeExtractionTask
Type:	Java Task - 
Import:	from herschel.ia.toolbox.cube import RangeExtractionTask

Description

RangeExtractionTask

A class to extract a part of the cube given in parameter and to store the result as a new simplespectral-cube the extraction can concentrate on the spectral domain or the spatial domain or both

API Summary

Properties
Cube simplecube [INPUT, MANDATORY, default=no default value]
integer startindex [INPUT, MANDATORY, default=no default value]
integer endIndex [INPUT, MANDATORY, default=no default value]
boolean Crop [INPUT, MANDATORY, default=no default value]
integer Xmin [INPUT, MANDATORY, default=no default value]
integer Xmax [INPUT, MANDATORY, default=no default value]
integer Ymin [INPUT, MANDATORY, default=no default value]
integer Ymax [INPUT, MANDATORY, default=no default value]
string log [OUTPUT, MANDATORY, default=no default value]
integer error [OUTPUT, MANDATORY, default=no default value]
Cube rangeCube [OUTPUT, MANDATORY, default=no default value]
Cube specRangeCube [OUTPUT, MANDATORY, default=no default value]

API details

Properties

Cube simplecube [INPUT, MANDATORY, default=no default value]	The Cue from which we want to extract a smaller part
integer startindex [INPUT, MANDATORY, default=no default value]	The first spectral index from which we start the extraction
integer endIndex [INPUT, MANDATORY, default=no default value]	The last spectral index for the selection which we start the extraction
boolean Crop [INPUT, MANDATORY, default=no default value]	A flag for activating a spatial extraction
integer Xmin [INPUT, MANDATORY, default=no default value]	the Xmin value for the spatial extraction

<code>integer Xmax [INPUT, MANDATORY, default=no default value]</code>
--

the Xmax value for the spatial extraction

<code>integer Ymin [INPUT, MANDATORY, default=no default value]</code>
--

the Ymin value for the spatial extraction

<code>integer Ymax [INPUT, MANDATORY, default=no default value]</code>
--

the Ymax value for the spatial extraction

<code>string log [OUTPUT, MANDATORY, default=no default value]</code>

a string containing a log of the extraction when something went wrong

<code>integer error [OUTPUT, MANDATORY, default=no default value]</code>
--

a code for the error if something wrong happen
--

<code>Cube rangeCube [OUTPUT, MANDATORY, default=no default value]</code>

a SpectralSimpleCube containing the cube extracted on the spectral dimension only


<code>Cube specRangeCube [OUTPUT, MANDATORY, default=no default value]</code>

a SpectralSimpleCube containing the cube extracted on the spectral and spatial dimension only

See also

- [???](#)
- [???](#)

2.324. REBIN

Full Name:	herschel.ia.numeric.toolbox.interp.Rebin
Alias:	REBIN
Type:	Java Class - 
Import:	from herschel.ia.numeric.toolbox.interp import Rebin
Category:	numeric

Description

*

The Rebin class resizes a vector or array to dimensions given by the parameters Di. The supplied dimensions must be integral multiples or factors of the original dimension. The expansion or compression of each dimension is independent of the others, so that each dimension can be expanded or compressed by a different value. If the dimensions of the desired result are not integer multiples of the original dimensions, use the REGRID function. The SAMPLE keyword: Normally, REBIN uses bilinear interpolation when magnifying and neighborhood averaging when minifying. Set the SAMPLE keyword to use nearest neighbor sampling for both magnification and minification. If Rebin is initialized without error, the algorithm in IDL is strictly followed. If the Rebin is initialized with error, the weighted mean is used at compression when SAMPLE is not set.

Examples

Example 1: Apply Rebin on Int1d array

```
from herschel.ia.toolbox.basic.interp import Rebin;
Int1d x;
int dimension;
y = x.apply(new Rebin(dimension));
```

Example 2: Apply Rebin on Double2d array

```
from herschel.ia.toolbox.basic.interp import Rebin;
Double2d x, y;
y=x.apply(new Rebin(d1,d2));
```

Example 3: In Jython

```
from herschel.ia.toolbox.basic.interp import Rebin;
x=Int1d.range(12)
y=x.apply(Rebin(24))
or
y=Rebin(24)(x)
```

Example 4: expansion

```
signal=Double1d.range(10)
dim = 20
rebin = Rebin(dim)
rSignal=rebin(signal)
print rSignal
[0.0,0.5,1.0,1.5,2.0,2.5,3.0,3.5,4.0,4.5,5.0,5.5,6.0,6.5,7.0,7.5,8.0,8.5,9.0,9.0]
```

Example 5: calculate error

```
signal=Double1d.range(20)+1
```

Example 5: calculate error

```

inError = SQRT(signal)
dim = 10
rebin = Rebin(dim)
rSignal=rebin(signal)
p_error = rebin.getError(inError)
print p_error
[0.8660254037844387,1.322875655322954,1.6583123951777,1.9364916731037087,2.179449471770337,
2.3979157616563596,2.598076211353316,2.7838821814150108,2.958039891549808,3.1224989991991996]

```

Example 6: Use SAMPLE keyword

```

signal=DoubleI.d.range(10)
dim = 20
rebin = Rebin(dim, Rebin.SAMPLE)
rSignal=rebin(signal)
print rSignal
[0.0,0.0,1.0,1.0,2.0,2.0,3.0,3.0,4.0,4.0,5.0,5.0,6.0,6.0,7.0,7.0,8.0,8.0,9.0,9.0]

```

Example 7: Use weight error

```

signal=DoubleI.d.range(20)+1
inError = SQRT(signal)
dim = 10
rebin = Rebin(dim, inError)
rSignal=rebin(signal)
p_error = rebin.getError()
print p_error
[0.6666666666666666,1.7142857142857142,2.7272727272727275,3.7333333333333343,4.736842105263158,
5.739130434782608,6.7407407407407405,7.741935483870969,8.742857142857142,9.743589743589745]

```

API Summary

Jython Syntax

```
<y>=Rebin(<d1[,d2,d3,d4,d5]>)(<x>)
```

or

```
<y>=Rebin(<int d[]>)(<x>)
```

where <x>=input

<di> = new dimensions of output array

d[] = an array containing the new dimensions

<y>=output resampled onto dimensions.

Properties

[type **di** INPUT](#) [the dimension of the corresponding array index, MANDATORY, default=no default value]

[array of integer **x**](#) [INPUT, short, default=long or double taken the form of Int1(2)]

[type **sample Normally**](#) [REBIN uses bilinear interpolation when magnifying and neighborhood averaging, MANDATORY, default=no default value]

Limitations

The supplied dimensions must be integral multiples or factors of the original dimension. Let $f = n/m$, the ratio of the size of the original vector, X to the size of the result. $1/f$ must be an integer if $n < m$ (expansion). f must be an integer if compressing, ($n > m$).

API details

Properties

```
type di INPUT [the dimension of the corresponding array index,  
MANDATORY, default=no default value]
```

```
array of integer x [INPUT, short, default=long or double taken the  
form of Int1(2)]
```

```
type sample Normally [REBIN uses bilinear interpolation when mag-  
nifying and neighborhood averaging, MANDATORY, default=no default  
value]
```

```
when minifying. Set the SAMPLE to Rebin.SAMPLE to use nearest neighbor sampling for  
both magnification and minification. Bilinear interpolation gives higher quality results but re-  
quires more time.
```

History

- 2006-08-30 - first: version
- 2009-03-23 - error: propagation (SCR-4833) is added.

2.325. RectangleHistogramExplorer

Full Name:	herschel.ia.gui.image.RectangleHistogramExplorer
Type:	Java Class - 
Import:	from herschel.ia.gui.image import RectangleHistogramExplorer

Description

An explorer for RectangleHistogramProducts.

An explorer for RectangleHistogramProducts.

API Summary

Constructors
RectangleHistogramExplorer() The construction of a new RectangleHistogramExplorer.
RectangleHistogramExplorer(Object object) The construction of a new RectangleHistogramExplorer associated with the given object.
Methods
String getName() Returns the name.
String getDescription() Returns the description.
boolean canHandle(Class className) Checks whether this RectangleHistogramExplorer can handle objects of the given class.
setObject(Object object) Sets the object.
RectangleHistogramProduct getObject() Returns the object.
Class getVariableType() Returns the expected variable type.
JScrollPane getParameterTable() Returns the parameter table.

API details

Constructors

RectangleHistogramExplorer() The construction of a new RectangleHistogramExplorer. The construction of a new RectangleHistogramExplorer.
RectangleHistogramExplorer(Object object) The construction of a new RectangleHistogramExplorer associated with the given object.

RectangleHistogramExplorer([Object](#) object)

The construction of a new RectangleHistogramExplorer associated with the given object.

Argument

[Object](#) object [INPUT, MANDATORY]

Methods

String getName()

Returns the name.

Returns the name for this RectangleHistogramExplorer.

Return

[String](#)

Returns the name for this RectangleHistogramExplorer.

String getDescription()

Returns the description.

Returns the description for this RectangleHistogramExplorer.

Return

[String](#)

Returns the description for this RectangleHistogramExplorer.

boolean canHandle([Class](#) className)

Checks whether this RectangleHistogramExplorer can handle objects of the given class.

Returns true if this RectangleHistogramExplorer can handle objects of the given class; false otherwise.

Argument

[Class](#) className [INPUT, MANDATORY]

Return

boolean

Returns true if this RectangleHistogramExplorer can handle objects of the given class; false otherwise.

setObject([Object](#) object)

Sets the object.

Sets the object for this RectangleHistogramExplorer to the given object.

Argument

[Object](#) object [INPUT, MANDATORY]

RectangleHistogramProduct getObject()

Returns the object.

[RectangleHistogramProduct](#) getObject()

Returns the object for this RectangleHistogramExplorer.

Return

[RectangleHistogramProduct](#)

Returns the object for this RectangleHistogramExplorer.

[Class](#) getVariableType()

Returns the expected variable type.

Returns the expected variable type for this RectangleHistogramExplorer.

Return

[Class](#)

Returns the expected variable type for this RectangleHistogramExplorer.

[JScrollPane](#) getParameterTable()

Returns the parameter table.


Returns the parameter table for this RectangleHistogramExplorer.

Return

[JScrollPane](#)

Returns the parameter table for this RectangleHistogramExplorer.

2.326. RectangleHistogramPanel

Full Name:	herschel.ia.toolbox.image.gui.RectangleHistogramPanel
Type:	Java Class - 
Import:	from herschel.ia.toolbox.image.gui import RectangleHistogramPanel

Description

A panel for the RectangleHistogramTask.

A panel to serve as GUI for the RectangleHistogramTask.

API Summary

Constructor
RectangleHistogramPanel() The construction of a new RectangleHistogramPanel.
Methods
drawFigure() Draws the rectangle on the associated image.
updateFigure() Updates the associated rectangle.
trigger() Triggers the execution of the associated task.
updateHistogram() Updates the associated histogram.

API details

Constructor

<code>RectangleHistogramPanel()</code>
The construction of a new RectangleHistogramPanel.
The construction of a new RectangleHistogramPanel.

Methods

<code>drawFigure()</code>
Draws the rectangle on the associated image.
Draws the rectangle on the image associated with this RectangleHistogramPanel.
<code>updateFigure()</code>
Updates the associated rectangle.
Updates the rectangle associated with this RectangleHistogramPanel.

trigger()

Triggers the execution of the associated task.
--

Triggers the execution of the task associated with this RectangleHistogramPanel.
--

updateHistogram()

Updates the associated histogram.

Updates the histogram associated with this RectangleHistogramPanel.

2.327. RectangleHistogramProduct

Full Name:	herschel.ia.dataset.image.RectangleHistogramProduct
Type:	Java Class - 
Import:	from herschel.ia.dataset.image import RectangleHistogramProduct

Description

A class to deal with the results of a rectangle histogram.

API Summary

Constructor
<p>RectangleHistogramProduct()</p> <p>The constructor of a new RectangleHistogramProduct.</p>
Methods
<p>setUpperLeftCorner(double upperLeftX, double upperLeftY)</p> <p>Sets the upper left corner for this RectangleHistogramProduct</p>
<p>setUpperLeftCorner(double upperLeftX, double upperLeftY, String upperLeftRA, String upperLeftDec)</p> <p>Sets the upper left corner for this RectangleHistogramProduct</p>
<p>setDimensions(double widthPixels, double heightPixels)</p> <p>Sets the dimensions for this RectangleHistogramProduct to the</p>
<p>setDimensions(double widthPixels, double heightPixels, double widthArcsec, double heightArcsec)</p> <p>Sets the dimensions for this RectangleHistogramProduct to the</p>
<p>DoubleId getUpperLeftCornerPixelCoordinates()</p> <p>Returns the upper left corner for this RectangleHistogramProduct in</p>
<p>StringId getUpperLeftCornerSkyCoordinates()</p> <p>Returns the upper left corner for this RectangleHistogramProduct in</p>
<p>double getWidthPixels()</p> <p>Returns the width for this RectangleHistogramProduct in pixels.</p>
<p>double getWidthArcsec()</p> <p>Returns the width for this RectangleHistogramProduct in arcsec.</p>
<p>double getHeightPixels()</p> <p>Returns the height for this RectangleHistogramProduct in pixels.</p>
<p>double getHeightArcsec()</p> <p>Returns the height for this RectangleHistogramProduct in arcsec.</p>

API details

Constructor

RectangleHistogramProduct()
The constructor of a new RectangleHistogramProduct.

Methods

setUpperLeftCorner(double upperLeftX, double upperLeftY)

Sets the upper left corner for this RectangleHistogramProduct to the given pixel coordinates.

Arguments

double **upperLeftX** [INPUT, MANDATORY]

double **upperLeftY** [INPUT, MANDATORY]

setUpperLeftCorner(double upperLeftX, double upperLeftY, [String](#) upperLeftRA, [String](#) upperLeftDec)

Sets the upper left corner for this RectangleHistogramProduct to the given pixel and sky coordinates.

Arguments

double **upperLeftX** [INPUT, MANDATORY]

double **upperLeftY** [INPUT, MANDATORY]

[String](#) **upperLeftRA** [INPUT, MANDATORY]

[String](#) **upperLeftDec** [INPUT, MANDATORY]

setDimensions(double widthPixels, double heightPixels)

Sets the dimensions for this RectangleHistogramProduct to the given dimensions in pixels.

Arguments

double **widthPixels** [INPUT, MANDATORY]

double **heightPixels** [INPUT, MANDATORY]

setDimensions(double widthPixels, double heightPixels, double widthArcsec, double heightArcsec)

Sets the dimensions for this RectangleHistogramProduct to the given dimensions in pixels and arcsec.

Arguments

double **widthPixels** [INPUT, MANDATORY]

double **heightPixels** [INPUT, MANDATORY]

double **widthArcsec** [INPUT, MANDATORY]

double **heightArcsec** [INPUT, MANDATORY]

[DoubleId](#) **getUpperLeftCornerPixelCoordinates()**

Returns the upper left corner for this RectangleHistogramProduct in pixel coordinates.

Return

[DoubleId](#)

[DoubleId](#) getUpperLeftCornerPixelCoordinates()

Returns the upper left corner for this RectangleHistogramProduct in pixel coordinates.

[StringId](#) getUpperLeftCornerSkyCoordinates()

Returns the upper left corner for this RectangleHistogramProduct in sky coordinates.

Return

[StringId](#)

Returns the upper left corner for this RectangleHistogramProduct in sky coordinates.

double getWidthPixels()

Returns the width for this RectangleHistogramProduct in pixels.

Return

double

Returns the width for this RectangleHistogramProduct in pixels.

double getWidthArcsec()

Returns the width for this RectangleHistogramProduct in arcsec.

Return

double

Returns the width for this RectangleHistogramProduct in arcsec.

double getHeightPixels()

Returns the height for this RectangleHistogramProduct in pixels.

Return

double

Returns the height for this RectangleHistogramProduct in pixels.

double getHeightArcsec()

Returns the height for this RectangleHistogramProduct in arcsec.

Return

double

Returns the height for this RectangleHistogramProduct in arcsec.

2.328. RectangleHistogramTask

Full Name:	herschel.ia.toolbox.image.RectangleHistogramTask
Type:	Java Task - 
Import:	from herschel.ia.toolbox.image import RectangleHistogramTask

Description

A Task to make a histogram within a rectangle.

A Task to make a histogram of a region of interest, which is bounded by a rectangle.

API Summary

Properties
Image image [INPUT, MANDATORY, default=No default value]
Double lowCut [INPUT, OPTIONAL, default=Default value : NaN]
Double highCut [INPUT, OPTIONAL, default=Default value : NaN]
Integer bins [INPUT, MANDATORY, default=Default value : new Integer(2000)]
RectangleHistogramProduct histogram [OUTPUT, MANDATORY, default=No default value]
Double minX [INPUT, OPTIONAL, default=Default value : NaN]
Double minY [INPUT, OPTIONAL, default=Default value : NaN]
String minRA [INPUT, OPTIONAL, default=Default value : "minRA"]
String minDec [INPUT, OPTIONAL, default=Default value: "minDec"]
Double widthPixels [INPUT, OPTIONAL, default=Default value : NaN]
Double heightPixels [INPUT, OPTIONAL, default=Default value : NaN]
Double widthArcsec [INPUT, OPTIONAL, default=Default value : NaN]
Double heightArcsec [INPUT, OPTIONAL, default=Default value : NaN]


API details

Properties

Image image [INPUT, MANDATORY, default=No default value]
The image.
Double lowCut [INPUT, OPTIONAL, default=Default value : NaN]
The lower cut level.
Double highCut [INPUT, OPTIONAL, default=Default value : NaN]
The upper cut level.
Integer bins [INPUT, MANDATORY, default=Default value : new Integer(2000)]
The number of bins.

RectangleHistogramProduct histogram [OUTPUT, MANDATORY, default=No default value]
The histogram.
Double minX [INPUT, OPTIONAL, default=Default value : NaN]
The x-pixel-coordinate of the upper left corner of the rectangle.
Double minY [INPUT, OPTIONAL, default=Default value : NaN]
The y-pixel-coordinate of the upper left corner of the rectangle.
String minRA [INPUT, OPTIONAL, default=Default value : "minRA"]
The right ascension of the upper left corner of the rectangle.
String minDec [INPUT, OPTIONAL, default=Default value: "minDec"]
The declination of the upper left corner of the rectangle.
Double widthPixels [INPUT, OPTIONAL, default=Default value : NaN]
The width of the rectangle in pixels.
Double heightPixels [INPUT, OPTIONAL, default=Default value : NaN]
The height of the rectangle in pixels.
Double widthArcsec [INPUT, OPTIONAL, default=Default value : NaN]
The width of the rectangle in arcsec.
Double heightArcsec [INPUT, OPTIONAL, default=Default value : NaN]
The height of the rectangle in arcsec.

2.329. RectangularSkyAperturePhotometry-Explorer

Full Name:	herschel.ia.gui.image.RectangularSkyAperturePhotometryExplorer
Type:	Java Class - 
Import:	from herschel.ia.gui.image import RectangularSkyAperturePhotometryExplorer

Description

An explorer for RectangularSkyAperturePhotometryProducts.

An explorer for RectangularSkyAperturePhotometryProducts.

API Summary

Constructors
<p>RectangularSkyAperturePhotometryExplorer()</p> <p>The construction of a new RectangularSkyAperturePhotometryExplorer.</p>
<p>RectangularSkyAperturePhotometryExplorer(Object object)</p> <p>The construction of a new RectangularSkyAperturePhotometryExplorer associated with the given object.</p>
Methods
<p>boolean canHandle(Class className)</p> <p>Checks whether this RectangularSkyAperturePhotometryExplorer can handle objects of the given class.</p>
<p>RectangularSkyAperturePhotometryProduct getObject()</p> <p>Returns the object.</p>
<p>Class getVariableType()</p> <p>Returns the expected variable type.</p>
<p>JScrollPane getParameterTable()</p> <p>Returns the parameter table.</p>
<p>JPanel getPlotPanel()</p> <p>Returns the plot panel.</p>

API details

Constructors

<p>RectangularSkyAperturePhotometryExplorer()</p> <p>The construction of a new RectangularSkyAperturePhotometryExplorer.</p> <p>The construction of a new RectangularSkyAperturePhotometryExplorer.</p>
<p>RectangularSkyAperturePhotometryExplorer(Object object)</p> <p>The construction of a new RectangularSkyAperturePhotometryExplorer associated with the given object.</p>

RectangularSkyAperturePhotometryExplorer(Object object)

The construction of a new RectangularSkyAperturePhotometryExplorer associated with the given object.

Argument

[Object](#) **object** [INPUT, MANDATORY]

Methods

boolean canHandle(Class className)

Checks whether this RectangularSkyAperturePhotometryExplorer can handle objects of the given class.

Returns true if this RectangularSkyAperturePhotometryExplorer can handle objects of the given class; false otherwise.

Argument

[Class](#) **className** [INPUT, MANDATORY]

Return

boolean

Returns true if this RectangularSkyAperturePhotometryExplorer can handle objects of the given class; false otherwise.

RectangularSkyAperturePhotometryProduct getObject()

Returns the object.

Returns the object for this RectangularSkyAperturePhotometryExplorer.

Return

[RectangularSkyAperturePhotometryProduct](#)

Returns the object for this RectangularSkyAperturePhotometryExplorer.

Class getVariableType()

Returns the expected variable type.

Returns the expected variable type for this RectangularAperturePhotometryExplorer.

Return

[Class](#)

Returns the expected variable type for this RectangularAperturePhotometryExplorer.

JScrollPane getParameterTable()

Returns the parameter table.

Constructs and returns the parameter table for this RectangularSkyAperturePhotometryExplorer.

Return

[JScrollPane](#)

Returns the parameter table for this RectangularSkyAperturePhotometryExplorer.

[JPanel](#) `getPlotPanel()`


Returns the plot panel.

Constructs and returns the plot panel for this `RectangularSkyAperturePhotometryExplorer`.

Return**[JPanel](#)**

Returns the plot panel for this `RectangularSkyAperturePhotometryExplorer`.

2.330. RectangularSkyAperturePhotometry-Panel

Full Name:	herschel.ia.toolbox.image.gui.RectangularSkyAperturePhotometryPanel
Type:	Java Class - 
Import:	from herschel.ia.toolbox.image.gui import RectangularSkyAperturePhotometryPanel

Description

A panel for the RectangularSkyAperturePhotometryTask.

A panel to serve as GUI for the RectangularSkyAperturePhotometryTask.

API Summary

Constructor
RectangularSkyAperturePhotometryPanel() The constructio of a new RectangularSkyAperturePhotometryPanel.
Methods
JPanel getAperturesPanel() Returns the apertures panel.
JComponent getSkyApertureComponent() Returns the panel where it is explained how to specify the rectangular sky aperture.
drawRectangle() Allows to draw a rectangle on the associated image.
int getNbOfRectanglePoints() Returns the number of confirmed points for the rectangle.
increaseNbOfRectanglePoints() Increases the number of confirmed rectangle points.
drawFigures() Draws the figures on the image.
moveConfirmedFigures() Allows the user to drag the figures bounding the apertures.
clearSkyAperture() Clears the rectangular sky aperture.

API details

Constructor

RectangularSkyAperturePhotometryPanel()
The constructio of a new RectangularSkyAperturePhotometryPanel.
The construction of a new RectangularSkyAperturePhotometryPanel.

Methods

JPanel getAperturesPanel ()
Returns the apertures panel.
Returns the panel where to specify the target radius and where it is explained how to specify the rectangular sky aperture for this RectangularSkyAperturePhotometryPanel.
Return
JPanel
Returns the panel where to specify the target radius and where it is explained to specify the rectangular sky aperture for this RectangularSkyAperturePhotometryPanel.
JComponent getSkyApertureComponent ()
Returns the panel where it is explained how to specify the rectangular sky aperture.
Returns the panel where it is explained how to specify the rectangular sky aperture for this RectangularSkyAperturePhotometryPanel.
Return
JComponent
Returns the panel where it is explained how to specify the rectangular sky aperture for this RectangularSkyAperturePhotometryPanel.
drawRectangle ()
Allows to draw a rectangle on the associated image.
Allows to draw a rectangle on the image associated with this RectangularSkyAperturePhotometryPanel.
int getNbOfRectanglePoints ()
Returns the number of confirmed points for the rectangle.
Returns the number of points that were confirmed before for the rectangle on the image for this RectangularSkyAperturePhotometryPanel.
Return
int
Returns the number of points that were confirmed before for the rectangle on the image for this RectangularSkyAperturePhotometryPanel.
increaseNbOfRectanglePoints ()
Increases the number of confirmed rectangle points.
Increases the number of rectangle points that were confirmed before on the image for this RectangularSkyAperturePhotometryPanel.
drawFigures ()
Draws the figures on the image.

drawFigures ()

Draws the figures on the image for this RectangularSkyAperturePhotometryPanel.
--

moveConfirmedFigures ()

Allows the user to drag the figures bounding the apertures.


Allows the user to drag the figures bounding the apertures for this RectangularSkyAperturePhotometryPanel.
--

clearSkyAperture ()

Clears the rectangular sky aperture.

Clears the rectangular sky aperture for this RectangularSkyAperturePhotometryPanel.

2.331. RectangularSkyAperturePhotometryProduct

Full Name:	herschel.ia.dataset.image.RectangularSkyAperturePhotometryProduct
Type:	Java Class - 
Import:	from herschel.ia.dataset.image import RectangularSkyAperturePhotometryProduct

Description

A class to deal with the results of aperture photometry with a rectangular sky aperture.

API Summary

Constructor	
RectangularSkyAperturePhotometryProduct()	The constructor of a new RectangularSkyAperturePhotometryProduct.
Methods	
setUpperLeftCorner(double upperLeftX, double upperLeftY)	Sets the upper left corner for this RectangularSkyAperturePhotometryProduct
setUpperLeftCorner(double upperLeftX, double upperLeftY, String upperLeftRA, String upperLeftDec)	Sets the upper left corner for this RectangularSkyAperturePhotometryProduct.
setDimensions(double widthPixels, double heightPixels)	Sets the dimensions for this RectangularSkyAperturePhotometryProduct to the
setDimensions(double widthPixels, double heightPixels, double widthArcsec, double heightArcsec)	Sets the dimensions for this RectangularSkyAperturePhotometryProduct to the
DoubleId getUpperLeftCornerPixelCoordinates()	Returns the upper left corner for this RectangularSkyAperturePhotometryProduct in
StringId getUpperLeftCornerSkyCoordinates()	Returns the upper left corner for this RectangularSkyAperturePhotometryProduct
double getWidthPixels()	Returns the width for this RectangularSkyAperturePhotometryProduct
double getWidthArcsec()	Returns the width for this RectangularSkyAperturePhotometryProduct
double getHeightPixels()	Returns the height for this RectangularSkyAperturePhotometryProduct
getHeightArcsec()	Returns the height for this RectangularSkyAperturePhotometryProduct

Miscellaneous

Aperture specific information is stored in meta data, all other data (results of calculations) are stored in TableDatasets.

API details

Constructor

RectangularSkyAperturePhotometryProduct()

The constructor of a new RectangularSkyAperturePhotometryProduct.

Methods

setUpperLeftCorner(double upperLeftX, double upperLeftY)

Sets the upper left corner for this RectangularSkyAperturePhotometryProduct

to the given pixel coordinates.

Arguments

double **upperLeftX** [INPUT, MANDATORY]

double **upperLeftY** [INPUT, MANDATORY]

setUpperLeftCorner(double upperLeftX, double upperLeftY, [String](#) upperLeftRA, [String](#) upperLeftDec)

Sets the upper left corner for this RectangularSkyAperturePhotometryProduct.

Arguments

double **upperLeftX** [INPUT, MANDATORY]

double **upperLeftY** [INPUT, MANDATORY]

[String](#) **upperLeftRA** [INPUT, MANDATORY]

[String](#) **upperLeftDec** [INPUT, MANDATORY]

setDimensions(double widthPixels, double heightPixels)

Sets the dimensions for this RectangularSkyAperturePhotometryProduct to the

given dimensions in pixels.

Arguments

double **widthPixels** [INPUT, MANDATORY]

double **heightPixels** [INPUT, MANDATORY]

setDimensions(double widthPixels, double heightPixels, double widthArcsec, double heightArcsec)

Sets the dimensions for this RectangularSkyAperturePhotometryProduct to the

given dimensions in pixels and arcsec.

Arguments

double **widthPixels** [INPUT, MANDATORY]

double **heightPixels** [INPUT, MANDATORY]

double **widthArcsec** [INPUT, MANDATORY]

double **heightArcsec** [INPUT, MANDATORY]

[DoubleId](#) getUpperLeftCornerPixelCoordinates()

Returns the upper left corner for this RectangularSkyAperturePhotometryProduct in

[Double1d](#) getUpperLeftCornerPixelCoordinates()

pixel coordinates.

Return

[Double1d](#)

Returns the upper left corner for this RectangularSkyAperturePhotometryProduct in pixel coordinates.

[String1d](#) getUpperLeftCornerSkyCoordinates()

Returns the upper left corner for this RectangularSkyAperturePhotometryProduct

in sky coordinates.

Return

[String1d](#)

Returns the upper left corner for this RectangularSkyAperturePhotometryProduct in sky coordinates.

double getWidthPixels()

Returns the width for this RectangularSkyAperturePhotometryProduct

in pixels.

Return

double

Returns the width for this RectangularSkyAperturePhotometryProduct in pixels.

double getWidthArcsec()

Returns the width for this RectangularSkyAperturePhotometryProduct

in arcsec.

Return

double

Returns the width for this RectangularSkyAperturePhotometryProduct in arcsec.

double getHeightPixels()

Returns the height for this RectangularSkyAperturePhotometryProduct

in pixels.

Return

double

Returns the height for this RectangularSkyAperturePhotometryProduct in pixels.


getHeightArcsec()

Returns the height for this RectangularSkyAperturePhotometryProduct

<code>getHeightArcsec()</code>

in arcsec.

2.332. RectangularSkyAperturePhotometry-Task

Full Name:	herschel.ia.toolbox.image.RectangularSkyAperturePhotometryTask
Type:	Java Task - 
Import:	from herschel.ia.toolbox.image import RectangularSkyAperturePhotometryTask

Description

A Task for aperture photometry.

A Task for aperture photometry, using a circular target aperture and a rectangular sky aperture.

API Summary

Properties
Image image [INPUT, MANDATORY, default=No default value]
Double centerX [INPUT, OPTIONAL, default=Default value : NaN]
Double centerY [INPUT, OPTIONAL, default=Default value : NaN]
String centerRA [INPUT, OPTIONAL, default=Default value : "center-RA"]
String centerDEC [INPUT, OPTIONAL, default=Default value : "centerDec"]
Double radiusPixels [INPUT, OPTIONAL, default=Default value : NaN]
Double radiusArcsec [INPUT, OPTIONAL, default=Default value : NaN]
Integer algorithm [INPUT, MANDATORY, default=Default value : 4]
RectangularSkyAperturePhotometryProduct result [OUTPUT, MANDATORY, default=No default value]
Double minX [INPUT, OPTIONAL, default=Default value : NaN]
Double minY [INPUT, OPTIONAL, default=Default value : NaN]
String minRA [INPUT, OPTIONAL, default=Default value : "upperLeft-RA"]
String minDec [INPUT, OPTIONAL, default=Default value : "upper-LeftDec"]
Double widthPixels [INPUT, OPTIONAL, default=Default value : NaN]
Double heightPixels [INPUT, OPTIONAL, default=Default value : NaN]
Double widthArcsec [INPUT, OPTIONAL, default=Default value : NaN]
Double heightArcsec [INPUT, OPTIONAL, default=Default value : NaN]

API details

Properties

Image image [INPUT, MANDATORY, default=No default value]
The image.

Double centerX [INPUT, OPTIONAL, default=Default value : NaN]
The x-pixel-coordinate of the target center.
Double centerY [INPUT, OPTIONAL, default=Default value : NaN]
The y-pixel-coordinate of the target center.
String centerRA [INPUT, OPTIONAL, default=Default value : "centerRA"]
The right ascension of the target center.
String centerDEC [INPUT, OPTIONAL, default=Default value : "centerDec"]
The declination of the target center.
Double radiusPixels [INPUT, OPTIONAL, default=Default value : NaN]
The target radius in pixels.
Double radiusArcsec [INPUT, OPTIONAL, default=Default value : NaN]
The target radius in arcsec.
Integer algorithm [INPUT, MANDATORY, default=Default value : 4]
The sky estimation algorithm.
RectangularSkyAperturePhotometryProduct result [OUTPUT, MANDATORY, default=No default value]
The result.
Double minX [INPUT, OPTIONAL, default=Default value : NaN]
The input x-pixel-coordinate of the upper left corner of the rectangular sky aperture.
Double minY [INPUT, OPTIONAL, default=Default value : NaN]
The input y-pixel-coordinate of the upper left corner of the rectangular sky aperture.
String minRA [INPUT, OPTIONAL, default=Default value : "upperLeftRA"]
The input right ascension of the upper left corner of the rectangular sky aperture.
String minDec [INPUT, OPTIONAL, default=Default value : "upperLeftDec"]
The input declination of the upper left corner of the rectangular sky aperture.
Double widthPixels [INPUT, OPTIONAL, default=Default value : NaN]
The input width of the rectangular sky aperture in pixels.
Double heightPixels [INPUT, OPTIONAL, default=Default value : NaN]
The input height of the rectangular sky aperture in pixels.


Double widthArcsec [INPUT, OPTIONAL, default=Default value : NaN]

The input width of the rectangular sky aperture in arcsec.

Double heightArcsec [INPUT, OPTIONAL, default=Default value : NaN]

The input height of the rectangular sky aperture in arcsec.

2.333. Regrid

Full Name:	herschel.ia.numeric.toolbox.interp.Regrid
Alias:	Regrid
Type:	Java Class - 
Import:	from herschel.ia.numeric.toolbox.interp import Regrid

Description

The Regrid class shrinks or expands the size of an array by an

arbitrary amount. This class is similar to "Rebin" in that it can resize a one, two, or three dimensional array. "Rebin", however, requires that the new array size must be an integer multiple of the original size. Regrid will resize an array to any arbitrary size (Rebin is somewhat faster, however). Rebin averages multiple points when shrinking an array, while Regrid just resamples the array. The returned array has the same number of dimensions as the original array and is of the same data type.

Example

Example 1: Use Regrid

```
#1-d example: resize a 1d array 10 elements long to one that is 6 elements long
x = Double1d([0,1,2,3,4,5,6,7,8,9])
y = Regrid(6)(x)
print x # [0.0,1.0,2.0,3.0,4.0,5.0,6.0,7.0,8.0,9.0]
print y # [0.0,1.8,3.6,5.4,7.2,9.0]
# do the same, but this time use a different scheme
y = Regrid(6,Regrid.NEAREST_NEIGHBOR)(x)
print y # [0.0,2.0,4.0,5.0,7.0,9.0]
#resize a 2d array 2x3 elements long to one that is 5x8 elements long
x = Double2d([[2.2, 3.1, 11.9],[109.2, 135.7, 210.8]])
y = Regrid(5,8,Regrid.LINEAR)(x)
#note the first column resamples between 2.2 and 109.2.
#i.e. going from 2 points to 5. this can be checked by -:
print y[:,0] # [2.2,28.95,55.7,82.45,109.2]
# or concentrating on the first row, which goes
# from 2.2 to 11.9 in three steps, the out is resampled to 8 points.
print y[0:] #
[2.2,2.4571428571428573,2.7142857142857144,2.9714285714285715,4.357142857142857,6.8714285714285715,9.257142857142857,11.642857142857143]
```

API Summary

Constructors
Regrid(int[] d)
Regrid(int[] d, String interpMethod)
Regrid(int[] d, String interpMethod, String allowExtrapolation)
Regrid(int d1)
Regrid(int d1, String interpMethod)
Regrid(int d1, String interpMethod, String allowExtrapolation)
Regrid(int d1, int d2)
Regrid(int d1, int d2, String interpMethod)
Regrid(int d1, int d2, String interpMethod, String allowExtrapolation)

Constructors
Regrid(int d1, int d2, int d3)
Regrid(int d1, int d2, int d3, String interpMethod)
Regrid(int d1, int d2, int d3, String interpMethod, String allowExtrapolation)
Regrid(int d1, int d2, int d3, int d4)
Regrid(int d1, int d2, int d3, int d4, String interpMethod)
Regrid(int d1, int d2, int d3, int d4, String interpMethod, String allowExtrapolation)
Regrid(int d1, int d2, int d3, int d4, int d5)
Regrid(int d1, int d2, int d3, int d4, int d5, String interpMethod)
Regrid(int d1, int d2, int d3, int d4, int d5, String interpMethod, String allowExtrapolation)

Limitations

The x, INPUT, array of Int(n)d, Short(n)d, Long(n)d, Float(n)d or Double(n)d, (1<= n <=5)

Miscellaneous

Synopsis:

resample rank=d.length array to new dimensions in d.

y = Regrid(d[])(x)

resamples a one to five dimension array using default (linear) interpolation.

y = Regrid(d1,...,d5)(x)

resamples a one to five dimension array using linear interpolation.

y = Regrid(d1,...,d5,Regrid.LINEAR)(x)

resamples an array using nearest neighbor interpolation.

y = Regrid(d1,...,d5,Regrid.NEAREST_NEIGHBOR)(x)

resamples an array using cubic spline interpolation.

y = Regrid(d1,...,d5,Regrid.CUBIC_SPLINE)(x)

allows Extrapolation.

y = Regrid(d1,...,d5,Regrid.LINEAR,Regrid.EXTRAPOLATION)(x)

API details

Constructors

Regrid(int[] d)
Argument
int[] d [INPUT, MANDATORY, default=no default value]

Regrid(int[] d)
array of Int(n)d, Short(n)d, Long(n)d, Float(n)d or Double(n)d, (1<= n <=5), MANDATORY.
Errors
UnsupportedOperationException
Bad dimension array! It should be 1<= d <=5.

Regrid(int[] d, String interpMethod)
Arguments
int[] d [INPUT, MANDATORY, default=no default value] array of Int(n)d, Short(n)d, Long(n)d, Float(n)d or Double(n)d, (1<= n <=5), MANDATORY.
String interpMethod [INPUT, NOT_MANDATORY, default=Regrid.LINEAR] allowable interpolation schemes are: CUBLIC_SPLINE, NEAREST_NEIGHBOR, and LINEAR (which is the default).
Errors
UnsupportedOperationException
Bad dimension array! It should be 1<= d <=5.

Regrid(int[] d, String interpMethod, String allowExtrapolation)
Arguments
int[] d [INPUT, MANDATORY, default=no default value] array of Int(n)d, Short(n)d, Long(n)d, Float(n)d or Double(n)d, (1<= n <=5), MANDATORY.
String interpMethod [INPUT, NOT_MANDATORY, default=Regrid.LINEAR] allowable interpolation schemes are: CUBLIC_SPLINE, NEAREST_NEIGHBOR, and LINEAR (which is the default).
String allowExtrapolation [INPUT, NOT_MANDATORY, default=Regrid.EXTRAPOLATION] default is no extrapolation allowed.
Errors
UnsupportedOperationException
Bad dimension array! It should be 1<= d <=5.

Regrid(int d1)
Argument
int d1 [INPUT, MANDATORY, default=no default value] d1 input integer

Regrid(int d1, String interpMethod)
Arguments
int d1 [INPUT, MANDATORY] String interpMethod [INPUT, NOT_MANDATORY, default=Regrid.LINEAR]

Regrid(int d1, [String](#) interpMethod)

allowable interpolation schemes are: CUBLIC_SPLINE, NEAREST_NEIGHBOR, and LINEAR (which is the default).

Regrid(int d1, [String](#) interpMethod, [String](#) allowExtrapolation)

Arguments

int **d1** [INPUT, MANDATORY]

[String](#) **interpMethod** [INPUT, NOT_MANDATORY, default=Regrid.LINEAR]

allowable interpolation schemes are: CUBLIC_SPLINE, NEAREST_NEIGHBOR, and LINEAR (which is the default).

[String](#) **allowExtrapolation** [INPUT, NOT_MANDATORY, default=Regrid.EXTRAPOLATION]

default is no extrapolation allowed.

Regrid(int d1, int d2)

Arguments

int **d1** [INPUT, MANDATORY, default=no default value]

d1 input integer

int **d2** [INPUT, MANDATORY, default=no default value]

d2 input integer

Regrid(int d1, int d2, [String](#) interpMethod)

Arguments

int **d1** [INPUT, MANDATORY, default=no default value]

d1 input integer

int **d2** [INPUT, MANDATORY, default=no default value]

d2 input integer

[String](#) **interpMethod** [INPUT, NOT_MANDATORY, default=Regrid.LINEAR]

allowable interpolation schemes are: CUBLIC_SPLINE, NEAREST_NEIGHBOR, and LINEAR (which is the default)

Regrid(int d1, int d2, [String](#) interpMethod, [String](#) allowExtrapolation)

Arguments

int **d1** [INPUT, MANDATORY, default=no default value]

d1 input integer

int **d2** [INPUT, MANDATORY, default=no default value]

d2 input integer

[String](#) **interpMethod** [INPUT, NOT_MANDATORY, default=Regrid.LINEAR]

allowable interpolation schemes are: CUBLIC_SPLINE, NEAREST_NEIGHBOR, and LINEAR (which is the default).

[String](#) **allowExtrapolation** [INPUT, NOT_MANDATORY, default=Regrid.EXTRAPOLATION]

Regrid(int d1, int d2, [String](#) interpMethod, [String](#) allowExtrapolation)

default is no extrapolation allowed.

Regrid(int d1, int d2, int d3)

Arguments

int **d1** [INPUT, MANDATORY, default=no default value]
 d1 input integer

int **d2** [INPUT, MANDATORY, default=no default value]
 d2 input integer

int **d3** [INPUT, MANDATORY, default=no default value]
 d3 input integer

Regrid(int d1, int d2, int d3, [String](#) interpMethod)

Arguments

int **d1** [INPUT, MANDATORY, default=no default value]
 d1 input integer

int **d2** [INPUT, MANDATORY, default=no default value]
 d2 input integer

int **d3** [INPUT, MANDATORY, default=no default value]
 d3 input integer

[String](#) **interpMethod** [INPUT, NOT_MANDATORY,
 default=Regrid.LINEAR]
 allowable interpolation schemes are: CUBLIC_SPLINE, NEAREST_NEIGHBOR, and
 LINEAR (which is the default).

Regrid(int d1, int d2, int d3, [String](#) interpMethod, [String](#) allowExtrapolation)

Arguments

int **d1** [INPUT, MANDATORY, default=no default value]
 d1 input integer

int **d2** [INPUT, MANDATORY, default=no default value]
 d2 input integer

int **d3** [INPUT, MANDATORY, default=no default value]
 d3 input integer

[String](#) **interpMethod** [INPUT, NOT_MANDATORY,
 default=Regrid.LINEAR]
 allowable interpolation schemes are: CUBLIC_SPLINE, NEAREST_NEIGHBOR, and
 LINEAR (which is the default).

[String](#) **allowExtrapolation** [INPUT, NOT_MANDATORY,
 default=Regrid.EXTRAPOLATION]
 default is no extrapolation allowed.

Regrid(int d1, int d2, int d3, int d4)

Arguments

int **d1** [INPUT, MANDATORY, default=no default value]

Regrid(int d1, int d2, int d3, int d4)

d1 input integer
 int **d2** [INPUT, MANDATORY, default=no default value]
 d2 input integer
 int **d3** [INPUT, MANDATORY, default=no default value]
 d3 input integer
 int **d4** [INPUT, MANDATORY, default=no default value]
 d4 input integer

Regrid(int d1, int d2, int d3, int d4, [String](#) interpMethod)

Arguments

int **d1** [INPUT, MANDATORY, default=no default value]
 d1 input integer
 int **d2** [INPUT, MANDATORY, default=no default value]
 d2 input integer
 int **d3** [INPUT, MANDATORY, default=no default value]
 d3 input integer
 int **d4** [INPUT, MANDATORY, default=no default value]
 d4 input integer
[String](#) **interpMethod** [INPUT, NOT_MANDATORY,
 default=Regrid.LINEAR]
 allowable interpolation schemes are: CUBLIC_SPLINE, NEAREST_NEIGHBOR, and
 LINEAR (which is the default).

**Regrid(int d1, int d2, int d3, int d4, [String](#) interpMethod,
[String](#) allowExtrapolation)**

Arguments

int **d1** [INPUT, MANDATORY, default=no default value]
 d1 input integer
 int **d2** [INPUT, MANDATORY, default=no default value]
 d2 input integer
 int **d3** [INPUT, MANDATORY, default=no default value]
 d3 input integer
 int **d4** [INPUT, MANDATORY, default=no default value]
 d4 input integer
[String](#) **interpMethod** [INPUT, NOT_MANDATORY,
 default=Regrid.LINEAR]
 allowable interpolation schemes are: CUBLIC_SPLINE, NEAREST_NEIGHBOR, and
 LINEAR (which is the default).
[String](#) **allowExtrapolation** [INPUT, NOT_MANDATORY,
 default=Regrid.EXTRAPOLATION]
 default is no extrapolation allowed.

Regrid(int d1, int d2, int d3, int d4, int d5)

Arguments

Regrid(int d1, int d2, int d3, int d4, int d5)

```

int d1 [INPUT, MANDATORY, default=no default value]
    d1 input integer
int d2 [INPUT, MANDATORY, default=no default value]
    d2 input integer
int d3 [INPUT, MANDATORY, default=no default value]
    d3 input integer
int d4 [INPUT, MANDATORY, default=no default value]
    d4 input integer
int d5 [INPUT, MANDATORY, default=no default value]
    d5 input integer

```

Regrid(int d1, int d2, int d3, int d4, int d5, [String](#) interpMethod)**Arguments**

```

int d1 [INPUT, MANDATORY, default=no default value]
    d1 input integer
int d2 [INPUT, MANDATORY, default=no default value]
    d2 input integer
int d3 [INPUT, MANDATORY, default=no default value]
    d3 input integer
int d4 [INPUT, MANDATORY, default=no default value]
    d4 input integer
int d5 [INPUT, MANDATORY, default=no default value]
    d5 input integer
String interpMethod [INPUT, NOT_MANDATORY,
    default=Regrid.LINEAR]
    allowable interpolation schemes are: CUBLIC_SPLINE, NEAREST_NEIGHBOR, and
    LINEAR (which is the default).

```

Regrid(int d1, int d2, int d3, int d4, int d5, [String](#) interpMethod, [String](#) allowExtrapolation)**Arguments**

```

int d1 [INPUT, MANDATORY, default=no default value]
    d1 input integer
int d2 [INPUT, MANDATORY, default=no default value]
    d2 input integer
int d3 [INPUT, MANDATORY, default=no default value]
    d3 input integer
int d4 [INPUT, MANDATORY, default=no default value]
    d4 input integer
int d5 [INPUT, MANDATORY, default=no default value]
    d5 input integer
String interpMethod [INPUT, NOT_MANDATORY,
    default=Regrid.LINEAR]

```

```
Regrid(int d1, int d2, int d3, int d4, int d5, String interp-  
Method, String allowExtrapolation)
```

allowable interpolation schemes are: CUBIC_SPLINE, NEAREST_NEIGHBOR, and LINEAR (which is the default).

[String](#) allowExtrapolation [INPUT, NOT_MANDATORY,
default=Regrid.EXTRAPOLATION]

default is no extrapolation allowed.


See also

- [REBIN](#)

History

- 2009-10-10 - Modified: by Lijun HCSS-8078 Since I did not find the algorithm used, I checked IDL and found Congrid funtion in IDL. I believe that Regrid used the same algorithm as Congrid. Therefore I made the following changes:
 1. Deprecated EXTRAPOLATION
 2. Set extrapolation as a default behavior
 3. Introduce MINUS_ONE to reflect the fact the IDL conter-partnership.
 4. When MINUS_ONE is true, I enforced the new coordinate at newDim -1 is the same as oldDim-1. Mathematically, $\text{RegridFact} = (\text{oldDim} - 1) / (\text{newDim} - 1)$; $\text{new_coordinate}(i) = i * \text{RegridFact}$ when $i = \text{newDim} - 1$, $\text{new_coordinate}(\text{newDim}-1) = \text{oldDim} - 1$. But due to the numerically error, there may be some small detla value introduced. Therefore, I need to enforce it to oldDim -1
 5. Replaced several false to `_allowExtrapolation` in calling, for example, `RegridFor1dArray(d1, _interp, _allowExtrapolation)`;
 6. Change the RegridConsts class to interface

2.334. REPEAT

Full Name:	herschel.ia.numeric.toolbox.array.Repeat
Alias:	REPEAT
Type:	Java Class - 
Import:	from herschel.ia.numeric.toolbox.array import Repeat

Description

Creates a new array which contains the specified repetitions of the original array.

Example

Example 1: Apply REPEAT on a Int1d
<pre>x=Int1d([0,1,2,3,4,5,6,7,8,9,10,11]) print REPEAT(x,3) # Int2d = [[0,1,2,3,4,5,6,7,8,9,10,11], [0,1,2,3,4,5,6,7,8,9,10,11],[0,1,2,3,4,5,6,7,8,9,10,11]]</pre>

API Summary


Jython Syntax
<y>=REPEAT(<x>,<repetitions=1>)
Properties
any array of any rank x [INPUT, MANDATORY, default=no default value]
integer repetitions [INPUT, MANDATORY, default=no default value]
array y [OUTPUT, MANDATORY, default=no default value]

API details

Properties

any array of any rank x [INPUT, MANDATORY, default=no default value]
The input array must be an array of rank 1 to 4 (both included)
integer repetitions [INPUT, MANDATORY, default=no default value]
The repetitions to apply to this array. It must be greater than 0.
array y [OUTPUT, MANDATORY, default=no default value]
Returns an increased dimensional array with the specified repetitions.

2.335. ReplaceFreqRanges

Full Name:	herschel.ia.toolbox.spectrum.ReplaceFreqRanges
Type:	Java Task - 
Import:	from herschel.ia.toolbox.spectrum import ReplaceFreqRanges

Description

Task for inserting and/or replacing ranges within the segments of a container.

Various different modes are available for how the replacement / insertion should be processed. For further details see the mode-parameter below.

Example

Example 1: from jide
<pre>from herschel.ia.toolbox.spectrum import ReplaceFreqRanges replace = ReplaceFreqRanges() replacedSlices1 = replace(ds=spectra, by=slices, mode="replace") replacedSlices2 = replace(ds=spectra, by=slices, mode="avg") replacedSlices3 = replace(ds=spectra, by=slices, mode="insert")</pre>

API Summary

Properties
SpectrumContainer ds [INPUT, OPTIONAL, default=No default value.]
SpectrumContainer by [INPUT, OPTIONAL, default=No default value.]
SpectrumContainer result [OUTPUT, OPTIONAL, default=No default value.]
String mode [INPUT, OPTIONAL, default=replace.]

API details

Properties

SpectrumContainer ds [INPUT, OPTIONAL, default=No default value.]
The container in which the spectra of the other input container should be inserted.
SpectrumContainer by [INPUT, OPTIONAL, default=No default value.]
The container with the slices that should be inserted into the first input container.
SpectrumContainer result [OUTPUT, OPTIONAL, default=No default value.]
The output container.
String mode [INPUT, OPTIONAL, default=replace.]
A mode specifying how the module should actually do the replacement. Three options are available:


`String mode [INPUT, OPTIONAL, default=replace.]`

- "replace": Resample the slices to be inserted to the frequency grid of the original container (ds) and replace the original ranges by the resampled slices.
- "avg": Resample the slices to be inserted to the frequency grid of the original container (ds) and replace the original ranges by the average of the original ranges and the resampled slices.
- "insert": Insert brute-force the slices by replacing the ranges of the original spectra by matching the frequency ranges - irrespective of the frequency grid.

History

- 2008-03-27 - meli: initial.

2.336. ResampleFrequency

Full Name:	herschel.ia.toolbox.spectrum.ResampleFrequency
Type:	Java Task - 
Import:	from herschel.ia.toolbox.spectrum import ResampleFrequency

Description

Task for resampling the flux values included in a spectrum container with respect

to a modified frequency grid. Different schemes are available - all should conserve total flux. The schemes are typically a filter method in combination with an interpolation scheme and an integration scheme to assure flux conservation.

The standard (default) scheme uses a box filter in combination with trapezoidal integration and linear interpolation. An alternative scheme consists of using a box filter in combination with euler integration and nearest neighbor interpolation. Further schemes to be implemented will be based on a Gaussian filter.

Flags and weights are processed if available:

- For the weights, the same integration/interpolation scheme is applied as is used for the flux. For the weights, we always assume that the weights are defined 'per channel'. (For the flux, we generally assume that it is given on a 'per frequency' basis - if not otherwise stated by the 'isDensity' quantity.
- The flag for an output channel is defined by combining the flags of all the input channels that are considered (for the given output channel) by an bitwise OR logic. This allows to recover all the distinct flag values that are involved in the computation of the given channel.

For output channels that are not supported by any input channels, we set for flux and weights values 'NaN' and as flag 'NotSampled' (64).

Other attributes found in the spectra are copied to the result container without any change.

The meta data parameter 'resolution_resampled' is adjusted by the new resampling width (w) according to the formula $\text{SQRT}(r_0 * r_0 + w * w)$ where r_0 is the resolution_resampled before applying the resampling operation. In case there was no 'resolution_resampled' but just a 'resolution' parameter a new parameter 'resolution_resampled' parameter is added as a copy of the 'resolution'. In case the resampling width is expressed at the velocity scale (km/s) it is transformed to a frequency scale by making use of the reference frequency (looked up from the meta data with key 'referenceFrequency'). If some information is missing (e.g. the reference frequency or its unit) the parameter 'resolution_resampled' is set to NaN.

Example

Example 1: from Jide:

```
resampled1 = resample(ds=spectra, density=True, resolution=1.0)
resampled1 = resample(ds=spectra, density=True, resolution=1.0, unit="MHz")
resampled1 = resample(ds=spectra, density=True, resolution=0.001, unit="GHz")
resampled2 = resample(ds=spectra, density=True, scheme="euler", resolution=1.0)
grid = [4000.0 + 2.0*Double1d.range(500), 5000.0 + 5.0*Double1d.range(200),
6000.0 + 1.0*Double1d.range(1000), 7000.0 + 2.0*Double1d.range(500)]
resampled3 = resample(ds=spectra, density=True, grid=grid)
for i in range(len(grid)):
    grid[i] = grid[i] * 0.001
resampled3 = resample(ds=spectra, density=True, grid=grid, unit="GHz")
```

API Summary

Properties
<code>SpectrumContainer ds</code> [INPUT, MANDATORY, default=no default value.]
<code>SpectrumContainer result</code> [OUTPUT, MANDATORY, default=no default value.]
<code>String scheme</code> [INPUT, OPTIONAL, default=no default value.]
<code>Double1d[] Double1d grid</code> [INPUT, OPTIONAL, default=no default value.]
<code>double resolution</code> [INPUT, OPTIONAL, default=no default value.]
<code>String unit</code> [INPUT, OPTIONAL, default=no default value.]
<code>Boolean density</code> [INPUT, OPTIONAL, default=no default value.]

API details

Properties

<code>SpectrumContainer ds</code> [INPUT, MANDATORY, default=no default value.]
The input container to be resampled.
<code>SpectrumContainer result</code> [OUTPUT, MANDATORY, default=no default value.]
The output container with the re-sampled spectra.
<code>String scheme</code> [INPUT, OPTIONAL, default=no default value.]
The scheme to be adopted for the resampling - available are "standard" (or equivalently "trapezoidal") and "simple" (or equivalently "euler").
<ul style="list-style-type: none"> The "standard" scheme combines a linear interpolation with a trapezoidal integration scheme. The "simple" scheme combines a nearest neighbor interpolation and an Euler integration scheme.
Both scheme are constructed such that total flux is conserved.
<code>Double1d[] Double1d grid</code> [INPUT, OPTIONAL, default=no default value.]
The new frequency grid the spectra should be resampled to. It is specified as a <code>Double1d</code> for each of the sub-segments. The grid is specified in the same units as the original frequency data. The grid may be non-equidistant.
<code>double resolution</code> [INPUT, OPTIONAL, default=no default value.]
In case no output frequency grid is specified, the task creates an output grid with this given resolution ($2 * \text{width}$) by determining, for each sub-segment, the smallest minimum frequency and the largest maximum frequency. The resolution is expressed in the unit specified as <code>unit-parameter</code> . If no <code>unit-parameter</code> is specified the unit of the original frequency grid is assumed. In case the width is specified as a negative number the resulting grid will be in decreasing order.

<code>String</code> unit [INPUT, OPTIONAL, default=no default value.]
--

The unit the resolution or the grid is expressed in.
--


<code>Boolean</code> density [INPUT, OPTIONAL, default=no default value.]
--

If set to true the flux data is treated as a flux density (per frequency unit) - otherwise the flux is treated as a per channel quantity.

History

- 2008-01-29 - meli: initial.
- 2008-03-16 - meli: major refactoring of the toolbox.
- 2008-09-30 - meli: trapezoidal scheme used as standard (default); some refactoring (interpolation scheme implicit in integration scheme).
- 2008-10-03 - meli: flag, weights processing, javadoc, ...

2.337. RESHAPE

Full Name:	herschel.ia.numeric.toolbox.basic.Reshape
Alias:	RESHAPE
Type:	Java Class - 
Import:	from herschel.ia.numeric.toolbox.basic import Reshape

Description

Creates a new array of the same type as the input array but with a different shape.

The input array can be an array of any rank, and the output array has a rank defined by the specified shape argument.

The shape argument must be an array of integers such as [3,4], and the resulting array must have the same number of elements as the input array. If the shape argument is not specified, the result will always be a 1d-array.

Example

Example 1: Apply RESHAPE on a Int1d

```
x=Int1d.range(12)
print RESHAPE(x,[2,6]) # [[0,1,2,3,4,5],[6,7,8,9,10,11]]
print RESHAPE(x,[6,2]) # [[0,1],[2,3],[4,5],[6,7],[8,9],[10,11]]
print RESHAPE(x,[2,2,3]) # [[[0,1,2],[3,4,5]],[[6,7,8],[9,10,11]]]
print RESHAPE(TRANSPOSE(RESHAPE(x,[4,3])))
# [0,3,6,9,1,4,7,10,2,5,8,11]
```

API Summary

Jython Syntax

```
<y>=RESHAPE(<x>, [<shape>])
```

Properties

any array of integral type **x** [INPUT, MANDATORY, default=no default value]

array of integers **shape** [INPUT, NOT_MANDATORY, default=1d]

boolean array or a boolean **y** [OUTPUT, MANDATORY, default=no default value]

API details

Properties

any array of integral type **x** [INPUT, MANDATORY, default=no default value]

The input array can be an array of any rank

array of integers **shape** [INPUT, NOT_MANDATORY, default=1d]

The shape argument must be an array of integers such as [3,4]. If the shape argument is not specified, the result will always be a 1d-array.


`boolean array or a boolean y [OUTPUT, MANDATORY, default=no default value]`

Returns an array with the same number of elements as the input array and with shape equal to is specified or 1d if not specified.

See also

- [CONCATENATE](#)

2.338. restore

Full Name:	herschel.ia.toolbox.util.RestoreTask
Alias:	restore
Type:	Java Task - 
Import:	from herschel.ia.toolbox.util import RestoreTask
Category:	task

Description

Restore variables from a file to a jython session

The restore task is used to restore one or more variables from a file to a jython session. The restore task is used to save one or more variable from a previously saved file into the specified namespace.

Please note that when used for restoring values using the local() option the result might be unpredictable if not executed as first command. Please note that only Serializable variable can be saved.

Examples

Example 1: restore variables (names and values) from a file

```
restore("foo.sav")
```

Example 2: restore variables (names and values) from a file into a function namespace

```
def my_function():
    restore("test.sav", space=locals())
    print locals()
```

API Summary

Jython Syntax

```
restore(<filename>[, <space>])
```

Properties

[String filename](#) [INPUT, MANDATORY, default=No default value]

[PyStringMap space](#) [INPUT, OPTIONAL, default=globals()]

Limitations

Only Serializable variable saved can be restored, when using for restoring variables using locals result might be unpredictable if not executed as first command.

Miscellaneous

No miscellaneous

API details

Properties

[String filename](#) [INPUT, MANDATORY, default=No default value]

The name of the file where the variables along with their values are stored

`PyStringMap space [INPUT, OPTIONAL, default=globals()]`

A jython dictionary (PyStringMap) containing the namespace to load the variables into, The choices available are:

1. "locals()" for reading variables from the local namespace (i.e. from where the function is used)
2. "globals()" for reading variables from the module where the function is used

When no space is specified the top level namespace is updated.


See also

- [save](#)

History

- 2004-07-13 - NdC: first release.
- 2009-01-21 - JDS: cleanup
- 2009-09-30 - JSS: modifiers are created through `getCustomModifiers` (SCR HCSS-8253)

2.339. resume

Full Name:	herschel.ia.toolbox.util.ResumeTask
Alias:	resume
Type:	Java Task - 
Import:	from herschel.ia.toolbox.util import ResumeTask
Category:	task

Description

Pause the execution of jython code

The resume task is used to resume the execution of jython after a call of pause. The resume can only be performed interactively from the debugger window and not programmatically as jython doesn't execute code just after a pause call.

Example

Example 1: resume the jython execution (only available from the debug command window)

```
resume()
```

Limitations

Resume can only be performed interactively from the debugger window

Miscellaneous

No miscellaneous


See also

- [resume](#)

History

- 2007-10-11 - NdC: first release.

2.340. REVERSE

Full Name:	herschel.ia.numeric.toolbox.basic.Reverse
Alias:	REVERSE
Type:	Java Class - 
Import:	from herschel.ia.numeric.toolbox.basic import Reverse

Description

Reverses the order of the elements in an array of rank 1

Example

Example 1: Apply REVERSE on a Double1d

```
print REVERSE(Double1d.range(3)) # [2.0,1.0,0.0]
```

API Summary

Jython Syntax

```
<y>=REVERSE(<x>)
```

Properties

[any array of rank 1 **x** \[INPUT, MANDATORY, default=no default value\]](#)

[boolean array or a boolean **y** \[OUTPUT, MANDATORY, default=no default value\]](#)

API details

Properties

any array of rank 1 **x [INPUT, MANDATORY, default=no default value]**

The input array can be an array of rank 1

boolean array or a boolean **y [OUTPUT, MANDATORY, default=no default value]**

Returns an array with the same number of elements as the input array and with the reverse order.

2.341. RgbSimpleImage

Full Name:	herschel.ia.dataset.image.RgbSimpleImage
Type:	Java Class - 
Import:	from herschel.ia.dataset.image import RgbSimpleImage
Category:	Image

Description

A Product describing three color images.

RgbSimpleImage is a Product describing three color images.

Example

Example 1: Creating a RgbSimpleImage
<pre>s = RgbSimpleImage() s.setRedImage(myInt2dRedImage) s.setGreenImage(myInt2dGreenImage) s.setBlueImage(myInt2dBlueImage)</pre>

API Summary

Constructors
RgbSimpleImage() The standard constructor
RgbSimpleImage(String description) Constructor with a description
RgbSimpleImage(Int2d red, Int2d green, Int2d blue) The standard constructor
RgbSimpleImage(RgbSimpleImage copy) Copy constructor
Methods
setRedImage(Int2d red) Sets the red part of the image.
setGreenImage(Int2d green) Sets the green part of the image.
setBlueImage(Int2d blue) Sets the blue part of the image.
RgbImage getPreview(float res) Returns a preview of the image
int getHeight() Returns the height.
Int2d getRedImage() returns the red image as a Int2d.

Methods
Int2d getGreenImage() returns the green image as an Int2d.
Int2d getBlueImage() returns the blue image as an Int2d.
Unit getUnit() Returns the unit.
int getWidth() Returns the width
setUnit(Unit unit) Sets the unit.
int[] getDimensions() Returns the dimension.
Wcs getWcs() Returns the World Coordinates System.
setWcs(Wcs wcs) Sets the world coordinates system.

API details

Constructors

RgbSimpleImage()
The standard constructor
A constructor which creates a standard RgbSimpleImage.
Example
Typical example on how to create an RgbSimpleImage.
<pre>image=RgbSimpleImage()</pre>

RgbSimpleImage(String description)
Constructor with a description
Creates an RgbSimpleImage with a given description.
Argument
<code>String description</code> [INPUT, MANDATORY]

RgbSimpleImage(Int2d red, Int2d green, Int2d blue)
The standard constructor
A constructor which creates a standard RgbSimpleImage. The standard SimpleImage consists of 3 Numeric2ds for the image
Arguments
<code>Int2d red</code> [INPUT, MANDATORY]
<code>Int2d green</code> [INPUT, MANDATORY]

RgbSimpleImage(Int2d red, Int2d green, Int2d blue)

[Int2d](#) **blue** [INPUT, MANDATORY]

Example

Typical example on how to create an RgbSimpleImage.

```
image=RgbSimpleImage(redImage, greenImage, blueImage, description="ngc
6992", wcs=wcs)
```

RgbSimpleImage(RgbSimpleImage copy)

Copy constructor

Constructor which makes a copy from an existent RgbSimpleImage.

Argument

[RgbSimpleImage](#) **copy** [INPUT, MANDATORY]

Methods

setRedImage(Int2d red)

Sets the red part of the image.

Sets the red part of the image.

Argument

[Int2d](#) **red** [INPUT, MANDATORY]

setGreenImage(Int2d green)

Sets the green part of the image.

Sets the green part of the image.

Argument

[Int2d](#) **green** [INPUT, MANDATORY]

setBlueImage(Int2d blue)

Sets the blue part of the image.

Sets the blue part of the image.

Argument

[Int2d](#) **blue** [INPUT, MANDATORY]

RgbImage getPreview(float res)

Returns a preview of the image

Returns a new RgbSimpleImage, but with a lower resolution if res is smaller than 1. A higher resolution is returned, if res is larger than 1. The spatial regridding is based on linear interpolation which will, in general, not be suitable for scientific purposes.

Argument

float **res** [INPUT, MANDATORY]

Return

RgbImage

RgbImage `getPreview(float res)`

The preview of the image

int `getHeight()`

Returns the height.

Returns the height of this RgbSimpleImage.

Return**int**

The height of this RgbSimpleImage.

Int2d `getRedImage()`

returns the red image as a Int2d.

Returns the image as an Int2d containing the data of the red image.

Return**Int2d**

The red image as an Int2d

Int2d `getGreenImage()`

returns the green image as an Int2d.

Returns the image as an Int2d containing the data of the green image.

Return**Int2d**

The green image as an Int2d

Int2d `getBlueImage()`

returns the blue image as an Int2d.

Returns the image as an Int2d containing the data of the blue image.

Return**Int2d**

The blue image as an Int2d

Unit `getUnit()`

Returns the unit.

Returns the unit of the image. The unit of the errors of this image is the same as the unit of the image.

Return**Unit**

The unit

int getWidth()

Returns the width

Returns the width of this SimpleImage.

Return

int

The width of this SimpleImage.

setUnit(Unit unit)

Sets the unit.

Sets the unit of the image.

Argument

Unit **unit** [INPUT, MANDATORY]

int[] getDimensions()

Returns the dimension.

Returns the dimension (width, height) of this RgbSimpleImage.

Return

int[]

The dimension (width, height) of this RgbSimpleImage.

[Wcs](#) getWcs()

Returns the World Coordinates System.

Returns the World Coordinates System of the image.

Return

[Wcs](#)

The World Coordinates System of the image.

setWcs([WCS](#) wcs)


Sets the world coordinates system.

Sets the world coordinates system of the image.

Argument

[Wcs](#) **wcs** [INPUT, MANDATORY]

2.342. Rotate

Full Name:	herschel.ia.numeric.toolbox.basic.Rotate
Type:	Java Class - 
Import:	from herschel.ia.numeric.toolbox.basic import Rotate

Description

Rotates numeric 2d and 3d arrays.

The result is an array of the double type. The returned array is usually larger than the original because a rotation of a rectangle array creates empty corners. The corners are filled with 0 by default. (This does not happen with rotations around multiples of 90 degrees).

The rotation is always around the center of the array. That means the result contains the complete rotated array without any cuts.

Possible interpolation is nearest neighbor, bilinear und bicubic.

The interpolation quality depends on the rotation angle and on the interpolation method. It can be measured as a sum over all array indexes. An example is presented with an 200X255 Int2d array. It is constructed as follows:

```
array = Int2d();
for i in range(200):
array.append(Int1d.range(255),1)
```

The sum is 6477000

With a rotation angle of 54 deg clockwise the sums are:

Nearest neighbor: 6477268 (0.004% accurate)

Bilinear: 6447345 (0.46% accurate)

Bicubic: 6451030 (0.40% accurate)

Example

Example 1: apply Rotate to a Int2d array

```
Clockwise nearest neighbour rotation (default)
result = Rotate(54.0)( array -)
Counterclockwise nearest neighbour rotation
result = Rotate(-54.0)( array -)
Clockwise bicubic rotation
result = Rotate(54.0, Rotate.BICUBIC) ( array -) or
result = Rotate(54.0, 3) ( array -)
Counterclockwise bilinear rotation
result = Rotate(54.0, Rotate.BILINEAR, False) ( array -)
array = Int2d();
for i in range(5):
    array.append(Int1d.range(5),1)
print array
[
[0,0,0,0,0],
[1,1,1,1,1],
[2,2,2,2,2],
[3,3,3,3,3],
[4,4,4,4,4]
```

Example 1: apply Rotate to a Int2d array

```

]
print Rotate(90) ( array -)
[
[4.0,3.0,2.0,1.0,0.0],
[4.0,3.0,2.0,1.0,0.0],
[4.0,3.0,2.0,1.0,0.0],
[4.0,3.0,2.0,1.0,0.0],
[4.0,3.0,2.0,1.0,0.0]
]

```

API Summary

Jython Syntax

```

<y> = <x>.apply( Rotate(angle, interpolation method, clockwise) )
or
<y> = Rotate(angle, interpolation method, clockwise)( <x> )

```

Properties

any 2- or 3d array of integral type **x** [INPUT, MANDATORY, default=no default value]

the rotation angle in degrees (360 based) **angle** [INPUT, MANDATORY, default=no default value]

the interpolation method **interpolation method** [INPUT, NOT_MANDATORY, default=Rotate.NEAREST_NEIGHBOR]

the direction of rotation **clockwise** [INPUT, NOT_MANDATORY, default=true]

DoubleNd array (same rank as the input array) **y** [OUTPUT, MANDATORY, default=no default value]

API details

Properties

any 2- or 3d array of integral type **x** [INPUT, MANDATORY, default=no default value]

the rotation angle in degrees (360 based) **angle** [INPUT, MANDATORY, default=no default value]

the angle can be positive or negative

the interpolation method **interpolation method** [INPUT, NOT_MANDATORY, default=Rotate.NEAREST_NEIGHBOR]

the possible values for the interpolation method are

Rotate.NEAREST_NEIGHBOR = 1

Rotate.BILINEAR = 2

Rotate.BICUBIC = 3


the direction of rotation clockwise [INPUT, NOT_MANDATORY, default=true]

clockwise is a boolean and has the value True for clockwise rotation and False for counter-clockwise rotation.

DoubleNd array (same rank as the input array) y [OUTPUT, MANDATORY, default=no default value]

Returns a Double2d array, if the input was a 2d array. Returns a Double3d array if the input was a 3d array. 3d arrays are treated as a stack of 2d arrays. Rotation is done around the depth (3rd) dimension.

2.343. rotate

Full Name:	herschel.ia.toolbox.image.RotateTask
Alias:	rotate
Type:	Java Task - 
Import:	from herschel.ia.toolbox.image import RotateTask
Category:	task/image

Description

A Task that rotates images.

A task which rotates an image. The Wcs is also adapted. It is possible to use 4 types of interpolation :

- **INTERP_NEAREST** : nearest-neighbor interpolation. Nearest-neighbor interpolation is simply pixel copying
- **INTERP_BILINEAR** : Bilinear interpolation requires a neighborhood extending one pixel to the right and below the central sample. If the fractional subsample position is given by (xfrac, yfrac), the resampled pixel value will be:

```
(1 -- yfrac) * [(1 -- xfrac)*s00 + xfrac*s01] + yfrac * [(1 -- xfrac)*s10 + xfrac*s11]
```

A neighborhood extending one sample to the right of, and one sample below the central sample is required to perform bilinear interpolation. This implementation maintains equal subsampleBits in x and y.

- **INTERP_BICUBIC** : InterpolationBicubic is a subclass of Interpolation that performs interpolation using the piecewise cubic polynomial:

```
r(x) = (a + 2)|x|^3 -- (a + 3)|x|^2 + 1 -, 0 <= -|x| < 1
r(x) = a|x|^3 -- 5a|x|^2 + 8a|x| -- 4a -, 1 <= -|x| < 2
r(x) = 0, otherwise
```

with 'a' set to -0.5. This definition is also sometimes known as "cubic convolution", using the parameter 'a' recommended by Rifman. (Reference: Digital Image Warping, George Wolberg, 1990, pp 129-131, IEEE Computer Society Press, ISBN 0-8186-8944-7)

A neighborhood extending one sample to the left of and above the central sample, and two samples to the right of and below the central sample is required to perform bicubic interpolation.

- **INTERP_BICUBIC_2** : InterpolationBicubic2 is a subclass of Interpolation that performs interpolation using the piecewise cubic polynomial:

```
r(x) = (a + 2)|x|^3 -- (a + 3)|x|^2 + 1 -, 0 <= -|x| < 1
r(x) = a|x|^3 -- 5a|x|^2 + 8a|x| -- 4a -, 1 <= -|x| < 2
r(x) = 0, otherwise
```

with 'a' set to -1.0. This definition is also sometimes known as "cubic convolution", using the parameter 'a' recommended by Keys. This interpolator may produce somewhat sharper results than InterpolationBicubic, but that result is image dependent. (Reference: Digital Image Warping, George Wolberg, 1990, pp 129-131, IEEE Computer Society Press, ISBN 0-8186-8944-7)

A neighborhood extending one sample to the left of and above the central sample, and two samples to the right of and below the central sample is required to perform bicubic interpolation.

When no interpolation is set, BiLinear interpolation (**INTERP_BILINEAR**) is taken as standard. **INTERP_BICUBIC** and **INTERP_BICUBIC_2** need an extra parameter (subsample precision, in bits)

to be set. If the subsample precision is not set explicitly, the value will be 16. The errors are not calculated. At the moment, the same operation is executed on the errors. The Wcs is not always calculated exactly.

Examples

Example 1: Rotating an image over 12.2 degrees (counterclockwise for astronomical image, clockwise for other images)

```
rotate(image = im, angle = 12.2)
```

Example 2: Rotating an image over 12.2 degrees (counterclockwise for astronomical image, clockwise for other images), using Bicubic interpolation, using 32 bits

```
rotate(image = im, angle = 12.2, interpolation = Rotate.INTERP_BICUBIC,
       subsampleBits = 32)
```

API Summary

Jython Syntax

```
rotate(image, 12.2, Rotate.INTERP_BICUBIC)
```

Properties

[Image **image**](#) [INPUT, MANDATORY, default=No default value]

[float **angle**](#) [INPUT, OPTIONAL, default=0.0]

[int **interpolation**](#) [INPUT, OPTIONAL, default=Rotate.INTERP_NEAREST]

[int **subsampleBits**](#) [INPUT, OPTIONAL, default=16.0]

[Image **rotatedImage**](#) [OUTPUT, MANDATORY, default=No default value]

API details

Properties

Image image [INPUT, MANDATORY, default=No default value]

The Image to rotate.

float angle [INPUT, OPTIONAL, default=0.0]

The amount of degrees to rotate the image (counterclockwise for astronomical image, clockwise for other images).

int interpolation [INPUT, OPTIONAL, default=Rotate.INTERP_NEAREST]

The type of interpolation to use (INTERP_NEAREST, INTERP_NEAREST, INTERP_BICUBIC, INTERP_BICUBIC_2).


int subsampleBits [INPUT, OPTIONAL, default=16.0]

The number of bits to use for the interpolation (only if interpolation is INTERP_BICUBIC or INTERP_BICUBIC_2).

Image rotatedImage [OUTPUT, MANDATORY, default=No default value]

The rotated Image.

2.344. RotateTaskSignatureComponent

Full Name:	herschel.ia.toolbox.image.gui.RotateTaskSignatureComponent
Type:	Java Class - 
Import:	from herschel.ia.toolbox.image.gui import RotateTaskSignatureComponent

Description

A task dialog for the RotateTask.

A task dialog to serve as GUI for the RotateTask.

API Summary

Constructor
RotateTaskSignatureComponent() The construction of a new RotateTaskSignatureComponent.
Methods
check() Validates the current modifications.
clear() Clears the current modifications.
getParameters() Returns a map of parameters and modifiers storing the values selected by the user.
setVariableSelection(VariableSelection selection) Setting the variable selection.
setSignature(SignatureApi sig) Setting the signature.
JComponent getComponent() Returns the component.

API details

Constructor


<code>RotateTaskSignatureComponent()</code>
The construction of a new RotateTaskSignatureComponent.
The construction of a new RotateTaskSignatureComponent.

Methods

<code>check()</code>
Validates the current modifications.
Validates the current modifications for this RotateTaskSignatureComponent.

clear()
Clears the current modifications. Clears the current modifications for this RotateTaskSignatureComponent.
getParameters()
Returns a map of parameters and modifiers storing the values selected by the user. Returns a map of parameters and modifiers storing the values selected by the user for this RotateTaskSignatureComponent.
setVariableSelection(VariableSelection selection)
Setting the variable selection. Sets the given selection as the variable selection for this RotateTaskSignatureComponent. Argument VariableSelection selection [INPUT, MANDATORY]
setSignature(SignatureApi sig)
Setting the signature. Sets the given signature as the signature for this RotateTaskSignatureComponent. Argument SignatureApi sig [INPUT, MANDATORY]
JComponent getComponent()
Returns the component. Returns the component for this RotateTaskSignatureComponent. Return JComponent Returns the component for this RotateTaskSignatureComponent.

2.345. ROUND

Full Name:	herschel.ia.numeric.toolbox.basic.Round
Alias:	ROUND
Type:	Java Class - 
Import:	from herschel.ia.numeric.toolbox.basic import Round

Description

Gives a rounded result of an array.

It can perform the operation over a decimal position (second argument) or over the integer part (no second argument or equals to '0'). For rounding to the closest integer representation, the result is functionally equivalent to adding 1/2 and taking the floor of the result. The function returns a float array for float input array and double array for any other numeric array type.

Example

Example 1: Apply ROUND on a Double1d

```
x=Double1d([-0.5001, --0.5, --0.4999, 0.4999, 0.5, 0.5001, 1.1234, 2.6236])
print ROUND(x) #[-1.0, 0.0, 0.0, 0.0, 1.0, 1.0, 1.0, 3.0]
print ROUND(x,3) #[-0.5, --0.5, --0.5, 0.5, 0.5, 0.5, 1.123, 2.624]
```

API Summary

Jython Syntax

```
<y>=ROUND(<x>[ , <n> ])
```

Properties

[any array of any rank **x** \[INPUT, MANDATORY, default=no default value\]](#)

[decimal position for rounding **n** \[INPUT, OPTIONAL, default=0\]](#)

[float or double array **y** \[OUTPUT, MANDATORY, default=no default value\]](#)

API details

Properties

any array of any rank **x [INPUT, MANDATORY, default=no default value]**

The input array can be an array of any type (except String and Complex) of any rank.

decimal position for rounding **n [INPUT, OPTIONAL, default=0]**

The decimal position for rounding. By default, it rounds to the closest integer value.


float or double array **y [OUTPUT, MANDATORY, default=no default value]**

Returns a float array for float input array and double array for any other numeric array type.

See also

- [CEIL](#)
- [FLOOR](#)

2.346. RowByRowAverageSpectrum

Full Name:	herschel.ia.toolbox.spectrum.RowByRowAverageSpectrum
Type:	Java Task - 
Import:	from herschel.ia.toolbox.spectrum import RowByRowAverageSpectrum

Description


Task for averaging two { @link SpectrumContainer } datasets on a scan-by-scan basis.

Included here for backwards compatibility reasons.

History

- 2007-08-17 - meli: initial.
- 2008-03-16 - meli: major refactoring of the toolbox.

2.347. save

Full Name:	herschel.ia.toolbox.util.SaveTask
Alias:	save
Type:	Java Task - 
Import:	from herschel.ia.toolbox.util import SaveTask
Category:	task

Description

Save variables from the jython session to a file

The save task is used to save one or more variable from the jython global namespace into the specified file. Variables can be specified with a string containing a comma separated list of variable names, if no variables are specified the entire namespace is saved.

Please note that only Serializable variable can be saved.

Examples

Example 1: save all global variable names and values

```
save("foo.sav")
```

Example 2: save specific variable names and values

```
save("foo.sav", "x,y,z")
```

Example 3: save specified variable names and values from the local namespace of a function

```
def my_function():
    x=100
    y=23
    save("test.sav", "x,y", space=locals())
my_function()
```

API Summary

Jython Syntax

```
save(<filename>[, <variable>][, <space>])
```

Properties

[String filename](#) [INPUT, MANDATORY, default=No default value]

[String variable](#) [INPUT, OPTIONAL, default=""]

[PyStringMap space](#) [INPUT, OPTIONAL, default=globals()]

Limitations

Only Serializable variables can be saved

Miscellaneous

No miscellaneous

API details

Properties

<code>String filename [INPUT, MANDATORY, default=No default value]</code>

The name of the file to store the values of variables.
--

<code>String variable [INPUT, OPTIONAL, default=""]</code>
--

The comma-separated list of variables to save.
--

<code>PyStringMap space [INPUT, OPTIONAL, default=globals()]</code>

A jython dictionary (PyStringMap) containing the namespace of the variables, The choices available are:

- | |
|---|
| <ol style="list-style-type: none">1. locals() for using variables from the local namespace (i.e. from where the function is used)2. globals() for using variables from the module where the function is used |
|---|

When no space is specified the top level namespace is updated.
--


See also

- [restore](#)

History

- 2004-07-13 - NdC: first release.
- 2009-01-14 - JDS: cleanup
- 2009-09-30 - JSS: modifiers are created through `getCustomModifiers` (SCR HCSS-8253)

2.348. saveObservation

Full Name:	herschel.ia.toolbox.util.jython.saveobs
Alias:	saveObservation
Type:	Jython Task - 
Import:	from herschel.ia.toolbox.util.jython import saveObservation
Category:	Input-Output

Description

Save an observation into a local pool.

The observation is saved into a pool on your local system. If only the observation is given as an argument, a local pool will be created in the default location, i.e. ~/.hcss/lstore. The default name of the pool is the OBSID for the given observation. To configure this you can use the following configuration properties: hcss.knownPoolLocations -> list of directories with pools hcss.dataPoolPrefix -> prefix of the pattern name of data pools, the final pattern has the form `_*` (where OD is the Observational Day) hcss.auxPoolPattern -> pattern that matches the names of auxiliary pools Note that you need to restart Hipe to change the value of these properties

By default the calibration tree is not saved with the observation. If you want to saved the calibration tree with the observation, set the argument `saveCalTree=True`.

Beware that saving an observation might take a long time since all the products associated to the observation will be saved locally, except the calibration tree.

Example

Example 1: Saving an observation from your HIPE session

```
saveObservation(obs)
saveObservation(obs, poolName="od144")
```

API Summary

Jython Syntax

```
saveObservation(obs[, verbose=False|True]
[, poolLocation=<directory>] [, poolName=<pool name>]
[, saveCalTree=False|True])
```

Properties

ObservationContext obs [INPUT, MANDATORY, default=no default]
boolean verbose [INPUT, OPTIONAL, default=False]
String poolName [INPUT, OPTIONAL, default=see description]
String poolLocation [INPUT, OPTIONAL, default=see description]
boolean saveCalTree [INPUT, OPTIONAL, default=False]

API details

Properties

<code>ObservationContext</code> obs [INPUT, MANDATORY, default=no default]
--

The observation that must be saved

<code>boolean</code> verbose [INPUT, OPTIONAL, default=False]

A flag for more verbose output.

<code>String</code> poolName [INPUT, OPTIONAL, default=see description]

The name of the pool to be used to save the requested observation

<code>String</code> poolLocation [INPUT, OPTIONAL, default=see description]

The directory where the pool for the given observation must be created
--


<code>boolean</code> saveCalTree [INPUT, OPTIONAL, default=False]

Indicates whether the calibration tree associated with the observation must be saved as well. By default the calTree is not saved in the local pool.
--

See also

- [???](#)

2.349. scale

Full Name:	herschel.ia.toolbox.image.ScaleTask
Alias:	scale
Type:	Java Task - 
Import:	from herschel.ia.toolbox.image import ScaleTask
Category:	task/image

Description

A Task to scale images.

A task to scale images. The Wcs is also adapted. It is possible to use 4 types of interpolation :

- **INTERP_NEAREST** : nearest-neighbor interpolation. Nearest-neighbor interpolation is simply pixel copying
- **INTERP_BILINEAR** : Bilinear interpolation requires a neighborhood extending one pixel to the right and below the central sample. If the fractional subsample position is given by (xfrac, yfrac), the resampled pixel value will be:

```
(1 -- yfrac) * [(1 -- xfrac)*s00 + xfrac*s01] + yfrac * [(1 -- xfrac)*s10 + xfrac*s11]
```

A neighborhood extending one sample to the right of, and one sample below the central sample is required to perform bilinear interpolation. This implementation maintains equal subsampleBits in x and y.

- **INTERP_BICUBIC** : InterpolationBicubic is a subclass of Interpolation that performs interpolation using the piecewise cubic polynomial:

```
r(x) = (a + 2)|x|^3 -- (a + 3)|x|^2 + 1 -, 0 <= -|x| < 1
r(x) = a|x|^3 -- 5a|x|^2 + 8a|x| -- 4a -, 1 <= -|x| < 2
r(x) = 0, otherwise
```

with 'a' set to -0.5. This definition is also sometimes known as "cubic convolution", using the parameter 'a' recommended by Rifman. (Reference: Digital Image Warping, George Wolberg, 1990, pp 129-131, IEEE Computer Society Press, ISBN 0-8186-8944-7)

A neighborhood extending one sample to the left of and above the central sample, and two samples to the right of and below the central sample is required to perform bicubic interpolation.

- **INTERP_BICUBIC_2** : InterpolationBicubic2 is a subclass of Interpolation that performs interpolation using the piecewise cubic polynomial:

```
r(x) = (a + 2)|x|^3 -- (a + 3)|x|^2 + 1 -, 0 <= -|x| < 1
r(x) = a|x|^3 -- 5a|x|^2 + 8a|x| -- 4a -, 1 <= -|x| < 2
r(x) = 0, otherwise
```

with 'a' set to -1.0. This definition is also sometimes known as "cubic convolution", using the parameter 'a' recommended by Keys. This interpolator may produce somewhat sharper results than InterpolationBicubic, but that result is image dependent. (Reference: Digital Image Warping, George Wolberg, 1990, pp 129-131, IEEE Computer Society Press, ISBN 0-8186-8944-7)

A neighborhood extending one sample to the left of and above the central sample, and two samples to the right of and below the central sample is required to perform bicubic interpolation.

When no interpolation is set, BiLinear interpolation (**INTERP_BILINEAR**) is taken as standard. **INTERP_BICUBIC** and **INTERP_BICUBIC_2** need an extra parameter (subsample precision, in bits)

to be set. If the subsample precision is not set explicitly, the value will be 16. The errors are not calculated. At the moment, the same operation is executed on the errors.

Examples

Example 1: Scaling an image by a factor 1.4 in x-direction, and -0.4 in y-direction (meaning flipping and scaling it by 0.4)

```
scale(image = im, x = 1.4, y = 0.4)
```

Example 2: Scaling an image by a factor 1.4 in x-direction, and 0.4 in y-direction, using Bicubic interpolation, using 32 bits

```
scale(image = im, x = 1.4, y = 0.4, interpolation = Scale.INTERP_BICUBIC,
      subsampleBits = 32)
```

API Summary

Jython Syntax

```
scale(image, 1.4, 0.4, ScaleTask.INTERP_BICUBIC, 32)
```

Properties

[Image image](#) [INPUT, MANDATORY, default=No default value]

[float x](#) [INPUT, OPTIONAL, default=1.0]

[float y](#) [INPUT, OPTIONAL, default=1.0]

[int interpolation](#) [INPUT, OPTIONAL, default=Rotate.INTERP_NEAREST]

[int subsampleBits](#) [INPUT, OPTIONAL, default=16.0]

[Image scaledImage](#) [OUTPUT, MANDATORY, default=No default value]

API details

Properties

Image image [INPUT, MANDATORY, default=No default value]

The input image that has to be scaled.

float x [INPUT, OPTIONAL, default=1.0]

The scale factor in x-direction.

float y [INPUT, OPTIONAL, default=1.0]

The scale factor in y-direction.

int interpolation [INPUT, OPTIONAL, default=Rotate.INTERP_NEAREST]

The type of interpolation to use (INTERP_NEAREST, INTERP_NEAREST, INTERP_BICUBIC, INTERP_BICUBIC_2).


int subsampleBits [INPUT, OPTIONAL, default=16.0]

The number of bits to use for the interpolation (only if interpolation is INTERP_BICUBIC or INTERP_BICUBIC_2).

Image scaledImage [OUTPUT, MANDATORY, default=No default value]
--

The scaled image.

2.350. ScaleTaskSignatureComponent

Full Name:	herschel.ia.toolbox.image.gui.ScaleTaskSignatureComponent
Type:	Java Class - 
Import:	from herschel.ia.toolbox.image.gui import ScaleTaskSignatureComponent

Description

The task dialog for the ScaleTask.

A task dialog to serve as GUI for the ScaleTask.

API Summary

Constructor
ScaleTaskSignatureComponent() The construction of a new ScaleTaskSignatureComponent.
Methods
check() Validates the current modifications.
clear() Clears the current modifications.
getParameters() Returns a map of parameters and modifiers storing the values selected by the user.
setVariableSelection(VariableSelection selection) Setting the variable selection.
setSignature(SignatureApi sig) Setting the signature.
JComponent getComponent() Returns the component.

API details

Constructor


<code>ScaleTaskSignatureComponent()</code>
The construction of a new ScaleTaskSignatureComponent.
The construction of a new ScaleTaskSignatureComponent.

Methods

<code>check()</code>
Validates the current modifications.
Validates the current modifications for this RotateTaskSignatureComponent.

clear()
<p>Clears the current modifications.</p> <p>Clears the current modifications for this ScaleTaskSignatureComponentt.</p>
getParameters()
<p>Returns a map of parameters and modifiers storing the values selected by the user.</p> <p>Returns a map of parameters and modifiers storing the values selected by the user for this Scale-TaskSignatureComponent.</p>
setVariableSelection(VariableSelection selection)
<p>Setting the variable selection.</p> <p>Sets the given selection as the variable selection for this ScaleTaskSignatureComponentt.</p> <p>Argument</p> <p>VariableSelection selection [INPUT, MANDATORY]</p>
setSignature(SignatureApi sig)
<p>Setting the signature.</p> <p>Sets the given signature as the signature for this ScaleTaskSignatureComponentt.</p> <p>Argument</p> <p>SignatureApi sig [INPUT, MANDATORY]</p>
JComponent getComponent()
<p>Returns the component.</p> <p>Returns the component for this ScaleTaskSignatureComponent.</p> <p>Return</p> <p>JComponent</p> <p>Returns the component for this ScaleTaskSignatureComponent.</p>

2.351. SelectSpectrum

Full Name:	herschel.ia.toolbox.spectrum.SelectSpectrum
Type:	Java Task - 
Import:	from herschel.ia.toolbox.spectrum import SelectSpectrum

Description

Task for selecting individual spectra from a `SpectrumContainer` or fromw arrays of such and pack them in a new spectrum container.

Different selection schemes are available for specifying the selection:

- `segments`: The most simple scheme is to use 'segments' where you specify which point spectra and subsegments therein you want to select.
- `selection_lookup`: Specify a PyDictionary with column names as keys and a PyList of admissible values as values.
- `selection`: General selection model (type `SelectionModel`) that allows you to specify selections such as select all rows with values in a given column lying in a predefined interval.
- `selection_index`: Specify a PyList of indices that reference the point spectra included in the container.

The different options to specify selections can be combined. Here, an AND logic is adopted.

Example

Example 1: from jide:

```
from herschel.ia.toolbox.spectrum import SelectSpectrum
from herschel.ia.toolbox.spectrum.selections.models import RangesSelectionModel
select = SelectSpectrum()
selected = select(ds=spectra, segments=[1,3])
selected = select(ds=spectra, selection={"bbtype":[6031, 6032], -"buffer":[2]})
selected = select(ds=spectra, selection=[0,1,2,3])
selected = select(ds=spectra, selection={"LoFrequency":(4000.0,5000.0)})
selected = select(ds=spectra, selection={"Chopper":([-4.4,5.9],0.1)})
from herschel.ia.toolbox.spectrum.selections.models import RangesSelectionModel
selected = select(ds=spectra, selection=RangesSelectionModel("Chopper",
[-4.4,5.9], 0.1)) # the same as above
```

API Summary

Properties
Object ds [INPUT, OPTIONAL, default=no default value.]
Object segments [INPUT, OPTIONAL, default=no default value.]
Object selection [INPUT, OPTIONAL, default=None.]
PyDictionary Map> selection_lookup [INPUT, OPTIONAL, default=no default value.]
PyList selection_index [INPUT, OPTIONAL, default=No default value.]
<code>SpectrumContainer</code> result [OUTPUT, OPTIONAL, default=no default value]

API details

Properties

Object ds [INPUT, OPTIONAL, default=no default value.]

Input data to be processed by the task. Several types are possible:

- SpectrumContainer
- Array of SpectrumContainer, i.e. SpectrumContainer[]
- List of SpectrumContainer, i.e. List
- Any product with implementations of SpectrumContainer inside.

Object segments [INPUT, OPTIONAL, default=no default value.]

Specify what segments to restrict on. There are two options available:

- Specify a PyList of segment indices or
- pass an instance of a SegmentSelection which gives the information on what segments for each point spectrum included in the container.

Object selection [INPUT, OPTIONAL, default=None.]

Specification of what point spectra select. Different ways to specify these selections are possible:

- Specify a list of indices (in jython) of the point spectra to select.
- Use a dictionary (in jython) to specify a selection model (lookup for attributes, filter attributes to be included in ranges or intervals).
- Pass a string representation of a jython list or a jython dictionary. The string will be parsed and the processing delegated to the two cases above.
- Pass any java instance that implements the SelectionModel interface (for the advanced user).

PyDictionary|Map>; selection_lookup [INPUT, OPTIONAL, default=no default value.]

Specify a PyDictionary with keys given by the attribute name to look up and a list of admissible values. Behind the scenes, a DiscreteValueSelectionModel is instantiated.

PyList selection_index [INPUT, OPTIONAL, default=No default value.]

Specify a PyList with the indices of the point spectra to be considered.

SpectrumContainer result [OUTPUT, OPTIONAL, default=no default value]

Result object containing the results of the operation applied.


See also

- [ManyToOneSpectrumTask](#)

History

- 2007-06-01 - meli: initial.
- 2007-07-13 - meli: Javadoc updated.
- 2008-03-16 - meli: major refactoring of the toolbox.
- 2008-10-03 - meli: segments, javadoc

2.352. SerialArchive

Full Name:	herschel.ia.io.serial.SerialArchive
Alias:	SerialArchive
Type:	Java Class - 
Import:	from herschel.ia.io.serial import SerialArchive
Category:	class

Description

A class for writing and reading serialized objects.

The SerialArchive provides a transparent way to store and retrieve Products as defined within the Herschel Data Processing software.

Example

Example 1: default usage:

```
from herschel.ia.io.serial import SerialArchive
# write/read a Product in a serialized file.
s = SerialArchive()
s.save("output.ser", Product())
p = s.load("output.ser")
```

API Summary

Methods
Product load(InputStream stream) Loads a Product from an input stream.
Product load(String fileName) Loads a Product from a serialized file.
save(String fileName, [Optionally derived] Product product) Saves a Product to a serialized file.
createOutputStream(String fileName) Creates an object output stream for storing in a file.
createOutputStream(OutputStream file) Creates an object output stream for storing in a file.
createOutputStream(OutputStream stream) Creates an object output stream for storing in a file.
createInputStream(String fileName) Creates an object input stream for reading from a file.
createInputStream(String file) Creates an object input stream for reading from a file.
createInputStream(InputStream stream) Creates an object input stream for reading from a file.

API details

Methods

Product load(InputStream stream)
Loads a Product from an input stream.
Loads a Product from an input stream using a serial reader.
Argument
InputStream stream [INPUT, MANDATORY, default=no default value] Input stream from where the Product is to be read.
Return
Product
An object of the herschel.ia.dataset.Product family.

Product load(String fileName)
Loads a Product from a serialized file.
Loads a Product from a serialized file using a serial reader.
Argument
String fileName [INPUT, MANDATORY, default=no default value] Name of the serialized file.
Return
Product
An object of the herschel.ia.dataset.Product family.

save(String fileName, [Optionally derived] Product product)
Saves a Product to a serialized file.
Saves a Product to a serialized file with sufficient information to preserve the quantities of the data as well as the original dataset type and contents.
Arguments
String fileName [INPUT, MANDATORY, default=no default value] Name of the serialized file
[Optionally derived] Product product [INPUT, MANDATORY, default=no default value] An object of the herschel.ia.dataset.Product family.

createOutputStream(String fileName)
Creates an object output stream for storing in a file.
Creates an object output stream for storing objects in a file, by calling its writeObject method. The returned stream treats Products and Datasets as as special case, so that intermediate classes (resolvers) can manage them appropriately, even in the presence of different package names or class versions.

createOutputStream([String](#) fileName)

Argument

[String](#) **fileName** [INPUT, MANDATORY]

createOutputStream(OutputStream file)

Creates an object output stream for storing in a file.

Creates an object output stream for storing objects in a file, by calling its writeObject method. The returned stream treats Products and Datasets as as special case, so that intermediate classes (resolvers) can manage them appropriately, even in the presence of different package names or class versions.

Argument

OutputStream **file** [INPUT, MANDATORY, default=no default value]
File in which the data is to be serialized

createOutputStream(OutputStream stream)

Creates an object output stream for storing in a file.

Creates an object output stream for storing objects in a file, by calling its writeObject method. The returned stream treats Products and Datasets as as special case, so that intermediate classes (resolvers) can manage them appropriately, even in the presence of different package names or class versions.

Argument

OutputStream **stream** [INPUT, MANDATORY, default=no default value]
Stream to be wrapped

createInputStream([String](#) fileName)

Creates an object input stream for reading from a file.

Creates an object input stream for reading objects from a file, by calling its readObject method. The returned stream treats Products and Datasets as as special case, so that intermediate classes (resolvers) can manage them appropriately, even in the presence of different package names or class versions.

Argument

[String](#) **fileName** [INPUT, MANDATORY]

createInputStream([String](#) file)

Creates an object input stream for reading from a file.

Creates an object input stream for reading objects from a file, by calling its readObject method. The returned stream treats Products and Datasets as as special case, so that intermediate classes (resolvers) can manage them appropriately, even in the presence of different package names or class versions.

Argument

[String](#) **file** [INPUT, MANDATORY, default=no default value]
Name of the serialized file

createInputStream(InputStream stream)

Creates an object input stream for reading from a file.

createInputStream(InputStream stream)

Creates an object input stream for reading objects from a file, by calling its readObject method. The returned stream treats Products and Datasets as as special case, so that intermediate classes (resolvers) can manage them appropriately, even in the presence of different package names or class versions.

Argument


InputStream **stream** [INPUT, MANDATORY, default=no default value]

Stream to be wrapped

See also

- [FitsArchive](#)

2.353. SerialClientPool

Full Name:	herschel.ia.pal.pool.serial.SerialClientPool
Type:	Java Class - 
Import:	from herschel.ia.pal.pool.serial import SerialClientPool

Description

A ProductPool implementation used for accessing products stored on a remote pool across the network.

In order to use this, two things need to be set up first at the remote site:

1. A ProductPool implementation should be created.
2. A SerialServer is running, that talks to the remote ProductPool and relays data back/forth to the SerialClientPool via an accessible network port.

Make a note of the URL at which the server is running (eg "myserver.esa.int"), the ID of the remote product pool, and the port number at which data is relayed.

Examples

Example 1: Creating the remote pool and SerialServer. The SerialServer relays

```
data via port 4444 server=SerialServer(4444,
SimplePool.getInstance())
```

Example 2: Create a SerialClientPool that talks to the remote pool of ID

```
"simple.default" at site of URL -"remotehost.esa.int", port 4444
storage=ProductStorage()
storage.register(SerialClientPool("myserver.esa.int", 4444,
"simple.default" -))
```

API Summary

Constructor

[SerialClientPool\(String hostName, String portNumber, String id\)](#)

Creates an instance of a SerialClientPool connected to a remote

API details

Constructor

[SerialClientPool\(String hostName, String portNumber, String id\)](#)

Creates an instance of a SerialClientPool connected to a remote

pool.

Arguments

[String](#) **hostName** [INPUT, MANDATORY, default=The URL at which the]

SerialClientPool([String](#) hostName, [String](#) portNumber, [String](#) id)

remote server is running, eg "remotehost.esa.int"

[String](#) portNumber [INPUT, MANDATORY, default=The port number on which]

the remote server is running, eg 4444

[String](#) id [INPUT, MANDATORY, default=The identifier for the remote]


pool, eg "simple.mypool"

Return

[SerialClientPool](#)

An instance of the SerialClientPool.

2.354. SHIFT

Full Name:	herschel.ia.numeric.toolbox.basic.Shift
Alias:	SHIFT
Type:	Java Class - 
Import:	from herschel.ia.numeric.toolbox.basic import Shift

Description

Creates a new array out of the input array with a circular shift applied to all elements along specified dimension.

Examples

Example 1: Apply SHIFT on a Int1d

```
x=Int1d([0,1,2,3,4,5,6,7,8,9,10,11])
print SHIFT(x,-3)      # [3,4,5,6,7,8,9,10,11,0,1,2]
```

Example 2: Apply SHIFT on a Int2d

```
x=Int2d([[0,1,2,3],[4,5,6,7],[8,9,10,11]])
print SHIFT(x,2)      # [[4,5,6,7],[8,9,10,11],[0,1,2,3]]
print SHIFT(x,2,1)    # [[2,3,0,1],[6,7,4,5],[10,11,8,9]]
```

API Summary

Jython Syntax

```
<y>=SHIFT(<x>,<shift>,<dimension=0>)
```

Properties

any array of any rank **x** [INPUT, MANDATORY, default=no default value]

integer **shift** [INPUT, MANDATORY, default=no default value]

integer **dimension** [INPUT, MANDATORY, default=no default value]

boolean array or a boolean **y** [OUTPUT, MANDATORY, default=no default value]

API details

Properties

any array of any rank **x** [INPUT, MANDATORY, default=no default value]

The input array can be an array of rank 1

integer **shift** [INPUT, MANDATORY, default=no default value]

The number of element to shift

integer dimension [INPUT, MANDATORY, default=no default value]

The dimension to apply the shift to

boolean array or a boolean y [OUTPUT, MANDATORY, default=no default value]

Returns an array with the same number of elements as the input array and with the reverse order.

2.355. Short1d


Full Name:	herschel.ia.numeric.Short1d
Type:	Java Class - 
Import:	from herschel.ia.numeric import Short1d

Description

A rectangular numeric short array of rank 1.

Numeric arrays are part of the Numeric library. An explanation on how to use them, you can find in chapter **Numeric** -> **Arrays**. (Yes, it should be a link)

2.356. Short2d


Full Name:	herschel.ia.numeric.Short2d
Type:	Java Class - 
Import:	from herschel.ia.numeric import Short2d

Description

A rectangular numeric short array of rank 2.

Numeric arrays are part of the Numeric library. An explanation on how to use them, you can find in chapter **Numeric** -> **Arrays**. (Yes, it should be a link)

2.357. Short3d


Full Name:	herschel.ia.numeric.Short3d
Type:	Java Class - 
Import:	from herschel.ia.numeric import Short3d

Description

A rectangular numeric short array of rank 3.

Numeric arrays are part of the Numeric library. An explanation on how to use them, you can find in chapter **Numeric** -> **Arrays**. (Yes, it should be a link)

2.358. Short4d


Full Name:	herschel.ia.numeric.Short4d
Type:	Java Class - 
Import:	from herschel.ia.numeric import Short4d

Description

A rectangular numeric short array of rank 4.

Numeric arrays are part of the Numeric library. An explanation on how to use them, you can find in chapter **Numeric** -> **Arrays**. (Yes, it should be a link)

2.359. Short5d


Full Name:	herschel.ia.numeric.Short5d
Type:	Java Class - 
Import:	from herschel.ia.numeric import Short5d

Description

A rectangular numeric short array of rank 5.

Numeric arrays are part of the Numeric library. An explanation on how to use them, you can find in chapter **Numeric** -> **Arrays**. (Yes, it should be a link)

2.360. SIGCLIP

Full Name:	herschel.ia.numeric.toolbox.basic.Sigclip
Alias:	SIGCLIP
Type:	Java Class - 
Import:	from herschel.ia.numeric.toolbox.basic import Sigclip

Description

Sigclip finds values in an array that are more than $n \times (\text{standard deviation})$ larger than a comparator (the median or mean

value). The comparator can be calculated for a running box of user defined size (behavior = filter - default) or for the whole array (behavior = clip). Sigclip replaces these higher values with the comparator. In filter mode Sigclip does not include the investigated Pixel for the computation of the comparator and the sigma. The user has the choice of 8 parameters: 1. envSize: the size of the environment of surrounding pixels. If i is the coordinate of the tested pixel, the environment will be $[i - \text{envSize}, i + \text{envSize}]$ in the 1d case. For multi-dimensional arrays the box is extended in all dimensions. The default value for envSize is 3. 2. nsigma: the number of sigmas that a value has to exceed its next smaller value before it will be replaced. The default value is also 3. 3. The returnmode: Sigclip.RETURN_ARRAY (=0) returns a copy of the input array with the replaced values. Sigclip.RETURN_BOOL (=1) returns a boolean array where the clipped locations are marked as true (the other values are false). 4. the mode: Sigclip.MEAN or Sigclip.MEDIAN determines if median or mean of the neighbouring values is used to replace a sigclipped value 5. the treatment at the array edges. This is defined with the method setEdge(int edge). The choices are: Sigclip.TRUNCATE (default: scales down the box-size), Sigclip.FILL_EDGEVALUE (fills the empty indices of the box with the last value of the array), Sigclip.FILL_MEAN (fills the empty indices of the box with the mean of the rest of the box array) and Sigclip.FILL_MEDIAN (fills the empty indices of the box with the median of the rest of the box array) 6. the treatment of the outliers. outliers = "positive" removes only the values that are larger than the comparator (median or mean). This is the default. Other possible values are outliers = "negative" or outliers = "both". deprecation: the default value for outliers is deprecated. Currently it is "positive", but it will be changed to "both". The deprecation is best treated in your code by explicitly specifying the value for outliers! 7. The behavior as filter (default) or clip. behavior = "filter" uses the box of size env and lets Sigclip act as a filter. behavior = "clip" calculates the comparator mean or median and the sigma for the whole array and clips all array values according to these values 8. reduced1d reduced1d = true: in 1-dimensional arrays values are removed instead of being clipped. The length of the returned array becomes shorter reduced1d = false (default): values are clipped. The returned array has the same size as the input array

Example

Example 1: apply Sigclip to an Int1d array

```
array = Int1d.range(9)
array.set(5, 20)
print array
[0,1,2,3,4,20,6,7,8]
print array.apply( Sigclip() -)
[0,1,2,3,4,5,6,7,8]
print array.apply( Sigclip(return = Sigclip.RETURN_BOOL) -)
[false,false,false,false,false,true,false,false,false]
print array.apply( Sigclip( env=4, nsigma=2.5, mode=Sigclip.MEDIAN,
edge=Sigclip.TRUNCATE, return=Sigclip.RETURN_BOOL) -)
false,false,false,false,false,true,false,false,false]
array = Int1d(20, 9)
array.set(7, 20)
array.set(17, 0)
print array
```

Example 1: apply Sigclip to an Int1d array

```
[9,9,9,9,9,9,9,20,9,9,9,9,9,9,9,9,9,0,9,9]
print array.apply( Sigclip( reduceId = True, outliers = -"both" ) -)
[9,9,9,9,9,9,9,9,9,9,9,9,9,9,9,9,9]
```

API Summary

Properties
any 1-3d array of integral type x [INPUT, MANDATORY, default=no default value]
the box size that is analysed for every sample envSize [INPUT, NOT_MANDATORY, default=3]
the number of sigmas that a value has to exceed the next smaller value to be sigclipped nsigma [INPUT, NOT_MANDATORY, default=3]
determines returnmode [INPUT, if a numeric or a boolean array is returned, default=NOT_MANDATORY]
determines mode [INPUT, if median or mean is used to replace sig-clipped values, default=NOT_MANDATORY]
determines outliers [INPUT, if only positive, default=only negative or both outliers are removed]
determines behavior [INPUT, if Sigclip acts as a filter or a clas-sical clip, default=NOT_MANDATORY]
determines reduceId [INPUT, if found values are removed from the returned array, default=NOT_MANDATORY]

API details

Properties

any 1-3d array of integral type **x** [INPUT, MANDATORY, default=no default value]

the box size that is analysed for every sample **envSize** [INPUT, NOT_MANDATORY, default=3]

the number of sigmas that a value has to exceed the next smaller value to be sigclipped **nsigma** [INPUT, NOT_MANDATORY, default=3]

determines **returnmode** [INPUT, if a numeric or a boolean array is returned, default=NOT_MANDATORY]

Sigclip.RETURN_ARRAY (=0) returns a copy of the input array with the replaced values.
 Sigclip.RETURN_BOOL (=1) returns a boolean array where the clipped locations are marked as true (the other values are false).

determines **mode** [INPUT, if median or mean is used to replace sig-clipped values, default=NOT_MANDATORY]

Sigclip.MEAN (=0) uses the mean of the environment to replace sigclipped values.
 Sigclip.MEDIAN (=1) uses the median of the environment to replace sigclipped values.

determines outliers [INPUT, if only positive, default=only negative or both outliers are removed]

"positive" (default) only outliers larger than the comparator are found "negative" only outliers smaller than the comparator are found "both" all outliers are found deprecation: the default value for outliers is deprecated. Currently it is "positive", but it will be changed to "both". The deprecation is best treated in your code by explicitly specifying the value for outliers!


determines behavior [INPUT, if Sigclip acts as a filter or a classical clip, default=NOT_MANDATORY]

"filter" (default) Sigclip behaves like a filter (runs a box of size env over the input array). "clip" Sigclip works on the whole array at once.

determines reduce1d [INPUT, if found values are removed from the returned array, default=NOT_MANDATORY]

"False" (default) the size of the returned array is the same as the size of the input array "True" the sigclipped values are removed from 1d arrays.

2.361. SIGNUM

Full Name:	herschel.ia.numeric.toolbox.basic.Signum
Alias:	SIGNUM
Type:	Java Class - 
Import:	from herschel.ia.numeric.toolbox.basic import Signum

Description

Computes $x = \text{signum}(x)$, where x is a number or a numeric array.

Example

Example 1: Apply SIGNUM on a Int1d

```
x=Int1d( [-1,0,2] -)
print SIGNUM(x) # [-1,0,1]
```

API Summary

Jython Syntax

```
<y>=SIGNUM(<x>)
```

Properties

[any number or array \$x\$ \[INPUT, MANDATORY, default=no default value\]](#)

[same type as input \$y\$ \[OUTPUT, MANDATORY, default=no default value\]](#)

API details

Properties


any number or array x [INPUT, MANDATORY, default=no default value]

The input can be a number or an array

same type as input y [OUTPUT, MANDATORY, default=no default value]

Returns an array (or a number if the input is not an array) where each element is the signum of the corresponding element of the input array

2.362. simpleAsciiTableReader

Full Name:	herschel.ia.toolbox.util.SimpleAsciiTableReaderTask
Alias:	simpleAsciiTableReader
Type:	Java Task - 
Import:	from herschel.ia.toolbox.util import SimpleAsciiTableReaderTask
Category:	task

Description

The simpleAsciiTableReader task.

Task for reading ascii files, with minimal configuration. It provides a simple interface to AsciiTableReader but is less flexible. If it does not read your file as desired, you will have to use AsciiTableReader. A CSV file must have four lines of header, also comma-separated, with the same number of columns as the data. First line : column names; second line: types, third and fourth: just commas.

Examples

Example 1: simpleAsciiTableReader for a table with fields separated with spaces

```
filepath = -"path_to_file"
table=simpleAsciiTableReader(file=filepath)
```

Example 2: simpleAsciiTableReader for a CSV file with a header

```
filepath = -"path_to_file"
table=simpleAsciiTableReader(file=filepath, tableType="CSV")
```

API Summary

Jython Syntax

```
table=simpleAsciiTableReader(<file>[, <tableType>])
```

Properties

[String file](#) [INPUT, MANDATORY, default=null]

[String tableType](#) [INPUT, OPTIONAL, default="SPACES"]

[TableDataset table](#) [OUTPUT, MANDATORY, default=null]

API details

Properties

[String file](#) [INPUT, MANDATORY, default=null]

The path of the text file to be read.

[String tableType](#) [INPUT, OPTIONAL, default="SPACES"]

The type of table in the input file. Valid values are "CSV" and "SPACES"


<code>TableDataset table [OUTPUT, MANDATORY, default=null]</code>

The TableDataset read from the file.

History

- 2008-11-13 - First: prototype JDS
- 2008-11-18 - Second: prototype JDS
- 2008-11-24 - first: release JDS

2.363. SimpleCube

Full Name:	herschel.ia.dataset.image.SimpleCube
Type:	Java Class - 
Import:	from herschel.ia.dataset.image import SimpleCube
Category:	Image

Description

A Product describing cubes.

The Simplecube is a way to store and work with Cubes in the HCSS.

Example

Example 1: Creating a SimpleCube

```
s = SimpleCube()
s.setImage(myDouble3dImage)
```

API Summary

Constructors
SimpleCube() The standard constructor
SimpleCube(SimpleCube copy) The copy constructor
SimpleCube(String description) Constructor with a description
Methods
Wcs getWcs() Returns the World Coordinates System.
setImage(AbstractOrdered3dData image, Unit unit, String description) Sets the cube.
setImage(AbstractOrdered3dData image, Unit unit) Sets the cube.
setWcs(Wcs wcs) Sets the world coordinates system.
getIntensity(int depth, double row, double column) Returns the intensity
double getIntensityWorldCoordinates(int depth, double x, double y) Returns the intensity of the cube
setIntensity(int depth, int row, int column, Number value) Sets the intensity of the image.
Number getPixel(int depth, int row, int column)

Methods
Returns 1 pixel value.
int getDepth() Returns the number of layers
Cube getPreview(float res) Returns a preview of the cube
AbstractOrdered3dData getError() Returns the error of the cube as a Numeric3d.
AbstractOrdered3dData getExposure() Returns the exposure of the cube as a Numeric3d.
AbstractOrdered3dData getCoverage() Returns the coverage of the cube as an AbstractOrdered3dData.
Flag getFlag() Returns the flag of the cube.
int getHeight() Returns the height.
AbstractOrdered3dData getImage() returns the cube as a Numeric3d.
Unit getUnit() Returns the unit.
Unit getUnit(String identifier) Returns the unit.
int getWidth() Returns the width
boolean hasError() Checks whether the cube has an error.
boolean hasExposure() Checks whether the cube has an exposure.
boolean hasCoverage() Checks whether the cube has a coverage.
boolean hasFlag() Checks when the cube has a flag.
setUnit(Unit unit) Sets the unit.
setUnit(String unit) Sets the unit.
setUnit(String identifier, Unit unit) Sets the unit.
setUnit(String identifier, String unit) Sets the unit.
removeError() Removes the error.

Methods
removeExposure() Removes the exposure.
removeCoverage() Removes the coverage.
removeFlag() Removes the flag.
setImage(AbstractOrdered3dData cube) Sets the cube.
setError(AbstractOrdered3dData error) Sets the error.
setError(AbstractOrdered3dData error, String description) Sets the error.
setFlag(Flag flag, String description) Sets the flag.
setFlag(Flag flag) Sets the flag.
setExposure(AbstractOrdered3dData exposure, String description, Unit unit) Sets the exposure.
setExposure(AbstractOrdered3dData exposure, String description) Sets the exposure.
setExposure(AbstractOrdered3dData exposure) Sets the exposure.
setCoverage(AbstractOrdered3dData coverage, String description, Unit unit) Sets the coverage.
setCoverage(AbstractOrdered3dData coverage, String description) Sets the coverage.
setCoverage(AbstractOrdered3dData coverage) Sets the coverage.
int[] getDimensions() Returns the dimension.

API details

Constructors

<code>SimpleCube()</code>
The standard constructor
A constructor which creates a standard SimpleCube. The standard SimpleCube consists of a Numeric3d for the image. The SimpleCube has the depth as the first (most slowly varying) index.
Example
Typical example on how to create an SimpleCube.

SimpleCube()

```
image=SimpleCube(description="ngc 6992", image=im, flag=flag, errors=er,
unit=unit, wcs=wcs)
```

SimpleCube([SimpleCube](#) copy)

The copy constructor

Creates a new SimpleCube with the same contents of the given SimpleCube.

Argument

[SimpleCube](#) **copy** [INPUT, MANDATORY]

SimpleCube([String](#) description)

Constructor with a description

Creates a SimpleCube with a given description.

Argument

[String](#) **description** [INPUT, MANDATORY]

Methods

[Wcs](#) getWcs()

Returns the World Coordinates System.

Returns the World Coordinates System of the image.

Return

[Wcs](#)

The World Coordinates System of the image.

setImage([AbstractOrdered3dData](#) image, Unit unit, [String](#) description)

Sets the cube.

Sets the cube. You can give a description to the cube and give a unit. The given unit will automatically also be used for the errors on this cube. If the new cube has other dimensions than the original cube, the errors, the Exposure and the Flag are removed. If the new cube has the same dimensions, then the old values for the errors, exposure and flag are kept (but the unit of the errors is not).

Arguments

[AbstractOrdered3dData](#) **image** [INPUT, MANDATORY]

Unit **unit** [INPUT, MANDATORY]

[String](#) **description** [INPUT, MANDATORY]

setImage([AbstractOrdered3dData](#) image, Unit unit)

Sets the cube.

Sets the cube. You can give a unit. The given unit will automatically also be used for the errors on this cube. If the new cube has other dimensions than the original cube, the errors, exposure and the flag are removed. If the new cube has the same dimensions, then the old values for the errors, exposure and flag are kept (but the unit of the errors is not).

```
setImage(AbstractOrdered3dData image, Unit unit)
```

Arguments

```
AbstractOrdered3dData image [INPUT, MANDATORY]
Unit unit [INPUT, MANDATORY]
```

```
setWcs(Wcs wcs)
```

Sets the world coordinates system.

Sets the world coordinates system of the cube.

Argument

```
Wcs wcs [INPUT, MANDATORY]
```

```
getIntensity(int depth, double row, double column)
```

Returns the intensity

Returns the intensity of the SimpleCube at a given point. If the pixel is flagged out, NaN is given back.

Arguments

```
int depth [INPUT, MANDATORY]
double row [INPUT, MANDATORY]
double column [INPUT, MANDATORY]
```

```
double getIntensityWorldCoordinates(int depth, double x, double y)
```

Returns the intensity of the cube

Returns the intensity of the vube at a given point in world coordinates. If the pixel is flagged out, NaN is given back.

Arguments

```
int depth [INPUT, MANDATORY]
double x [INPUT, MANDATORY]
double y [INPUT, MANDATORY]
```

Return

double

The intensity

```
setIntensity(int depth, int row, int column, Number value)
```

Sets the intensity of the image.

Sets the intensity of the image at a given point.

Arguments

```
int depth [INPUT, MANDATORY]
int row [INPUT, MANDATORY]
int column [INPUT, MANDATORY]
Number value [INPUT, MANDATORY]
```


<p>Number <code>getPixel(int depth, int row, int column)</code></p> <p>Returns 1 pixel value.</p> <p>Returns 1 pixel value of the image.</p> <p>Arguments</p> <p> int depth [INPUT, MANDATORY]</p> <p> int row [INPUT, MANDATORY]</p> <p> int column [INPUT, MANDATORY]</p> <p>Return</p> <p> Number</p> <p> 1 pixel value of the image.</p>
<p><code>int getDepth()</code></p> <p>Returns the number of layers</p> <p>Returns the depth of this SimpleCube. This is the first axis (slowly varying axis).</p> <p>Return</p> <p> int</p> <p> Returns the depth of this SimpleCube.</p>
<p><code>Cube getPreview(float res)</code></p> <p>Returns a preview of the cube</p> <p>Returns a new SimpleCube, but with a lower resolution if res is smaller than 1. A higher resolution is returned, if res is larger than 1. The spatial regridding is based on linear interpolation which will, in general, not be suitable for scientific purposes.</p> <p>Argument</p> <p> float res [INPUT, MANDATORY]</p> <p>Return</p> <p> Cube</p> <p> The preview of the cube</p>
<p><code>AbstractOrdered3dData getError()</code></p> <p>Returns the error of the cube as a Numeric3d.</p> <p>Returns the error of the cube as a Numeric3d containing the error of every pixel.</p> <p>Return</p> <p> AbstractOrdered3dData</p> <p> The error</p>
<p><code>AbstractOrdered3dData getExposure()</code></p> <p>Returns the exposure of the cube as a Numeric3d.</p> <p>Returns the exposure of the cube as a Numeric3d containing the exposure of every pixel.</p>

AbstractOrdered3dData `getExposure()`**Return****AbstractOrdered3dData**

The exposure

AbstractOrdered3dData `getCoverage()`

Returns the coverage of the cube as an AbstractOrdered3dData.

Returns the coverage of the cube as an AbstractOrdered3dData containing the coverage of every pixel.

Return**AbstractOrdered3dData**

The coverage

Flag `getFlag()`

Returns the flag of the cube.

Returns the flag of the cube.

Return**Flag**

The flag

int `getHeight()`

Returns the height.

Returns the height of this SimpleCube.

Return**int**

Returns the height of this SimpleCube.

AbstractOrdered3dData `getImage()`

returns the cube as a Numeric3d.

Returns the cube as a Numeric3d containing the data of the cube.

Return**AbstractOrdered3dData**

The cube as a Numeric3d

Unit `getUnit()`

Returns the unit.

Returns the unit of the cube. The unit of the errors of this cube is the same as the unit of the cube.

Return

Unit <code>getUnit()</code>
Unit The unit
Unit <code>getUnit(String identifier)</code>
Returns the unit. Returns the unit of the cube, the coverage or the exposure. The unit of the errors of this cube is the same as the unit of the cube. Argument <code>String identifier</code> [INPUT, MANDATORY] Return Unit The unit
int <code>getWidth()</code>
Returns the width Returns the width of this SimpleImage. Return int Returns the width of this SimpleImage.
boolean <code>hasError()</code>
Checks whether the cube has an error. Returns true if the cube has an error. Return boolean True if the error is set.
boolean <code>hasExposure()</code>
Checks whether the cube has an exposure. Returns true if the cube has an exposure. Return boolean True if the exposure is set.
boolean <code>hasCoverage()</code>
Checks whether the cube has a coverage. Returns true if the cube has a coverage.

```
boolean hasCoverage()
```

Return**boolean**

True if the coverage is set.

```
boolean hasFlag()
```

Checks when the cube has a flag.

Returns true if the cube has a flag.

Return**boolean**

True if the cube has a flag.

```
setUnit(Unit unit)
```

Sets the unit.

Sets the unit of the cube. Adapting the unit of the cube will also adapt the unit of the errors on the cube.

Argument**Unit unit** [INPUT, MANDATORY]

```
setUnit(String unit)
```

Sets the unit.

Sets the unit of the image. Adapting the unit of the image will also adapt the unit of the errors on the image.

Argument**String unit** [INPUT, MANDATORY]

```
setUnit(String identifier, Unit unit)
```

Sets the unit.

Sets the unit of the cube, coverage or exposure. Adapting the unit of the cube will also adapt the unit of the errors on the cube.

Arguments**String identifier** [INPUT, MANDATORY]**Unit unit** [INPUT, MANDATORY]

```
setUnit(String identifier, String unit)
```

Sets the unit.

Sets the unit of the image, coverage or exposure. Adapting the unit of the image will also adapt the unit of the errors on the image.

Arguments**String identifier** [INPUT, MANDATORY]

setUnit([String](#) identifier, [String](#) unit)

[String](#) unit [INPUT, MANDATORY]

removeError()

Removes the error.

Removes the error, if an error exists.

removeExposure()

Removes the exposure.

Removes the exposure, if an exposure exists.

removeCoverage()

Removes the coverage.

Removes the coverage, if a coverage exists.

removeFlag()

Removes the flag.

Removes the flag, if a flag exists.

setImage([AbstractOrdered3dData](#) cube)

Sets the cube.

Sets the cube. If the new cube has other dimensions than the original cube, the errors, exposure and flag are removed. If the new cube has the same dimensions, then the old values for the errors, exposure and flag are kept.

Argument

[AbstractOrdered3dData](#) cube [INPUT, MANDATORY]

setError([AbstractOrdered3dData](#) error)

Sets the error.

Sets the errors of the cube. The [Numeric3d](#) containing the errors should have the same dimensions as the cube. If this is not the case, the errors will not be adapted.

Argument

[AbstractOrdered3dData](#) error [INPUT, MANDATORY]

setError([AbstractOrdered3dData](#) error, [String](#) description)

Sets the error.

Sets the errors of the cube. The [Numeric3d](#) containing the errors should have the same dimensions as the cube. If this is not the case, the errors will not be adapted. The errors are also described.

Arguments

[AbstractOrdered3dData](#) error [INPUT, MANDATORY]

[String](#) description [INPUT, MANDATORY]

setFlag(Flag flag, String description)

Sets the flag.

Sets the flag of every pixel of the cube. The Flag should have the same dimensions as the cube. If this is not the case, the Flag will not be adapted. The flag is also described.

Arguments

Flag flag [INPUT, MANDATORY]

String description [INPUT, MANDATORY]

setFlag(Flag flag)

Sets the flag.

Sets the flag of every pixel of the cube. The Flag should have the same dimensions as the cube. If this is not the case, the Flag will not be adapted.

Argument

Flag flag [INPUT, MANDATORY]

setExposure(AbstractOrdered3dData exposure, String description, Unit unit)

Sets the exposure.

Sets the exposure of every pixel of the cube. The Numeric3d containing the exposure should have the same dimensions as the cube. If this is not the case, the exposure will not be adapted.

Arguments

AbstractOrdered3dData exposure [INPUT, MANDATORY]

String description [INPUT, MANDATORY]

Unit unit [INPUT, MANDATORY]

setExposure(AbstractOrdered3dData exposure, String description)

Sets the exposure.

Sets the exposure of every pixel of the cube. The Numeric3d containing the exposure should have the same dimensions as the cube. If this is not the case, the exposure will not be adapted.

Arguments

AbstractOrdered3dData exposure [INPUT, MANDATORY]

String description [INPUT, MANDATORY]

setExposure(AbstractOrdered3dData exposure)

Sets the exposure.

Sets the exposure of every pixel of the cube. The Numeric3d containing the exposure should have the same dimensions as the cube. If this is not the case, the exposure will not be adapted.

Argument

AbstractOrdered3dData exposure [INPUT, MANDATORY]

setCoverage(AbstractOrdered3dData coverage, String description, Unit unit)

Sets the coverage.

setCoverage(AbstractOrdered3dData coverage, [String](#) description, Unit unit)

Sets the coverage of every pixel of the cube. The AbstractOrdered3dData containing the coverage should have the same dimensions as the cube. If this is not the case, the coverage will not be adapted.

Arguments

AbstractOrdered3dData **coverage** [INPUT, MANDATORY]

[String](#) **description** [INPUT, MANDATORY]

Unit **unit** [INPUT, MANDATORY]

setCoverage(AbstractOrdered3dData coverage, [String](#) description)

Sets the coverage.

Sets the coverage of every pixel of the cube. The AbstractOrdered3dData containing the coverage should have the same dimensions as the cube. If this is not the case, the coverage will not be adapted.

Arguments

AbstractOrdered3dData **coverage** [INPUT, MANDATORY]

[String](#) **description** [INPUT, MANDATORY]

setCoverage(AbstractOrdered3dData coverage)

Sets the coverage.

Sets the coverage of every pixel of the cube. The AbstractOrdered3dData containing the coverage should have the same dimensions as the cube. If this is not the case, the coverage will not be adapted.

Argument

AbstractOrdered3dData **coverage** [INPUT, MANDATORY]

int[] getDimensions()

Returns the dimension.


Returns the dimension (depth, width, height) of this SimpleCube.

Return

int[]

Returns the dimension (depth, width, height) of this SimpleCube.

2.364. simpleFitsReader

Full Name:	herschel.ia.toolbox.util.SimpleFitsReaderTask
Alias:	simpleFitsReader
Type:	Java Task - 
Import:	from herschel.ia.toolbox.util import SimpleFitsReaderTask
Category:	task

Description

The SimpleFitsReaderTask task.

Creates a product from a FITS file.

Examples

Example 1: SimpleFitsReaderTask with just a file parameter

```
filepath = -"path_to_file"
product=simpleFitsReader(file=filepath)
```

Example 2: SimpleFitsReaderTask with an optional reader type

```
filepath = -"path_to_file"
readertype = SimpleFitsReaderTask.ReaderType.STANDARD
product=simpleFitsReader(file=filepath, reader=readertype)
```

API Summary

Jython Syntax

```
product=simpleFitsReader(<file>[, <reader>])
```

Properties

[String](#) **file** [INPUT, MANDATORY, default=null]

[SimpleFitsReaderTask.ReaderType](#) **reader** [INPUT, OPTIONAL, default=[SimpleFitsReaderTask.ReaderType.HCSS_THEN_STANDARD](#)]

[Product](#) **product** [OUTPUT, MANDATORY, default=null]

API details

Properties

[String](#) **file** [INPUT, MANDATORY, default=null]

The path of the FITS file to be read.

[SimpleFitsReaderTask.ReaderType](#) **reader** [INPUT, OPTIONAL, default=[SimpleFitsReaderTask.ReaderType.HCSS_THEN_STANDARD](#)]

The strategy used to parse the contents. If an unrecognized value is typed, the default value is used. HCSS has priority over STANDARD as any FITS file can be read as STANDARD but only some of them are also HCSS FITS files. Possible values:

<code>SimpleFitsReaderTask.ReaderType reader [INPUT, OPTIONAL, default=SimpleFitsReaderTask.ReaderType.HCSS_THEN_STANDARD]</code>

- | |
|--|
| <ul style="list-style-type: none">• <code>SimpleFitsReaderTask.ReaderType.HCSS_THEN_STANDARD</code> (default): try to read the FITS Archive as an HCSS FITS file, if it fails then try to read it as a standard FITS file.• <code>SimpleFitsReaderTask.ReaderType.HCSS</code> : read it as an HCSS FITS file.• <code>SimpleFitsReaderTask.ReaderType.STANDARD</code> : read it as an STANDARD FITS file. |
|--|


<code>Product product [OUTPUT, MANDATORY, default=null]</code>
--

The Product read from the FITS file.

History

- 2008-07-09 - JCS: first release
- 2008-08-18 - JDS: added optional parameter reader (type) to allow transparent failover reading
- 2009-01-14 - JDS: cleanup start (SPR HCSS-5525)
- 2009-09-30 - JSS: modifiers are created through `getCustomModifiers` (SCR HCSS-8253)

2.365. simpleFitsWriter

Full Name:	herschel.ia.toolbox.util.SimpleFitsWriterTask
Alias:	simpleFitsWriter
Type:	Java Task - 
Import:	from herschel.ia.toolbox.util import SimpleFitsWriterTask
Category:	task

Description

Saves a product in a FITS file

SimpleFitsWriterTask flow:

- if no filename is given, flow is finished with RuntimeException "File is required."
- if the file already exists and warn is checked: user is asked to confirm overwriting
- if the user does not confirm overwriting (or mode is not interactive), flow is finished with RuntimeException "Execution cancelled."
- if the user asked for compression, the output will be compressed
- Otherwise the product is saved in HCSS' FITS format in the chosen file

Examples

Example 1: Saving a product into a file

```
p = Product()
f = -"path_to_file"
simpleFitsWriter(product=p,file=f)
```

Example 2: Saving a product into a file asking before overwriting

```
p = Product()
f = -"path_to_file"
simpleFitsWriter(product=p,file=f,warn=True)
```

Example 3: Saving a product into a gzipped file

```
p = Product()
f = -"path_to_file"
simpleFitsWriter(product=p,file=f,compression="GZIP")
```

API Summary

Jython Syntax

```
simpleFitsWriter(<product>, <file>[, <warn>, <compression>])
```

Properties

[Product](#) **product** [INPUT, MANDATORY, default=null]

[String](#) **file** [INPUT, MANDATORY, default=null]

Properties
Boolean <code>warn</code> [INPUT, OPTIONAL, default=false]
String <code>compression</code> [INPUT, OPTIONAL, default="NONE"]

Limitations

SimpleFitsReader does not support zipped fits.

API details

Properties

Product <code>product</code> [INPUT, MANDATORY, default=null]
Product to be saved.
String <code>file</code> [INPUT, MANDATORY, default=null]
FITS filename to save the product into.
Boolean <code>warn</code> [INPUT, OPTIONAL, default=false]
If true, asks confirmation before overwriting.
String <code>compression</code> [INPUT, OPTIONAL, default="NONE"]
The type of compression in the output file. Valid values are "NONE", "ZIP" and "GZIP"

History

- 2008-07-08 - JCS: first release
- 2008-08-19 - JDS: Using PopUpDialog, SCR 4573: asking before overwriting
- 2008-11-24 - JDS: SPR 5525: cleanup start
- 2009-01-07 - JDS: SCR-8864: support compression

2.366. SimpleImageExplorer

Full Name:	herschel.ia.gui.image.SimpleImageExplorer
Type:	Java Class - 
Import:	from herschel.ia.gui.image import SimpleImageExplorer

Description

An explorer for Images.

An explorer for Images.

API Summary

Constructors
SimpleImageExplorer() The construction of a new SimpleImageExplorer.
SimpleImageExplorer(Object object) The construction of a new SimpleImageExplorer associated with the given object.
Methods
String getDescription() Returns the description.
boolean canHandle(Class className) Checks whether this SimpleImageExplorer can handle objects of the given class.
setObject(Object object) Sets the object.
Object getObject() Returns the object.
Class getVariableType() Returns the expected variable type.
JComponent getComponent() Returns the component that is responsible for displaying the data object.
boolean makeEditorContent() Creates the editor content for this Explorer.
addDataObjectListener(DataObjectListener arg0) Adds the given listener.
removeDataObjectListener(DataObjectListener arg0) Removes the given listener.
setVariableSelection(VariableSelection selection) Setting the variable selection.
VariableSelection getVariableSelection() Returns the variable selection for this SimpleImageExplorer,
overplotImageContours(ImageContour contours) Overplotting contours.

API details

Constructors

SimpleImageExplorer()
The construction of a new SimpleImageExplorer. The construction of a new SimpleExplorer.
SimpleImageExplorer(Object object)
The construction of a new SimpleImageExplorer associated with the given object. The construction of a new SimpleExplorer associated with the given object.
Argument
<code>Object object</code> [INPUT, MANDATORY]

Methods

String getDescription()
Returns the description. Returns the description for this SimpleImageExplorer.
Return
<code>String</code>
Returns the description for this SimpleImageExplorer.
boolean canHandle(Class className)
Checks whether this SimpleImageExplorer can handle objects of the given class. Returns true if this SimpleImageExplorer can handle objects of the given class; false otherwise.
Argument
<code>Class className</code> [INPUT, MANDATORY]
Return
<code>boolean</code>
Returns true of this SimpleImageExplorer can handle objects of the given class; false otherwise.
setObject(Object object)
Sets the object. Sets the object for this SimpleImageExplorer to the given object.
Argument
<code>Object object</code> [INPUT, MANDATORY]
Object getObject()
Returns the object.

Object getObject()
Returns the object for this SimpleExplorer.
Return
Object
Returns the object for this SimpleExplorer.
Class getVariableType()
Returns the expected variable type.
Returns the expected variable type for this SimpleImageExplorer.
Return
Class
Returns the expected variable type for this SimpleImageExplorer.
JComponent getComponent()
Returns the component that is responsible for displaying the data object.
Returns the component that is responsible for displaying the data object for this SimpleImageExplorer.
Return
JComponent
Returns the component that is responsible for displaying the data object for this SimpleImageExplorer.
boolean makeEditorContent()
Creates the editor content for this Explorer.
Returns true if the editor content for this SimpleImageExplorer could be made; false otherwise.
Return
boolean
Returns true if the editor content for this SimpleImageExplorer could be made; false otherwise.
addDataObjectListener(DataObjectListener arg0)
Adds the given listener.
Adds the given listener to this SimpleImageExplorer to receive data object events from it.
Argument
DataObjectListener arg0 [INPUT, MANDATORY]
removeDataObjectListener(DataObjectListener arg0)
Removes the given listener.
Removes the given listener for this SimpleImageExplorer, so that it no longer receives data object events by this explorer.

```
removeDataObjectListener(DataObjectListener arg0)
```

Argument

DataObjectListener **arg0** [INPUT, MANDATORY]

```
setVariableSelection(VariableSelection selection)
```

Setting the variable selection.

Sets the given variable selection as the variable selection for this SimpleImageExplorer, in order to allow drag and drop.

Argument

VariableSelection **selection** [INPUT, MANDATORY]

```
VariableSelection getVariableSelection()
```

Returns the variable selection for this SimpleImageExplorer,

in order to allow drag and drop.

Return

VariableSelection

Returns the variable selection. Returns the variable selection for this SimpleImageExplorer, in order to allow drag and drop.

```
overplotImageContours(ImageContour contours)
```


Overplotting contours.

Allows to overplot contours on the image displayed by this SimpleImageExplorer.

Argument

[ImageContour](#) **contours** [INPUT, MANDATORY]

2.367. SimpleImage

Full Name:	herschel.ia.dataset.image.SimpleImage
Type:	Java Class - 
Import:	from herschel.ia.dataset.image import SimpleImage
Category:	Image

Description

A Product describing images.

SimpleImage is a Product describing images.

Example

Example 1: Creating a SimpleImage
<pre>s = SimpleImage() s.setImage(myDouble2dImage)</pre>

API Summary

Constructors
SimpleImage() The standard constructor
SimpleImage(SimpleImage copy) Copy constructor
SimpleImage(String description) Constructor with a description
Methods
setImage(AbstractOrdered2dData image, Unit unit, String description) Sets the image.
setWcs(Wcs wcs) Sets the world coordinates system.
Wcs getWcs() Returns the World Coordinates System.
setImage(AbstractOrdered2dData image, Unit unit) Sets the image.
getIntensity(double row, double column) Returns the intensity
double getIntensityWorldCoordinates(double x, double y) Returns the intensity of the image
setIntensity(int row, int column, Number value) Sets the intensity of the image.
Number getPixel(int row, int column) Returns 1 pixel value.

Methods	
Image <code>getPreview(float res)</code>	Returns a preview of the image
AbstractOrdered2dData <code>getError()</code>	Returns the error of the image as a Numeric2d.
AbstractOrdered2dData <code>getExposure()</code>	Returns the exposure of the image as a Numeric2d.
AbstractOrdered2dData <code>getCoverage()</code>	Returns the coverage of the image as an AbstractOrdered2dData.
Flag <code>getFlag()</code>	Returns the flag of the image.
int <code>getHeight()</code>	Returns the height.
AbstractOrdered2dData <code>getImage()</code>	returns the image as a Numeric2d.
Unit <code>getUnit()</code>	Returns the unit.
Unit <code>getUnit(String identifier)</code>	Returns the unit.
int <code>getWidth()</code>	Returns the width
double <code>getWavelength()</code>	Returns the reference wavelength
double <code>getWavelength(Length unit)</code>	Returns the reference wavelength
boolean <code>hasError()</code>	Checks whether the image has an error.
boolean <code>hasExposure()</code>	Checks whether the image has an exposure.
boolean <code>hasCoverage()</code>	Checks whether the image has a coverage.
boolean <code>hasFlag()</code>	Checks when the image has a flag.
setUnit(Unit unit)	Sets the unit.
setUnit(String unit)	Sets the unit.
setUnit(String identifier, Unit unit)	Sets the unit.
setUnit(String identifier, String unit)	Sets the unit.
setWavelength(double wavelength)	

Methods
Sets the reference wavelength.
setWavelength(double wavelength, Length unit)
Sets the reference wavelength.
removeError()
Removes the error.
removeExposure()
Removes the exposure.
removeCoverage()
Removes the coverage.
removeFlag()
Removes the flag.
setImage(AbstractOrdered2dData image)
Sets the image.
setError(AbstractOrdered2dData error)
Sets the error.
setError(AbstractOrdered2dData error, String description)
Sets the error.
setFlag(Flag flag, String description)
Sets the flag.
setFlag(Flag flag)
Sets the flag.
setExposure(AbstractOrdered2dData exposure, String description, Unit unit)
Sets the exposure.
setExposure(AbstractOrdered2dData exposure, String description)
Sets the exposure.
setExposure(AbstractOrdered2dData exposure)
Sets the exposure.
setCoverage(AbstractOrdered2dData coverage, String description, Unit unit)
Sets the coverage.
setCoverage(AbstractOrdered2dData coverage, String description)
Sets the coverage.
setCoverage(AbstractOrdered2dData coverage)
Sets the coverage.
double getFrequency()
Returns the reference frequency.
double getFrequency(Frequency freq)
Returns the reference frequency.
setFrequency(double frequency)
Sets the reference frequency.
setFrequency(double frequency, Frequency unit)

Methods
Sets the reference frequency.
<code>int[] getDimensions()</code>
Returns the dimension.
<code>Double2d getImageData()</code>
Returns the Image data.

API details

Constructors

<code>SimpleImage()</code>
The standard constructor
A constructor which creates a standard SimpleImage. The standard SimpleImage consists of a Numeric2d for the image, no error and no integration time. The dimension of the standard SimpleImage is 0x0. The reference wavelength is set to 0.0 microns.
Example
Typical example on how to create an SimpleImage.
<pre>image=SimpleImage(description="ngc 6992", image=im, flag=flag, errors=er, unit=unit, wavelength=wavelength, wcs=wcs)</pre>

<code>SimpleImage(SimpleImage copy)</code>
Copy constructor
Constructor which makes a copy from an existent SimpleImage.
Argument
<code>SimpleImage copy</code> [INPUT, MANDATORY]

<code>SimpleImage(String description)</code>
Constructor with a description
Creates a SimpleImage with a given description.
Argument
<code>String description</code> [INPUT, MANDATORY]

Methods

<code>setImage(AbstractOrdered2dData image, Unit unit, String description)</code>
Sets the image.
Sets the image. You can give a description to the image and give a unit. The given unit will automatically also be used for the errors on this image. If the new image has other dimensions than the original image, the errors, exposure and the flag are removed. If the new image has the same dimensions, then the old values for the errors, exposure and flag are kept (but the unit of the errors is not).
Arguments

setImage(AbstractOrdered2dData **image**, Unit **unit**, [String](#) **description**)

AbstractOrdered2dData **image** [INPUT, MANDATORY]
 Unit **unit** [INPUT, MANDATORY]
[String](#) **description** [INPUT, MANDATORY]

setWcs([Wcs](#) **wcs**)

Sets the world coordinates system.

Sets the world coordinates system of the image.

Argument

[Wcs](#) **wcs** [INPUT, MANDATORY]

[Wcs](#) **getWcs()**

Returns the World Coordinates System.

Returns the World Coordinates System of the image.

Return

[Wcs](#)

The World Coordinates System of the image.

setImage(AbstractOrdered2dData **image**, Unit **unit**)

Sets the image.

Sets the image. You can give a unit. The given unit will automatically also be used for the errors on this image. If the new image has other dimensions than the original image, the errors, the exposure and the flag are removed. If the new image has the same dimensions, then the old values for the errors, exposure and the flag are kept (but the unit of the errors is not).

Arguments

AbstractOrdered2dData **image** [INPUT, MANDATORY]
 Unit **unit** [INPUT, MANDATORY]

getIntensity(double row, double column)

Returns the intensity

Returns the intensity of the SimpleImage at a given point. If the pixel is flagged out, NaN is given back.

Arguments

double **row** [INPUT, MANDATORY]
 double **column** [INPUT, MANDATORY]

double getIntensityWorldCoordinates(double x, double y)

Returns the intensity of the image

Returns the intensity of the image at a given point in world coordinates. If the pixel is flagged out, NaN is given back.

Arguments

double **x** [INPUT, MANDATORY]

double <code>getIntensityWorldCoordinates(double x, double y)</code>
double <code>y</code> [INPUT, MANDATORY]
Return
double
The intensity
setIntensity(int row, int column, Number value)
Sets the intensity of the image.
Sets the intensity of the image at a given point.
Arguments
int <code>row</code> [INPUT, MANDATORY]
int <code>column</code> [INPUT, MANDATORY]
Number <code>value</code> [INPUT, MANDATORY]
Number getPixel(int row, int column)
Returns 1 pixel value.
Returns 1 pixel value of the image.
Arguments
int <code>row</code> [INPUT, MANDATORY]
int <code>column</code> [INPUT, MANDATORY]
Return
Number
1 pixel value of the image.
Image getPreview(float res)
Returns a preview of the image
Returns a new SimpleImage, but with a lower resolution if res is smaller than 1. A higher resolution is returned, if res is larger than 1. The spatial regridding is based on linear interpolation which will, in general, not be suitable for scientific purposes.
Argument
float <code>res</code> [INPUT, MANDATORY]
Return
Image
The preview of the image
AbstractOrdered2dData getError()
Returns the error of the image as a Numeric2d.
Returns the error of the image as a Numeric2d containing the error of every pixel.
Return
AbstractOrdered2dData

AbstractOrdered2dData `getError()`

The error

AbstractOrdered2dData `getExposure()`

Returns the exposure of the image as a Numeric2d.

Returns the exposure of the image as a Numeric2d containing the exposure of every pixel.

Return

AbstractOrdered2dData

The exposure

AbstractOrdered2dData `getCoverage()`

Returns the coverage of the image as an AbstractOrdered2dData.

Returns the coverage of the image as an AbstractOrdered2dData containing the coverage of every pixel.

Return

AbstractOrdered2dData

The coverage

Flag `getFlag()`

Returns the flag of the image.

Returns the flag of the image.

Return

[Flag](#)

The flag

int `getHeight()`

Returns the height.

Returns the height of this SimpleImage.

Return

int

The height of this SimpleImage.

AbstractOrdered2dData `getImage()`

returns the image as a Numeric2d.

Returns the image as a Numeric2d containing the data of the image.

Return

AbstractOrdered2dData

The image as a Numeric2d

Unit `getUnit()`

Returns the unit.

Returns the unit of the image. The unit of the errors of this image is the same as the unit of the image.

Return

Unit

The unit

Unit `getUnit(String identifier)`

Returns the unit.

Returns the unit of the image, the coverage or the exposure. The unit of the errors of this image is the same as the unit of the image.

Argument

`String identifier` [INPUT, MANDATORY]

Return

Unit

The unit

int `getWidth()`

Returns the width

Returns the width of this SimpleImage.

Return

int

The width of this SimpleImage.

double `getWavelength()`

Returns the reference wavelength

Returns the reference wavelength of the image.

Return

double

The reference wavelength

double `getWavelength(Length unit)`

Returns the reference wavelength

Returns the reference wavelength of the image.

Argument

`Length unit` [INPUT, MANDATORY]

Return

double getWavelength(Length unit)**double**

The reference wavelength

boolean hasError()

Checks whether the image has an error.

Returns true if the image has an error.

Return**boolean**

True if the error is set.

boolean hasExposure()

Checks whether the image has an exposure.

Returns true if the image has an exposure.

Return**boolean**

True if the exposure is set.

boolean hasCoverage()

Checks whether the image has a coverage.

Returns true if the image has a coverage.

Return**boolean**

True if the coverage is set.

boolean hasFlag()

Checks when the image has a flag.

Returns true if the image has a flag.

Return**boolean**

True if the image has a flag.

setUnit(Unit unit)

Sets the unit.

Sets the unit of the image. Adapting the unit of the image will also adapt the unit of the errors on the image.

ArgumentUnit **unit** [INPUT, MANDATORY]

setUnit(String unit)
Sets the unit.
Sets the unit of the image. Adapting the unit of the image will also adapt the unit of the errors on the image.
Argument
String unit [INPUT, MANDATORY]
setUnit(String identifier, Unit unit)
Sets the unit.
Sets the unit of the image, coverage or exposure. Adapting the unit of the image will also adapt the unit of the errors on the image.
Arguments
String identifier [INPUT, MANDATORY]
Unit unit [INPUT, MANDATORY]
setUnit(String identifier, String unit)
Sets the unit.
Sets the unit of the image, coverage or exposure. Adapting the unit of the image will also adapt the unit of the errors on the image.
Arguments
String identifier [INPUT, MANDATORY]
String unit [INPUT, MANDATORY]
setWavelength(double wavelength)
Sets the reference wavelength.
Set the reference wavelength of the image in microns.
Argument
double wavelength [INPUT, MANDATORY]
setWavelength(double wavelength, Length unit)
Sets the reference wavelength.
Set the reference wavelength of the image.
Arguments
double wavelength [INPUT, MANDATORY]
Length unit [INPUT, MANDATORY]
removeError()
Removes the error.
Removes the error, if an error exists.
removeExposure()
Removes the exposure.

removeExposure()

Removes the exposure, if an exposure exists.

removeCoverage()

Removes the coverage.

Removes the coverage, if a coverage exists.

removeFlag()

Removes the flag.

Removes the flag, if a flag exists.

setImage(AbstractOrdered2dData image)

Sets the image.

Sets the image. If the new image has other dimensions than the original image, the errors, exposure and flag are removed. If the new image has the same dimensions, then the old values for the errors, exposure and flag are kept.

Argument

AbstractOrdered2dData **image** [INPUT, MANDATORY]

setError(AbstractOrdered2dData error)

Sets the error.

Sets the errors of the image. The Numeric2d containing the errors should have the same dimensions as the image. If this is not the case, the errors will not be adapted.

Argument

AbstractOrdered2dData **error** [INPUT, MANDATORY]

setError(AbstractOrdered2dData error, String description)

Sets the error.

Sets the errors of the image. The Numeric2d containing the errors should have the same dimensions as the image. If this is not the case, the errors will not be adapted. The errors are also described.

Arguments

AbstractOrdered2dData **error** [INPUT, MANDATORY]

[String](#) **description** [INPUT, MANDATORY]

setFlag(Flag flag, String description)

Sets the flag.

Sets the flag of every pixel of the image. The Flag should have the same dimensions as the image. If this is not the case, the Flag will not be adapted. The flag is also described.

Arguments

[Flag](#) **flag** [INPUT, MANDATORY]

setFlag(Flag flag, String description)

String description [INPUT, MANDATORY]

setFlag(Flag flag)

Sets the flag.

Sets the flag of every pixel of the image. The Flag should have the same dimensions as the image. If this is not the case, the Flag will not be adapted.

Argument

Flag flag [INPUT, MANDATORY]

setExposure(AbstractOrdered2dData exposure, String description, Unit unit)

Sets the exposure.

Sets the exposure of every pixel of the image. The Numeric2d containing the exposure should have the same dimensions as the image. If this is not the case, the exposure will not be adapted.

Arguments

AbstractOrdered2dData exposure [INPUT, MANDATORY]

String description [INPUT, MANDATORY]

Unit unit [INPUT, MANDATORY]

setExposure(AbstractOrdered2dData exposure, String description)

Sets the exposure.

Sets the exposure of every pixel of the image. The Numeric2d containing the exposure should have the same dimensions as the image. If this is not the case, the exposure will not be adapted.

Arguments

AbstractOrdered2dData exposure [INPUT, MANDATORY]

String description [INPUT, MANDATORY]

setExposure(AbstractOrdered2dData exposure)

Sets the exposure.

Sets the exposure of every pixel of the image. The Numeric2d containing the exposure should have the same dimensions as the image. If this is not the case, the exposure will not be adapted.

Argument

AbstractOrdered2dData exposure [INPUT, MANDATORY]

setCoverage(AbstractOrdered2dData coverage, String description, Unit unit)

Sets the coverage.

Sets the coverage of every pixel of the image. The AbstractOrdered2dData containing the coverage should have the same dimensions as the image. If this is not the case, the coverage will not be adapted.

Arguments

AbstractOrdered2dData coverage [INPUT, MANDATORY]

setCoverage(AbstractOrdered2dData coverage, [String](#) description, Unit unit)

[String](#) **description** [INPUT, MANDATORY]

Unit **unit** [INPUT, MANDATORY]

setCoverage(AbstractOrdered2dData coverage, [String](#) description)

Sets the coverage.

Sets the coverage of every pixel of the image. The AbstractOrdered2dData containing the coverage should have the same dimensions as the image. If this is not the case, the coverage will not be adapted.

Arguments

AbstractOrdered2dData **coverage** [INPUT, MANDATORY]

[String](#) **description** [INPUT, MANDATORY]

setCoverage(AbstractOrdered2dData coverage)

Sets the coverage.

Sets the coverage of every pixel of the image. The AbstractOrdered2dData containing the coverage should have the same dimensions as the image. If this is not the case, the coverage will not be adapted.

Argument

AbstractOrdered2dData **coverage** [INPUT, MANDATORY]

double getFrequency()

Returns the reference frequency.

Returns the reference frequency of the image.

Return

double

The reference frequency of the image.

double getFrequency(Frequency freq)

Returns the reference frequency.

Returns the reference frequency of the image.

Argument

Frequency **freq** [INPUT, MANDATORY]

Return

double

The reference frequency of the image.

setFrequency(double frequency)

Sets the reference frequency.

Set the reference frequency of the image in gigahertz.

setFrequency(double frequency)**Argument**

double **frequency** [INPUT, MANDATORY]

setFrequency(double frequency, Frequency unit)

Sets the reference frequency.

Set the reference frequency of the image.

Arguments

double **frequency** [INPUT, MANDATORY]

Frequency **unit** [INPUT, MANDATORY]

int[] getDimensions()

Returns the dimension.

Returns the dimension (width, height) of this SimpleImage.

Return

int[]

The dimension (width, height) of this SimpleImage.

[Double2d](#) getImageData()

Returns the Image data.


Returns the image data as a Double2d.

Return

[Double2d](#)

The image data as Double2d

2.368. SimpleStack

Full Name:	herschel.ia.slice.image.SimpleStack
Type:	Java Class - 
Import:	from herschel.ia.slice.image import SimpleStack
Category:	Image

Description

A SimpleStack is a stack of Images.

The SimpleStack is a way to store and work with a stack of images in the HCSS

Example

Example 1: Typical example on how to create a SimpleStack.

```
stack = SimpleStack(description="Some images")
stack.setImage(new SimpleImage())
```

API Summary

Constructors	
SimpleStack()	The standard constructor
SimpleStack(SimpleStack copy)	The copy constructor
SimpleStack(String description)	The constructor with a description
Methods	
copy()	Returns a copy.
addImage(Image image)	Adds a new Image to the stack
replaceImage(int layer, Image image)	Replaces an image of the stack.
int getDepth()	Returns the number of layers of this stack
SimpleImage getImage(int layer)	Returns the chosen image from the Stack.
boolean hasExposure(int layer)	Checks whether the given layer has an exposure.
boolean hasError(int layer)	Checks whether the given layer has an error.
boolean hasFlag(int layer)	Checks whether the given layer has a flag.
AbstractOrdered2dData getExposure(int layer)	

Methods
Returns the exposure of the given layer as a Numeric2d.
AbstractOrdered2dData <code>getError(int layer)</code>
Returns the error of the given layer as a Numeric2d.
Flag <code>getFlag(int layer)</code>
Returns the flag of the given layer.
<code>setExposure(int layer, AbstractOrdered2dData exposure)</code>
Sets the exposure of a certain layer.
<code>setError(int layer, AbstractOrdered2dData errors)</code>
Sets the error of a certain layer.
<code>setFlag(int layer, Flag flag)</code>
Sets the flag of a certain layer.
<code>removeError(int layer)</code>
Removes the error of a certain layer.
<code>removeExposure(int layer)</code>
Removes the exposure of a certain layer.
<code>removeFlag(int layer)</code>
Removes the flag of a certain layer.
double <code>getIntensity(int layer, double row, double column)</code>
Returns the intensity.
<code>setIntensity(int layer, int row, int column, Number value)</code>
Sets the intensity of the stack.
double <code>getIntensityWorldCoordinates(int layer, double x, double y)</code>
Returns the intensity of the stack
int[] <code>getDimensions(int layer)</code>
Return the dimensions of the layer.
int <code>getHeight(int layer)</code>
Returns the height
int <code>getWidth(int layer)</code>
Returns the width.
Unit <code>getUnit(int layer)</code>
Returns the unit.
<code>setUnit(int layer, Unit unit)</code>
Sets the unit.
<code>setWavelength(int layer, double wavelength)</code>
Sets the reference wavelength.
<code>setWavelength(int layer, double wavelength, Length unit)</code>
Sets the reference wavelength.
double <code>getWavelength(int layer)</code>
Returns the reference wavelength
double <code>getWavelength(int layer, Length unit)</code>
Returns the reference wavelength

Methods
<p>Wcs getWcs(int layer)</p> <p>Returns the world coordinates system.</p>
<p>setWcs(int layer, Wcs wcs)</p> <p>Sets the world coordinates system.</p>
<p>Stack getPreview(float resolution)</p> <p>Returns a preview of the stack</p>
<p>removeImage(int layer)</p> <p>Removes a layer.</p>

API details

Constructors

<code>SimpleStack()</code>
<p>The standard constructor</p> <p>A constructor which creates a standard SimpleStack.</p> <p>Example</p> <p>Typical example on how to create an SimpleStack.</p> <pre>image=SimpleStack(description="Some images")</pre>

<code>SimpleStack(SimpleStack copy)</code>
<p>The copy constructor</p> <p>Makes a new SimpleStack from a copy of the given Stack.</p> <p>Argument</p> <p>SimpleStack copy [INPUT, MANDATORY]</p>

<code>SimpleStack(String description)</code>
<p>The constructor with a description</p> <p>This constructor also sets the description of the SimpleStack</p> <p>Argument</p> <p>String description [INPUT, MANDATORY]</p>

Methods

<p><code>copy()</code></p> <p>Returns a copy.</p> <p>Returns a copy from this SimpleStack.</p>
<p><code>addImage(Image image)</code></p> <p>Adds a new Image to the stack</p> <p>Adds a new image to the stack. The images is added as the last image.</p>

addImage(Image image)
Argument Image image [INPUT, MANDATORY]
replaceImage(int layer, Image image)
Replaces an image of the stack. Replaces an image of the stack. Arguments int layer [INPUT, MANDATORY] Image image [INPUT, MANDATORY]
int getDepth()
Returns the number of layers of this stack Returns the number of layers of this stack. Return int Returns the number of layers of this stack.
SimpleImage getImage(int layer)
Returns the chosen image from the Stack. Returns the chosen layer of the Stack as a SimpleImage. Argument int layer [INPUT, MANDATORY] Return SimpleImage The chosen layer of the Stack.
boolean hasExposure(int layer)
Checks whether the given layer has an exposure. Returns true if the layer has an exposure. Argument int layer [INPUT, MANDATORY] Return boolean True if the exposure of the given layer is set.
boolean hasError(int layer)
Checks whether the given layer has an error. Returns true if the layer has an error.

boolean hasError(int layer)
Argument int layer [INPUT, MANDATORY]
Return boolean True if the error of the given layer is set.
boolean hasFlag(int layer)
Checks whether the given layer has a flag. Returns true if the layer has a flag.
Argument int layer [INPUT, MANDATORY]
Return boolean True if the flag of the given layer is set.
AbstractOrdered2dData getExposure(int layer)
Returns the exposure of the given layer as a Numeric2d. Returns the exposure of the given layer as a Numeric2d containing the exposure of every pixel.
Argument int layer [INPUT, MANDATORY]
Return AbstractOrdered2dData The exposure
AbstractOrdered2dData getError(int layer)
Returns the error of the given layer as a Numeric2d. Returns the error of the given layer as a Numeric2d containing the error of every pixel.
Argument int layer [INPUT, MANDATORY]
Return AbstractOrdered2dData The error
Flag getFlag(int layer)
Returns the flag of the given layer. Returns the flag of the given layer.
Argument int layer [INPUT, MANDATORY]

Flag <code>getFlag(int layer)</code>
Return Flag The flag
setExposure(int layer, AbstractOrdered2dData exposure)
Sets the exposure of a certain layer. Set the exposure of the given layer. Arguments int layer [INPUT, MANDATORY] AbstractOrdered2dData exposure [INPUT, MANDATORY]
setError(int layer, AbstractOrdered2dData errors)
Sets the error of a certain layer. Set the error of the given layer. Arguments int layer [INPUT, MANDATORY] AbstractOrdered2dData errors [INPUT, MANDATORY]
setFlag(int layer, Flag flag)
Sets the flag of a certain layer. Set the flag of the given layer. Arguments int layer [INPUT, MANDATORY] Flag flag [INPUT, MANDATORY]
removeError(int layer)
Removes the error of a certain layer. Removes the error of the given layer. Argument int layer [INPUT, MANDATORY]
removeExposure(int layer)
Removes the exposure of a certain layer. Remove the exposure of the given layer. Argument int layer [INPUT, MANDATORY]
removeFlag(int layer)
Removes the flag of a certain layer. Removes the flag of the given layer.

```
removeFlag(int layer)
```

Argument

```
int layer [INPUT, MANDATORY]
```

```
double getIntensity(int layer, double row, double column)
```

Returns the intensity.

Returns the intensity of the stack at a given point. If the pixel is masked out, NaN is given back.

Arguments

```
int layer [INPUT, MANDATORY]
```

```
double row [INPUT, MANDATORY]
```

```
double column [INPUT, MANDATORY]
```

Return

```
double
```

The intensity

```
setIntensity(int layer, int row, int column, Number value)
```

Sets the intensity of the stack.

Sets the intensity of the stack at a given point.

Arguments

```
int layer [INPUT, MANDATORY]
```

```
int row [INPUT, MANDATORY]
```

```
int column [INPUT, MANDATORY]
```

```
Number value [INPUT, MANDATORY]
```

```
double getIntensityWorldCoordinates(int layer, double x, double y)
```

Returns the intensity of the stack

Returns the intensity of the stack at a given point in world coordinates. If the pixel is masked out, NaN is given back.

Arguments

```
int layer [INPUT, MANDATORY]
```

```
double x [INPUT, MANDATORY]
```

```
double y [INPUT, MANDATORY]
```

Return

```
double
```

The intensity

```
int[] getDimensions(int layer)
```

Return the dimensions of the layer.

Returns the dimension (width * height) of the given layer of this SimpleStack.

Argument

int[] getDimensions(int layer)

int layer [INPUT, MANDATORY]

Return

int[]

Returns the dimension (width * height) of the given layer this SimpleStack.

int getHeight(int layer)

Returns the height

Returns the height of the given layer of this SimpleStack.

Argument

int layer [INPUT, MANDATORY]

Return

int

The height of the given layer of this SimpleStack.

int getWidth(int layer)

Returns the width.

Returns the width of the given layer of this SimpleStack.

Argument

int layer [INPUT, MANDATORY]

Return

int

The width of the given layer of this SimpleStack.

Unit getUnit(int layer)

Returns the unit.

Returns the unit of the given layer of the stack. The unit of the errors on this layer of the stack is the same as the unit of the image.

Argument

int layer [INPUT, MANDATORY]

Return

Unit

The unit

setUnit(int layer, Unit unit)

Sets the unit.

Sets the unit of the given layer of the stack. Adapting the unit of the image will also adapt the unit of the errors on the given layer of the stack.

Arguments

setUnit(int layer, Unit unit)

```
int layer [INPUT, MANDATORY]
Unit unit [INPUT, MANDATORY]
```

setWavelength(int layer, double wavelength)

Sets the reference wavelength.

Set the reference wavelength of the given layer of the stack in microns.

Arguments

```
int layer [INPUT, MANDATORY]
double wavelength [INPUT, MANDATORY]
```

setWavelength(int layer, double wavelength, Length unit)

Sets the reference wavelength.

Set the reference wavelength of the given layer of the stack.

Arguments

```
int layer [INPUT, MANDATORY]
double wavelength [INPUT, MANDATORY]
Length unit [INPUT, MANDATORY]
```

double getWavelength(int layer)

Returns the reference wavelength

Returns the reference wavelength of the given layer of the stack.

Argument

```
int layer [INPUT, MANDATORY]
```

Return

double

The reference wavelength

double getWavelength(int layer, Length unit)

Returns the reference wavelength

Returns the reference wavelength of the given layer of the stack.

Arguments

```
int layer [INPUT, MANDATORY]
Length unit [INPUT, MANDATORY]
```

Return

double

The reference wavelength

[Wcs](#) getWcs(int layer)

Returns the world coordinates system.

Wcs getWcs(int layer)

Returns the World Coordinates System of the given layer of the stack.

Argument

`int layer` [INPUT, MANDATORY]

Return

[Wcs](#)

The world coordinates system

setWcs(int layer, Wcs wcs)

Sets the world coordinates system.

Sets the World Coordinates System of the given layer of the stack.

Arguments

`int layer` [INPUT, MANDATORY]

[Wcs](#) `wcs` [INPUT, MANDATORY]

Stack getPreview(float resolution)

Returns a preview of the stack

Returns a preview of a stack with the same number of layers, but with a decrease in resolution in width and height.

Argument

`float resolution` [INPUT, MANDATORY]

Return

Stack

The preview

removeImage(int layer)


Removes a layer.

Removes a layer from the stack.

Argument

`int layer` [INPUT, MANDATORY]

2.369. SincModel

Full Name:	herschel.ia.numeric.toolbox.fit.SincModel
Type:	Java Class - 
Import:	from herschel.ia.numeric.toolbox.fit import SincModel

Description

Sinc Model.

Also known as Cardinal Sine.

$$f(x;p) = p_0 * \sin((x - p_1) / p_2) / ((x - p_1) / p_2)$$

where p_0 = amplitude, p_1 = offset and p_2 = width (=Distance between first zero-crossings divided by 2 Pi.)

As always x = input.


The parameters are initialized at {1.0, 0.0, 1.0}.

Example

Example 1: SincModel

```
sinc = SincModel()
print sinc.getNumberOfParameters()      # 3
print sinc( DoubleId.range(15)-7 -)    # sinc function between [-7,+7]
```

2.370. SineAmpModel

Full Name:	herschel.ia.numeric.toolbox.fit.SineAmpModel
Type:	Java Class - 
Import:	from herschel.ia.numeric.toolbox.fit import SineAmpModel

Description

Find amplitudes/phases for sinusoidal of a given frequency.


$$f(x;p) = p_0 \cos(2 \pi x) + p_1 \sin(2 \pi x)$$

It is a linear model.

Example

Example 1: SineAmpModel
<pre>sine = SineAmpModel(150 -) # fixed frequency of 150 Hz # Use a (linear)Fitter</pre>

2.371. SineModel

Full Name:	herschel.ia.numeric.toolbox.fit.SineModel
Type:	Java Class - 
Import:	from herschel.ia.numeric.toolbox.fit import SineModel

Description

Sinusoidal Model.

$$f(x;p) = p_1 * \cos(2 * \# * p_0 * x) + p_2 * \sin(2 * \# * p_0 * x)$$

where p_0 = frequency, p_1 = amplitude cosine and p_2 = amplitude sine. As always x = input.

The parameters are initialized at {1.0, 1.0, 1.0}. It is a non-linear model.


See [example](#)

Example

Example 1: SineModel

```
sine = SineModel()
print sine.getNumberOfParameters()      # 3
pars = Doubleld( [0.1,0,1] -)
print sine.setParameters( pars -)
print sine( Doubleld.range(11) -)      # One sine period
pars = Doubleld( [0.1,1,0] -)
print sine.setParameters( pars -)
print sine( Doubleld.range(11) -)      # One cosine period
```

2.372. SingularValueDecompositionFitter

Full Name:	herschel.ia.numeric.toolbox.fit.SingularValueDecompositionFitter
Type:	Java Class - 
Import:	from herschel.ia.numeric.toolbox.fit import SingularValueDecompositionFitter

Description

SingularValueDecompositionFitter implements a linear fitter based on

Singular Value Decomposition (SVD).

SVDFitter is much more robust in case of (nearly) degenerated models. Of course at the cost of more CPU use.

See Numerical Recipes for more information.

Example


Example 1: SingularValueDecompositionFitter (for linear models.)

```
# assume x and y are Double1d data arrays.
poly = PolynomialModel( 1 -)          # line
svdfit = SingularValueDecompositionFitter( x, poly -)
svdfit.setThreshold( 1e-8 -)         # for detecting degeneracy
param = svdfit.fit( y -)
print svdfit.hasDegeneracy()
stdev = svdfit.getStandardDeviation() # stdevs on the parameters
chisq = svdfit.getChiSquared()
svd    = svdfit.getSvd()              # the SVD decomposition
```

Limitations

The SVDFitter does **not** work with limits.

2.373. SingularValueDecomposition

Full Name:	herschel.ia.numeric.toolbox.matrix.SingularValueDecomposition
Alias:	SingularValueDecomposition
Type:	Java Class - 
Import:	from herschel.ia.numeric.toolbox.matrix import SingularValueDecomposition

Example

Example 1:

```
##### SingularValueDecomposition #####
#####
#
### HIPE ###
A = Double2d([[ 2.0,  1.0,  1.0],[ 4.0, --6.0,  0.0],[ -2.0,  7.0,  2.0]])
svd=A.apply(SingularValueDecomposition())
leftSingularValues = svd.u
print leftSingularValues
[
[0.1938226555206747,-0.816496580927726,0.5438438301022152],
[0.47224728623497236,-0.408248290463863,-0.7812271334106601],
[0.8598925972763216,0.4082482904638631,0.3064605267937704]
]
rightSingularValues = svd.v
print rightSingularValues
[
[0.19382265552067465,-0.816496580927726,0.5438438301022155],
[0.47224728623497236,-0.40824829046386335,-0.78122713341066],
[0.8598925972763218,0.40824829046386324,0.3064605267937702]
]
singularValues = svd.singularValues
print singularValues
array([7.87298334620742, 1.0000000000000004, 0.12701665379258284], double)
diagonalMatrixOfSingularValues = svd.s
print diagonalMatrixOfSingularValues
[
[7.87298334620742,0.0,0.0],
[0.0,1.0000000000000004,0.0],
[0.0,0.0,0.12701665379258284]
]
norm2 = svd.norm2()
print norm2
7.87298334620742
cond = svd.cond()
print cond
61.98386676965949
rank = svd.rank()
print rank
3.0
```

API Summary

Jython Syntax

```
A=Double2d()
svd=B.apply(SingularValueDecomposition())
leftSingularValues = svd.u
rightSingularValues = svd.v
singularValues = svd.singularValues
```

Jython Syntax

```
diagonalMatrixOfSingularValues = svd.s  
norm2 = svd.norm2  
cont = cvs.cond  
rank = svd.rank
```

Property

[Double2d](#) **A** [INPUT, MANDATORY, default=no default value]


API details

Property

[Double2d](#) **A** [INPUT, MANDATORY, default=no default value]

Input must be a Double2d or Float2d array.

2.374. SIN

Full Name:	herschel.ia.numeric.toolbox.basic.Sin
Alias:	SIN
Type:	Java Class - 
Import:	from herschel.ia.numeric.toolbox.basic import Sin

Description

Computes the trigonometric sine of a number or array

Gives the sine of a number or array. The input is in radians, and may be of any type. Returns a float array for float input array, complex array for input complex array, and double array for any other numeric array type.

Example

Example 1: Apply SIN on a Float1d

```
x=Float1d([0,0.5])
print SIN(x) # [0.0,0.47942555]
```

API Summary

Jython Syntax

```
<y>=SIN(<x>)
```

Properties

[any type **x** \[INPUT, MANDATORY, default=no default value\]](#)

[any type **y** \[OUTPUT, NOT_MANDATORY, default=no default value\]](#)

API details

Properties

any type **x [INPUT, MANDATORY, default=no default value]**

The input is in radians, and may be of any type.


any type **y [OUTPUT, NOT_MANDATORY, default=no default value]**

Returns a float array for float input array, complex array for input complex array, and double array for any other numeric array type.

See also

- [ARCSIN](#)
- [COS](#)
- [SINH](#)
- [TAN](#)

2.375. SINH

Full Name:	herschel.ia.numeric.toolbox.basic.SinH
Alias:	SINH
Type:	Java Class - 
Import:	from herschel.ia.numeric.toolbox.basic import SinH

Description

Computes the trigonometric hyperbolic sine of an number or array

Gives the hyperbolic sine of a number or array. The input is in radians, and may be of any type. Returns a float array for float input array, complex array for input complex array, and double array for any other numeric array type.

API Summary

Jython Syntax
<code><y>=SINH (<x>)</code>

Properties
<code>any type x [INPUT, MANDATORY, default=no default value]</code>
<code>any type y [OUTPUT, NOT_MANDATORY, default=no default value]</code>

Miscellaneous

Does not work for complex values.

API details

Properties


<code>any type x [INPUT, MANDATORY, default=no default value]</code>
The input is in radians, and may be of any type.

<code>any type y [OUTPUT, NOT_MANDATORY, default=no default value]</code>
Returns a float array for float input array, complex array for input complex array, and double array for any other numeric array type.

See also

- [SIN](#)
- [ARCSIN](#)

2.376. SkewGaussModel

Full Name:	herschel.ia.numeric.toolbox.fit.SkewGaussModel
Type:	Java Class - 
Import:	from herschel.ia.numeric.toolbox.fit import SkewGaussModel

Description

A class that defines a skew gaussian model.

A skew gaussian model is defined as $f(x;p) = p_0 * \exp(-0.5 * ((x - p_1) / p_2)^{**2}) * (1 + \operatorname{erf}(p_3 * ((x - p_1) / p_2) / \operatorname{SQRT}(2)))$ with $p_0 = \text{amplitude}$ $p_1 = \text{x-shift}$ $p_2 = \text{sigma}$ $p_3 = \text{scale, "skew"}$

API Summary

Constructor
SkewGaussModel() The construction of a new SkewGaussModel.
Method
double result(double input, DoubleId param) Returns the result of this SkewGaussModel.

API details


Constructor

<code>SkewGaussModel()</code>
The construction of a new SkewGaussModel.
A new SkewGaussModel is constructed with its parameters set to [1.0, 0.0, 1.0, 0.0].

Method

<code>double result(double input, DoubleId param)</code>
Returns the result of this SkewGaussModel.
Returns the result of this SkewGaussModel for the given input and the given parameters.
Arguments
double input [INPUT, MANDATORY] DoubleId param [INPUT, MANDATORY]
Return
double
Returns the result of this SkewGaussModel for the given input and fit parameters.

2.377. SKEWNESS

Full Name:	herschel.ia.numeric.toolbox.basic.Skewness
Alias:	SKEWNESS
Type:	Java Class - 
Import:	from herschel.ia.numeric.toolbox.basic import Skewness

Description

Yields the skewness of the elements in the input array.

The skewness is defined as the third moment divided by the standard deviation raised to the third power.

Example

Example 1: Apply SKEWNESS on a Float1d

```
x=Float1d([1,3,2,3,4])
print SKEWNESS(x) # --0.19430208
```

API Summary

Jython Syntax

```
<y>=SKEWNESS(<x>)
```

Properties

[any scalar array **x** \[INPUT, MANDATORY, default=no default value\]](#)

[double **y** \[OUTPUT, MANDATORY, default=no default value\]](#)

API details

Properties

any scalar array **x [INPUT, MANDATORY, default=no default value]**

The input array integral or floating-point arrays; the former implicitly transformed to a double array.

double **y [OUTPUT, MANDATORY, default=no default value]**


Returns a double

See also

- [KURTOSIS](#)
- [MEAN](#)
- [MEDIAN](#)
- [STDDEV](#)

- [VARIANCE](#)

2.378. SkyAperturePhotometryExplorer

Full Name:	herschel.ia.gui.image.SkyAperturePhotometryExplorer
Type:	Java Class - 
Import:	from herschel.ia.gui.image import SkyAperturePhotometryExplorer

Description

An explorer for SkyAperturePhotometryProducts.

An explorer for SkyAperturePhotometryProducts.

API Summary

Constructors
SkyAperturePhotometryExplorer() The construction of a new SkyAperturePhotometryExplorer.
SkyAperturePhotometryExplorer(Object object) The construction of a new SkyAperturePhotometryExplorer associated with the given object.
Methods
Class getVariableType() Returns the expected variable type.
JScrollPane getResultsTable() Returns the results table.

API details

Constructors

SkyAperturePhotometryExplorer() The construction of a new SkyAperturePhotometryExplorer. The construction of a new SkyAperturePhotometryExplorer.
SkyAperturePhotometryExplorer(Object object) The construction of a new SkyAperturePhotometryExplorer associated with the given object. The construction of a new SkyAperturePhotometryExplorer associated with the given object. Argument Object object [INPUT, MANDATORY]

Methods

Class getVariableType() Returns the expected variable type. Returns the expected variable type for this SkyAperturePhotometryExplorer.

[Class](#) `getVariableType()`

Return

[Class](#)

Returns the expected variable type for this SkyAperturePhotometryExplorer.

[JScrollPane](#) `getResultsTable()`

Returns the results table.


Constructs and returns the results table for this SkyAperturePhotometryExplorer.

Return

[JScrollPane](#)

Returns the results table for this SkyAperturePhotometryExplorer.

2.379. SkyAperturePhotometryProduct

Full Name:	herschel.ia.dataset.image.SkyAperturePhotometryProduct
Type:	Java Class - 
Import:	from herschel.ia.dataset.image import SkyAperturePhotometryProduct

Description

A class to deal with the results of aperture photometry with a circular target aperture and an annular/rectangular sky aperture.

API Summary

Constructor
SkyAperturePhotometryProduct() The constructor of a new SkyAperturePhotometryProduct.
Methods
setAlgorithm(int index) Sets the sky estimation algorithm for this SkyAperturePhotometryProduct to the
setResultsTable(Double2d resultsTable) Sets the results table for this SkyAperturePhotometryProduct to the
String getAlgorithm() Returns the String representation of the sky estimation algorithm for this
double getSkyTotal() Returns the total flux for this SkyAperturePhotometryProduct.
double getNbOfSkyPixels() Returns the number of pixels for the sky for this SkyAperturePhotometryProduct.
double getSkyError() Returns the error for the sky for this SkyAperturePhotometryProduct.
double getTargetTotal() Returns the total flux for the target (sky subtracted) for this
double getNbOfTargetPixels() Returns the number of pixels for the target (sky subtracted) for this
double getIntensityPerTargetPixel() Returns the intensity per pixel for the target (sky subtracted) for this
double getTargetError() Returns the error for the target (sky subtracted) for this

Miscellaneous

Aperture specific information is stored in meta data, all other data (results of calculations) are stored in TableDatasets.

API details

Constructor

SkyAperturePhotometryProduct()

The constructor of a new SkyAperturePhotometryProduct.

Methods

setAlgorithm(int index)

Sets the sky estimation algorithm for this SkyAperturePhotometryProduct to the algorithm with the given index.

Argument

int **index** [INPUT, MANDATORY]

setResultsTable(Double2d resultsTable)

Sets the results table for this SkyAperturePhotometryProduct to the given table.

Argument

[Double2d](#) **resultsTable** [INPUT, MANDATORY]

String getAlgorithm()

Returns the String representation of the sky estimation algorithm for this SkyAperturePhotometryProduct.

Return

[String](#)

Returns the String representation of the sky estimation algorithm for this SkyAperturePhotometryProduct.

double getSkyTotal()

Returns the total flux for this SkyAperturePhotometryProduct.

Return

double

Returns the total flux for this SkyAperturePhotometryProduct.

double getNbOfSkyPixels()

Returns the number of pixels for the sky for this SkyAperturePhotometryProduct.

Return

double

Returns the number of pixels for the sky for this SkyAperturePhotometryProduct.

double getSkyError()
Returns the error for the sky for this SkyAperturePhotometryProduct.
Return
double
Returns the error for the sky for this SkyAperturePhotometryProduct.


double getTargetTotal()
Returns the total flux for the target (sky subtracted) for this SkyAperturePhotometryProduct.
Return
double
Returns the total flux for the target (sky subtracted) for this SkyAperturePhotometryProduct.

double getNbOfTargetPixels()
Returns the number of pixels for the target (sky subtracted) for this SkyAperturePhotometryProduct.
Return
double
Returns the number of pixels for the target (sky subtracted) for this SkyAperturePhotometryProduct.

double getIntensityPerTargetPixel()
Returns the intensity per pixel for the target (sky subtracted) for this SkyAperturePhotometryProduct.
Return
double
Returns the intensity per pixel for the target (sky subtracted) for this SkyAperturePhotometryProduct.

double getTargetError()
Returns the error for the target (sky subtracted) for this SkyAperturePhotometryProduct.
Return
double
Returns the error for the target (sky subtracted) for this SkyAperturePhotometryProduct.

2.380. SkyAperturePhotometryTask

Full Name:	herschel.ia.toolbox.image.SkyAperturePhotometryTask
Type:	Java Task - 
Import:	from herschel.ia.toolbox.image import SkyAperturePhotometryTask

Description

An abstract Task for aperture photometry with a sky aperture.

An abstract Task for aperture photometry, using a circular target aperture. The subclasses/subtasks are `AnnularSkyAperturePhotometryTask` and `RectangularSkyAperturePhotometryTask`, using an annular and a rectangular sky aperture respectively.

API Summary

Properties
Image image [INPUT, MANDATORY, default=No default value]
Double centerX [INPUT, OPTIONAL, default=Default value : NaN]
Double centerY [INPUT, OPTIONAL, default=Default value : NaN]
String centerRA [INPUT, OPTIONAL, default=Default value : "center-RA"]
String centerDec [INPUT, OPTIONAL, default=Default value : "centerDec"]
Double radiusPixels [INPUT, OPTIONAL, default=Default value : NaN]
Integer radiusArcsec [INPUT, OPTIONAL, default=Default value : 4]
String algorithm [INPUT, MANDATORY, default=Default value : NaN]
boolean fractional [INPUT, OPTIONAL, default=Default value : true]
AperturePhotometryProduct result [OUTPUT, MANDATORY, default=No default value]

Miscellaneous

The calculation of the error on the fluxes is calculated in the same way as in the `aper.pro` routine of IDL. Three factors contribute to the error on the target flux : * scatter in sky values * random photon noise * uncertainty in mean sky brightness The error for the target flux including the background is defined as the sqrt of the (absolute value of the) total flux in the target aperture (including the background). The error for the sky is defined as the standard deviation on the sky value. Note that this standard deviation is calculated only with the sky intensities that are used to calculate the sky intensity (as there are sky estimation algorithms that reject sky values). The squared error for the target flux is thus : * the absolute value of the target flux of which the sky contribution has been subtracted * the standard deviation of the sky value * the surface of the target aperture * the squared standard deviation on the sky value


API details

Properties

Image image [INPUT, MANDATORY, default=No default value]
The image.

Double centerX [INPUT, OPTIONAL, default=Default value : NaN]
The x-pixel-coordinate of the target center.
Double centerY [INPUT, OPTIONAL, default=Default value : NaN]
The y-pixel-coordinate of the target center.
String centerRA [INPUT, OPTIONAL, default=Default value : "centerRA"]
The right ascension of the target center.
String centerDec [INPUT, OPTIONAL, default=Default value : "centerDec"]
The declination of the target center.
Double radiusPixels [INPUT, OPTIONAL, default=Default value : NaN]
The radius of the circular target aperture in pixels.
Integer radiusArcsec [INPUT, OPTIONAL, default=Default value : 4]
The radius of the circular target aperture in arcsec.
String algorithm [INPUT, MANDATORY, default=Default value : NaN]
The sky estimation algorithm.
boolean fractional [INPUT, OPTIONAL, default=Default value : true]
The type of pixels.
AperturePhotometryProduct result [OUTPUT, MANDATORY, default=No default value]
The result.

2.381. SlicedCube

Full Name:	herschel.ia.slice.image.SlicedCube
Type:	Java Class - 
Import:	from herschel.ia.slice.image import SlicedCube
Category:	Image

Description

A SlicedCube is a large cube which is stored in slices to make the memory usage lower.

A SlicedCube is a large cube which is stored in slices to make the memory usage lower.

Example

Example 1: Typical example on how to create a SlicedCube.

```
cube = SlicedCube(description="My large cube")
cube.setImage(myData)
```

API Summary

Constructors	
SlicedCube()	A constructor which creates a standard SlicedCube. The cube is sliced
SlicedCube(SlicedCube copy)	The copy constructor
SlicedCube(String description)	The constructor with a description
Methods	
copy()	Returns a copy.
setSliceSize(int depth, int row, int column)	Sets the size of the slices of the cube.
int[] getSliceSize()	Returns the size of the slices of the cube.
SimpleCube getSlice(int sliceNumber)	Returns the chosen slice.
setImage(AbstractOrdered3dData cube, Unit unit)	Sets the cube.
Wcs getWcs()	Returns the World Coordinates System.
setWcs(Wcs wcs)	Sets the world coordinates system.
int[] getDimensions()	Returns the dimensions.
setImage(AbstractOrdered3dData cube)	

Methods
Sets the cube.
int getHeight() Returns the height.
int getWidth() Returns the width
int getDepth() Returns the depth
Unit getUnit() Returns the unit.
setUnit(Unit unit) Sets the unit.
getIntensity(int depth, double row, double column) Returns the intensity
Double getIntensityWorldCoordinates(int i, double x, double y) Returns the intensity of the cube
Number getPixel(int depth, int row, int column) Returns 1 pixel value.
setIntensity(int depth, int row, int column, Number value) Sets the intensity of the cube.
AbstractOrdered3dData getImage(int sliceNumber) returns the image data of the slice as a Numeric3d.
AbstractOrdered3dData getImage() returns the image as a Numeric2d.
boolean hasCoverage() Checks whether the cube has a coverage.
boolean hasError() Checks whether the cube has an error.
setCoverage(AbstractOrdered3dData coverage) Sets the coverage.
setError(AbstractOrdered3dData error) Sets the error.
removeError() Removes the error.
removeCoverage() Removes the coverage.
Number getCoverage(int depth, int row, int column) Returns the coverage of 1 pixel.
AbstractOrdered3dData getCoverage(int sliceNumber) returns the coverage of the slice as a Numeric3d.
AbstractOrdered3dData getCoverage() Returns the coverage of the cube as a Numeric3d.

Methods
Number <code>getError(int depth, int row, int column)</code> Returns the error of 1 pixel.
AbstractOrdered3dData <code>getError(int sliceNumber)</code> returns the error of the slice as a Numeric3d.
AbstractOrdered3dData <code>getError()</code> Returns the error of the cube as a Numeric3d.
boolean <code>hasExposure()</code> Checks whether the cube has an exposure.
setExposure(AbstractOrdered3dData exposure) Sets the exposure.
setExposure(AbstractOrdered3dData exposure, String description) Sets the exposure.
removeExposure() Removes the exposure.
Number <code>getExposure(int depth, int row, int column)</code> Returns the exposure of 1 pixel.
AbstractOrdered3dData <code>getExposure(int sliceNumber)</code> returns the exposure of the slice as a Numeric3d.
AbstractOrdered3dData <code>getExposure()</code> Returns the exposure of the cube as a Numeric3d.
boolean <code>hasFlag()</code> Checks when the image has a flag.
setFlag(Flag flag) Sets the flag.
setFlag(Flag flag, String description) Sets the flag.
removeFlag() Removes the flag.
boolean <code>getFlag(int depth, int row, int column)</code> Returns the flag of 1 pixel.
Flag <code>getFlag(int sliceNumber)</code> returns the flag of the slice.
Flag <code>getFlag()</code> Returns the flag of the cube.
Cube <code>getPreview(float res)</code> Returns a preview of the cube
Double3d <code>getImageData()</code> Returns the Image data.

API details

Constructors

<code>SlicedCube()</code>
<p>A constructor which creates a standard SlicedCube. The cube is sliced in pieces of 50x50x50 pixels. Every slice is kept in a ListContext as a SimpleCube. The usage of a SlicedCube should be transparent to the user. Exactly the same methods are available on a SlicedCube as on a SimpleCube.</p> <p>Example</p> <p>Typical example on how to create a SlicedCube.</p> <pre>cube=SlicedCube(description="ngc 6992", image=im, flag=flag, errors=er, quantity=quant, wavelength=wavelength, wcs=wcs)</pre>

<code>SlicedCube(SlicedCube copy)</code>
<p>The copy constructor</p> <p>Makes a new SlicedCube from a copy of the given Cube.</p> <p>Argument</p> <p><code>SlicedCube copy</code> [INPUT, MANDATORY]</p>

<code>SlicedCube(String description)</code>
<p>The constructor with a description</p> <p>This constructor also sets the description of the SlicedCube</p> <p>Argument</p> <p><code>String description</code> [INPUT, MANDATORY]</p>

Methods

<code>copy()</code>
<p>Returns a copy.</p> <p>Returns a copy from this SlicedCube.</p>

<code>setSliceSize(int depth, int row, int column)</code>
<p>Sets the size of the slices of the cube.</p> <p>Sets the size of the slices of the cube. The standard size is 50x50x50. At this moment, the size should be set before adding data to the SlicedImage!</p> <p>Arguments</p> <p><code>int depth</code> [INPUT, MANDATORY]</p> <p><code>int row</code> [INPUT, MANDATORY]</p> <p><code>int column</code> [INPUT, MANDATORY]</p>

<code>int[] getSliceSize()</code>
<p>Returns the size of the slices of the cube.</p>

int[] getSlicesSize()
Returns the size of the slices of the cube. The standard size is 50x50x50.
Return
int[]
The size of the slices (in depth, rows and columns)
SimpleCube getSlice(int sliceNumber)
Returns the chosen slice.
Returns the chosen slice of the cube as a SimpleCube.
Argument
int sliceNumber [INPUT, MANDATORY]
Return
SimpleCube
The chosen slice of the Cube.
setImage(AbstractOrdered3dData cube, Unit unit)
Sets the cube.
Sets the cube. You can give a unit. The given unit will automatically also be used for the errors on this cube. If the new cube has other dimensions than the original cube, the errors, exposure and flag are removed. If the new cube has the same dimensions, then the old values for the errors, exposures and flag are kept (but the unit of the errors is not). The cube data will be automatically sliced.
Arguments
AbstractOrdered3dData cube [INPUT, MANDATORY]
Unit unit [INPUT, MANDATORY]
Wcs getWcs()
Returns the World Coordinates System.
Returns the World Coordinates System of the cube.
Return
Wcs
The World Coordinates System of the cube.
setWcs(Wcs wcs)
Sets the world coordinates system.
Sets the world coordinates system of the cube.
Argument
Wcs wcs [INPUT, MANDATORY]
int[] getDimensions()
Returns the dimensions.

int[] getDimensions()

Returns the dimensions (depth, width, height) of this SlicedCube.

Return

int[]

Returns the dimensions (depth, width, height) of this SlicedCube.

setImage(AbstractOrdered3dData cube)

Sets the cube.

Sets the cube. The description is SlicedCube and the unit is Scalar.ONE. The given unit will automatically also be used for the errors on this cube. If the new cube has other dimensions than the original cube, the errors, exposure and flag are removed. If the new cube has the same dimensions, then the old values for the errors, exposures and flag are kept (but the unit of the errors is not). The cube data will be automatically sliced.

Argument

AbstractOrdered3dData **cube** [INPUT, MANDATORY]

int getHeight()

Returns the height.

Returns the height of this SimpleCube.

Return

int

The height of this SimpleCube.

int getWidth()

Returns the width

Returns the width of this SlicedCube.

Return

int

The width of this SlicedCube.

int getDepth()

Returns the depth

Returns the depth of this SlicedCube.

Return

int

The depth of this SlicedCube.

Unit getUnit()

Returns the unit.

Unit <code>getUnit()</code>
Returns the unit of the cube. The unit of the errors of this cube is the same as the unit of the cube.
Return
Unit
The unit
setUnit(Unit unit)
Sets the unit.
Sets the unit of the cube. Adapting the unit of the cube will also adapt the unit of the errors on the cube.
Argument
Unit unit [INPUT, MANDATORY]
getIntensity(int depth, double row, double column)
Returns the intensity
Returns the intensity of the SimpleCube at a given point. If the pixel is flagged out, NaN is given back.
Arguments
int depth [INPUT, MANDATORY]
double row [INPUT, MANDATORY]
double column [INPUT, MANDATORY]
Double <code>getIntensityWorldCoordinates(int i, double x, double y)</code>
Returns the intensity of the cube
Returns the intensity of the cube at a given point in world coordinates. If the pixel is flagged out, NaN is given back.
Arguments
int i [INPUT, MANDATORY]
double x [INPUT, MANDATORY]
double y [INPUT, MANDATORY]
Return
Double
The intensity
Number <code>getPixel(int depth, int row, int column)</code>
Returns 1 pixel value.
Returns 1 pixel value of the cube.
Arguments
int depth [INPUT, MANDATORY]
int row [INPUT, MANDATORY]
int column [INPUT, MANDATORY]

Number `getPixel(int depth, int row, int column)`

Return

Number

1 pixel value of the cube.

setIntensity(int depth, int row, int column, Number value)

Sets the intensity of the cube.

Sets the intensity of the cube at a given point.

Arguments

int **depth** [INPUT, MANDATORY]

int **row** [INPUT, MANDATORY]

int **column** [INPUT, MANDATORY]

Number value [INPUT, MANDATORY]

AbstractOrdered3dData `getImage(int sliceNumber)`

returns the image data of the slice as a Numeric3d.

Returns the image data as a Numeric3d containing the data of the slice of the cube.

Argument

int **sliceNumber** [INPUT, MANDATORY]

Return

AbstractOrdered3dData

The image data of the slice as a Numeric3d

AbstractOrdered3dData `getImage()`

returns the image as a Numeric2d.

Returns the image as a Numeric2d containing the data of the image.

Return

AbstractOrdered3dData

The image as a Numeric2d

boolean `hasCoverage()`

Checks whether the cube has a coverage.

Returns true if the cube has a coverage.

Return

boolean

True if the coverage is set.

boolean `hasError()`

Checks whether the cube has an error.

boolean hasError()
Returns true if the cube has an error.
Return
boolean
True if the error is set.
setCoverage(AbstractOrdered3dData coverage)
Sets the coverage.
Sets the coverages of the cube. The Numeric3d containing the coverages should have the same dimensions as the cube. If this is not the case, the coverages will not be adapted.
Argument
AbstractOrdered3dData coverage [INPUT, MANDATORY]
setError(AbstractOrdered3dData error)
Sets the error.
Sets the errors of the cube. The Numeric3d containing the errors should have the same dimensions as the cube. If this is not the case, the errors will not be adapted.
Argument
AbstractOrdered3dData error [INPUT, MANDATORY]
removeError()
Removes the error.
Removes the error, if an error exists.
removeCoverage()
Removes the coverage.
Removes the coverage, if an error exists.
Number getCoverage(int depth, int row, int column)
Returns the coverage of 1 pixel.
Returns the coverage of 1 pixel.
Arguments
int depth [INPUT, MANDATORY]
int row [INPUT, MANDATORY]
int column [INPUT, MANDATORY]
Return
Number
The coverage of 1 pixel.
AbstractOrdered3dData getCoverage(int sliceNumber)
returns the coverage of the slice as a Numeric3d.

AbstractOrdered3dData <code>getCoverage(int sliceNumber)</code>
Returns the coverage as a Numeric3d.
Argument
int sliceNumber [INPUT, MANDATORY]
Return
AbstractOrdered3dData
The coverage of the slice as a Numeric3d

AbstractOrdered3dData <code>getCoverage()</code>
Returns the coverage of the cube as a Numeric3d.
Returns the coverage of the cube as a Numeric3d containing the coverage of every pixel.
Return
AbstractOrdered3dData
The coverage

Number <code>getError(int depth, int row, int column)</code>
Returns the error of 1 pixel.
Returns the error of 1 pixel.
Arguments
int depth [INPUT, MANDATORY]
int row [INPUT, MANDATORY]
int column [INPUT, MANDATORY]
Return
Number
The error of 1 pixel.

AbstractOrdered3dData <code>getError(int sliceNumber)</code>
returns the error of the slice as a Numeric3d.
Returns the error as a Numeric3d.
Argument
int sliceNumber [INPUT, MANDATORY]
Return
AbstractOrdered3dData
The error of the slice as a Numeric3d

AbstractOrdered3dData <code>getError()</code>
Returns the error of the cube as a Numeric3d.
Returns the error of the cube as a Numeric3d containing the error of every pixel.

AbstractOrdered3dData <code>getError()</code>
<p>Return</p> <p>AbstractOrdered3dData</p> <p>The error</p>
boolean <code>hasExposure()</code>
<p>Checks whether the cube has an exposure.</p> <p>Returns true if the cube has an exposure.</p> <p>Return</p> <p>boolean</p> <p>True if the exposure is set.</p>
setExposure(AbstractOrdered3dData exposure)
<p>Sets the exposure.</p> <p>Sets the exposure of every pixel of the cube. The Numeric3d containing the exposure should have the same dimensions as the cube. If this is not the case, the exposure will not be adapted.</p> <p>Argument</p> <p>AbstractOrdered3dData exposure [INPUT, MANDATORY]</p>
setExposure(AbstractOrdered3dData exposure, String description)
<p>Sets the exposure.</p> <p>Sets the exposure of every pixel of the cube. The Numeric3d containing the exposure should have the same dimensions as the cube. If this is not the case, the exposure will not be adapted.</p> <p>Arguments</p> <p>AbstractOrdered3dData exposure [INPUT, MANDATORY]</p> <p>String description [INPUT, MANDATORY]</p>
removeExposure()
<p>Removes the exposure.</p> <p>Removes the exposure, if an exposure exists.</p>
Number getExposure(int depth, int row, int column)
<p>Returns the exposure of 1 pixel.</p> <p>Returns the exposure of 1 pixel.</p> <p>Arguments</p> <p>int depth [INPUT, MANDATORY]</p> <p>int row [INPUT, MANDATORY]</p> <p>int column [INPUT, MANDATORY]</p> <p>Return</p> <p>Number</p>

Number `getExposure(int depth, int row, int column)`

The exposure of 1 pixel.

AbstractOrdered3dData `getExposure(int sliceNumber)`

returns the exposure of the slice as a Numeric3d.

Returns the exposure as a Numeric3d.

Argument

`int sliceNumber` [INPUT, MANDATORY]

Return

AbstractOrdered3dData

The exposure of the slice as a Numeric3d

AbstractOrdered3dData `getExposure()`

Returns the exposure of the cube as a Numeric3d.

Returns the exposure of the cube as a Numeric3d containing the exposure of every pixel.

Return

AbstractOrdered3dData

The exposure

boolean `hasFlag()`

Checks when the image has a flag.

Returns true if the image has a flag.

Return

boolean

True if the image has a flag.

setFlag(Flag flag)

Sets the flag.

Sets the flag of every pixel of the cube. The Flag should have the same dimensions as the cube. If this is not the case, the Flag will not be adapted.

Argument

`Flag flag` [INPUT, MANDATORY]

setFlag(Flag flag, String description)

Sets the flag.

Sets the flag of every pixel of the cube. The Flag should have the same dimensions as the cube. If this is not the case, the Flag will not be adapted. The flag is also described.

Arguments

`Flag flag` [INPUT, MANDATORY]

setFlag([Flag](#) flag, [String](#) description)

[String](#) description [INPUT, MANDATORY]

removeFlag()

Removes the flag.

Removes the flag, if a flag exists.

boolean getFlag(int depth, int row, int column)

Returns the flag of 1 pixel.

Returns the flag of 1 pixel.

Arguments

int **depth** [INPUT, MANDATORY]

int **row** [INPUT, MANDATORY]

int **column** [INPUT, MANDATORY]

Return

boolean

The flag of 1 pixel. True if flagged out.

[Flag](#) getFlag(int sliceNumber)

returns the flag of the slice.

Returns the flag.

Argument

int **sliceNumber** [INPUT, MANDATORY]

Return

[Flag](#)

The flag of the slice

[Flag](#) getFlag()

Returns the flag of the cube.

Returns the flag of the cube.

Return

[Flag](#)

The flag

Cube getPreview(float res)

Returns a preview of the cube

Returns a new SlicedImage, but with a lower resolution if res is smaller than 1. A higher resolution is returned, if res is larger than 1. The spatial regridding is based on linear interpolation which will, in general, not be suitable for scientific purposes.

Argument

Cube `getPreview(float res)`

float **res** [INPUT, MANDATORY]

Return

Cube

The preview of the cube

Double3d `getImageData()`

Returns the Image data.


Returns the cube data as a Double3d.

Return

Double3d

The cube data as Double3d

2.382. SlicedImage

Full Name:	herschel.ia.slice.image.SlicedImage
Type:	Java Class - 
Import:	from herschel.ia.slice.image import SlicedImage
Category:	Image

Description

A SlicedImage is a large image which is stored in slices to make the memory usage lower.

A SlicedImage is a large image which is stored in slices to make the memory usage lower.

Example

Example 1: Typical example on how to create a SlicedImage.

```
image = SlicedImage(description="My large image")
image.setImage(myData)
```

API Summary

Constructors	
SlicedImage()	A constructor which creates a standard SlicedImage. The image is sliced
SlicedImage(SlicedImage copy)	The copy constructor
SlicedImage(String description)	The constructor with a description
Methods	
copy()	Returns a copy.
setSliceSize(int row, int column)	Sets the size of the slices of the image.
int[] getSliceSize()	Returns the size of the slices of the image.
SimpleImage getSlice(int sliceNumber)	Returns the chosen slice.
setImage(AbstractOrdered2dData image, Unit unit)	Sets the image.
setWavelength(double wavelength)	Sets the reference wavelength.
setWavelength(double wavelength, Length unit)	Sets the reference wavelength.
double getWavelength()	Returns the reference wavelength
double getWavelength(Length unit)	

Methods
Returns the reference wavelength
Wcs getWcs() Returns the World Coordinates System.
setWcs(Wcs wcs) Sets the world coordinates system.
int[] getDimensions() Returns the dimension.
setImage(AbstractOrdered2dData image) Sets the image.
int getHeight() Returns the height.
int getWidth() Returns the width
Unit getUnit() Returns the unit.
setUnit(Unit unit) Sets the unit.
getIntensity(double row, double column) Returns the intensity
double getIntensityWorldCoordinates(double x, double y) Returns the intensity of the image
Number getPixel(int row, int column) Returns 1 pixel value.
setIntensity(int row, int column, Number value) Sets the intensity of the image.
AbstractOrdered2dData getImage(int sliceNumber) returns the image data of the slice as a Numeric2d.
AbstractOrdered2dData getImage() returns the image as a Numeric2d.
boolean hasCoverage() Checks whether the image has an coverage.
boolean hasError() Checks whether the image has an error.
setCoverage(AbstractOrdered2dData coverage) Sets the coverage.
setError(AbstractOrdered2dData error) Sets the error.
removeCoverage() Removes the coverage.
removeError() Removes the error.

Methods
Number <code>getCoverage(int row, int column)</code> Returns the coverage of 1 pixel.
AbstractOrdered2dData <code>getCoverage(int sliceNumber)</code> returns the coverage of the slice as a Numeric2d.
AbstractOrdered2dData <code>getCoverage()</code> Returns the coverage of the image as a Numeric2d.
Number <code>getError(int row, int column)</code> Returns the error of 1 pixel.
AbstractOrdered2dData <code>getError(int sliceNumber)</code> returns the error of the slice as a Numeric2d.
AbstractOrdered2dData <code>getError()</code> Returns the error of the image as a Numeric2d.
boolean <code>hasExposure()</code> Checks whether the image has an exposure.
setExposure(AbstractOrdered2dData exposure) Sets the exposure.
setExposure(AbstractOrdered2dData exposure, String description) Sets the exposure.
removeExposure() Removes the exposure.
Number <code>getExposure(int row, int column)</code> Returns the exposure of 1 pixel.
AbstractOrdered2dData <code>getExposure(int sliceNumber)</code> returns the exposure of the slice as a Numeric2d.
AbstractOrdered2dData <code>getExposure()</code> Returns the exposure of the image as a Numeric2d.
boolean <code>hasFlag()</code> Checks when the image has a flag.
setFlag(Flag flag) Sets the flag.
setFlag(Flag flag, String description) Sets the flag.
removeFlag() Removes the flag.
boolean <code>getFlag(int row, int column)</code> Returns the flag of 1 pixel.
Flag <code>getFlag(int sliceNumber)</code> returns the flag of the slice.
Flag <code>getFlag()</code> Returns the flag of the image.
Image <code>getPreview(float res)</code>

Methods
Returns a preview of the image
Double2d getImageData() Returns the Image data.
double getFrequency() Returns the reference frequency.
double getFrequency(Frequency freq) Returns the reference frequency.
setFrequency(double frequency) Sets the reference frequency.
setFrequency(double frequency, Frequency unit) Sets the reference frequency.

API details

Constructors

<code>SlicedImage()</code>
<p>A constructor which creates a standard SlicedImage. The image is sliced in pieces of 500x500 pixels. Every slice is kept in a ListContext as a SimpleImage. The usage of a SlicedImage should be transparent to the user. Exactly the same methods are available on a SlicedImage as on a SimpleImage.</p> <p>Example</p> <p>Typical example on how to create a SlicedImage.</p> <pre>image=SlicedImage(description="ngc 6992", image=im, flag=flag, errors=er, quantity=quant, wavelength=wavelength, wcs=wcs)</pre>

<code>SlicedImage(SlicedImage copy)</code>
<p>The copy constructor</p> <p>Makes a new SlicedImage from a copy of the given Image.</p> <p>Argument</p> <p>SlicedImage copy [INPUT, MANDATORY]</p>

<code>SlicedImage(String description)</code>
<p>The constructor with a description</p> <p>This constructor also sets the description of the SlicedImage</p> <p>Argument</p> <p>String description [INPUT, MANDATORY]</p>

Methods

<code>copy()</code>
Returns a copy.

copy()
Returns a copy from this SlicedImage.
setSliceSize(int row, int column)
Sets the size of the slices of the image.
Sets the size of the slices of the image. The standard size is 500 rows and 500 columns. At this moment, the size should be set before adding data to the SlicedImage!
Arguments
int row [INPUT, MANDATORY]
int column [INPUT, MANDATORY]
int[] getSliceSize()
Returns the size of the slices of the image.
Returns the size of the slices of the image. The standard size is 500 rows and 500 columns.
Return
int[]
The size of the slices (in rows and columns)
SimpleImage getSlice(int sliceNumber)
Returns the chosen slice.
Returns the chosen slice of the image as a SimpleImage.
Argument
int sliceNumber [INPUT, MANDATORY]
Return
SimpleImage
The chosen slice of the Image.
setImage(AbstractOrdered2dData image, Unit unit)
Sets the image.
Sets the image. You can give a unit. The given unit will automatically also be used for the errors on this image. If the new image has other dimensions than the original image, the errors, exposure and flag are removed. If the new image has the same dimensions, then the old values for the errors, exposures and flag are kept (but the unit of the errors is not). The image data will be automatically sliced.
Arguments
AbstractOrdered2dData image [INPUT, MANDATORY]
Unit unit [INPUT, MANDATORY]
setWavelength(double wavelength)
Sets the reference wavelength.
Set the reference wavelength of the image in microns.

setWavelength(double wavelength)
Argument double wavelength [INPUT, MANDATORY]
setWavelength(double wavelength, Length unit)
Sets the reference wavelength. Set the reference wavelength of the image. Arguments double wavelength [INPUT, MANDATORY] Length unit [INPUT, MANDATORY]
double getWavelength()
Returns the reference wavelength Returns the reference wavelength of the image. Return double The reference wavelength
double getWavelength(Length unit)
Returns the reference wavelength Returns the reference wavelength of the image. Argument Length unit [INPUT, MANDATORY] Return double The reference wavelength
Wcs getWcs()
Returns the World Coordinates System. Returns the World Coordinates System of the image. Return Wcs The World Coordinates System of the image.
setWcs(Wcs wcs)
Sets the world coordinates system. Sets the world coordinates system of the image. Argument Wcs wcs [INPUT, MANDATORY]

int[] getDimensions()

Returns the dimension.

Returns the dimension (width, height) of this SlicedImage.

Return

int[]

The dimension (width, height) of this SlicedImage.

setImage(AbstractOrdered2dData image)

Sets the image.

Sets the image. If the new image has other dimensions than the original image, the errors, exposure and flag are removed. If the new image has the same dimensions, then the old values for the errors, exposure and flag are kept.

Argument

AbstractOrdered2dData **image** [INPUT, MANDATORY]

int getHeight()

Returns the height.

Returns the height of this SimpleImage.

Return

int

The height of this SimpleImage.

int getWidth()

Returns the width

Returns the width of this SlicedImage.

Return

int

The width of this SlicedImage.

Unit getUnit()

Returns the unit.

Returns the unit of the image. The unit of the errors of this image is the same as the unit of the image.

Return

Unit

The unit

setUnit(Unit unit)

Sets the unit.

setUnit(Unit unit)
Sets the unit of the image. Adapting the unit of the image will also adapt the unit of the errors on the image.
Argument
Unit unit [INPUT, MANDATORY]
getIntensity(double row, double column)
Returns the intensity
Returns the intensity of the SlicedImage at a given point. If the pixel is flagged out, NaN is given back.
Arguments
double row [INPUT, MANDATORY]
double column [INPUT, MANDATORY]
double getIntensityWorldCoordinates(double x, double y)
Returns the intensity of the image
Returns the intensity of the image at a given point in world coordinates. If the pixel is flagged out, NaN is given back.
Arguments
double x [INPUT, MANDATORY]
double y [INPUT, MANDATORY]
Return
double
The intensity
Number getPixel(int row, int column)
Returns 1 pixel value.
Returns 1 pixel value of the image.
Arguments
int row [INPUT, MANDATORY]
int column [INPUT, MANDATORY]
Return
Number
1 pixel value of the image.
setIntensity(int row, int column, Number value)
Sets the intensity of the image.
Sets the intensity of the image at a given point.
Arguments
int row [INPUT, MANDATORY]
int column [INPUT, MANDATORY]

setIntensity(int row, int column, [Number](#) value)

[Number](#) value [INPUT, MANDATORY]

AbstractOrdered2dData getImage(int sliceNumber)

returns the image data of the slice as a Numeric2d.

Returns the image data as a Numeric2d containing the data of the slice of the image.

Argument

int **sliceNumber** [INPUT, MANDATORY]

Return

AbstractOrdered2dData

The image data of the slice as a Numeric2d

AbstractOrdered2dData getImage()

returns the image as a Numeric2d.

Returns the image as a Numeric2d containing the data of the image.

Return

AbstractOrdered2dData

The image as a Numeric2d

boolean hasCoverage()

Checks whether the image has an coverage.

Returns true if the image has an coverage.

Return

boolean

True if the coverage is set.

boolean hasError()

Checks whether the image has an error.

Returns true if the image has an error.

Return

boolean

True if the error is set.

setCoverage(AbstractOrdered2dData coverage)

Sets the coverage.

Sets the coverages of the image. The Numeric2d containing the coverages should have the same dimensions as the image. If this is not the case, the coverages will not be adapted.

Argument

AbstractOrdered2dData **coverage** [INPUT, MANDATORY]

setError(AbstractOrdered2dData error)

Sets the error.

Sets the errors of the image. The Numeric2d containing the errors should have the same dimensions as the image. If this is not the case, the errors will not be adapted.

Argument

AbstractOrdered2dData **error** [INPUT, MANDATORY]

removeCoverage()

Removes the coverage.

Removes the coverage, if an coverage exists.

removeError()

Removes the error.

Removes the error, if an error exists.

[Number](#) getCoverage(int row, int column)

Returns the coverage of 1 pixel.

Returns the coverage of 1 pixel.

Arguments

int **row** [INPUT, MANDATORY]

int **column** [INPUT, MANDATORY]

Return

[Number](#)

The coverage of 1 pixel.

AbstractOrdered2dData getCoverage(int sliceNumber)

returns the coverage of the slice as a Numeric2d.

Returns the coverage as a Numeric2d.

Argument

int **sliceNumber** [INPUT, MANDATORY]

Return

AbstractOrdered2dData

The coverage of the slice as a Numeric2d

AbstractOrdered2dData getCoverage()

Returns the coverage of the image as a Numeric2d.

Returns the coverage of the image as a Numeric2d containing the coverage of every pixel.

Return

AbstractOrdered2dData

AbstractOrdered2dData getCoverage()

The coverage

[Number](#) getError(int row, int column)

Returns the error of 1 pixel.

Returns the error of 1 pixel.

Arguments

int **row** [INPUT, MANDATORY]

int **column** [INPUT, MANDATORY]

Return

[Number](#)

The error of 1 pixel.

AbstractOrdered2dData getError(int sliceNumber)

returns the error of the slice as a Numeric2d.

Returns the error as a Numeric2d.

Argument

int **sliceNumber** [INPUT, MANDATORY]

Return

AbstractOrdered2dData

The error of the slice as a Numeric2d

AbstractOrdered2dData getError()

Returns the error of the image as a Numeric2d.

Returns the error of the image as a Numeric2d containing the error of every pixel.

Return

AbstractOrdered2dData

The error

boolean hasExposure()

Checks whether the image has an exposure.

Returns true if the image has an exposure.

Return

boolean

True if the exposure is set.

setExposure(AbstractOrdered2dData exposure)

Sets the exposure.

setExposure(AbstractOrdered2dData exposure)
Sets the exposure of every pixel of the image. The Numeric2d containing the exposure should have the same dimensions as the image. If this is not the case, the exposure will not be adapted.
Argument
AbstractOrdered2dData exposure [INPUT, MANDATORY]
setExposure(AbstractOrdered2dData exposure, String description)
Sets the exposure.
Sets the exposure of every pixel of the image. The Numeric2d containing the exposure should have the same dimensions as the image. If this is not the case, the exposure will not be adapted.
Arguments
AbstractOrdered2dData exposure [INPUT, MANDATORY]
String description [INPUT, MANDATORY]
removeExposure()
Removes the exposure.
Removes the exposure, if an exposure exists.
Number getExposure(int row, int column)
Returns the exposure of 1 pixel.
Returns the exposure of 1 pixel.
Arguments
int row [INPUT, MANDATORY]
int column [INPUT, MANDATORY]
Return
Number
The exposure of 1 pixel.
AbstractOrdered2dData getExposure(int sliceNumber)
returns the exposure of the slice as a Numeric2d.
Returns the exposure as a Numeric2d.
Argument
int sliceNumber [INPUT, MANDATORY]
Return
AbstractOrdered2dData
The exposure of the slice as a Numeric2d
AbstractOrdered2dData getExposure()
Returns the exposure of the image as a Numeric2d.
Returns the exposure of the image as a Numeric2d containing the exposure of every pixel.

AbstractOrdered2dData <code>getExposure()</code>
<p>Return</p> <p>AbstractOrdered2dData</p> <p>The exposure</p>
boolean <code>hasFlag()</code>
<p>Checks when the image has a flag.</p> <p>Returns true if the image has a flag.</p> <p>Return</p> <p>boolean</p> <p>True if the image has a flag.</p>
setFlag(Flag flag)
<p>Sets the flag.</p> <p>Sets the flag of every pixel of the image. The Flag should have the same dimensions as the image. If this is not the case, the Flag will not be adapted.</p> <p>Argument</p> <p>Flag flag [INPUT, MANDATORY]</p>
setFlag(Flag flag, String description)
<p>Sets the flag.</p> <p>Sets the flag of every pixel of the image. The Flag should have the same dimensions as the image. If this is not the case, the Flag will not be adapted. The flag is also described.</p> <p>Arguments</p> <p>Flag flag [INPUT, MANDATORY]</p> <p>String description [INPUT, MANDATORY]</p>
removeFlag()
<p>Removes the flag.</p> <p>Removes the flag, if a flag exists.</p>
boolean <code>getFlag(int row, int column)</code>
<p>Returns the flag of 1 pixel.</p> <p>Returns the flag of 1 pixel.</p> <p>Arguments</p> <p>int row [INPUT, MANDATORY]</p> <p>int column [INPUT, MANDATORY]</p> <p>Return</p> <p>boolean</p> <p>The flag of 1 pixel. True if flagged out.</p>

Flag <code>getFlag(int sliceNumber)</code>
returns the flag of the slice. Returns the flag. Argument int sliceNumber [INPUT, MANDATORY] Return Flag The flag of the slice
Flag <code>getFlag()</code>
Returns the flag of the image. Returns the flag of the image. Return Flag The flag
Image <code>getPreview(float res)</code>
Returns a preview of the image Returns a new SlicedImage, but with a lower resolution if res is smaller than 1. A higher resolution is returned, if res is larger than 1. The spatial regridding is based on linear interpolation which will, in general, not be suitable for scientific purposes. Argument float res [INPUT, MANDATORY] Return Image The preview of the image
Double2d <code>getImageData()</code>
Returns the Image data. Returns the image data as a Double2d. Return Double2d The image data as Double2d
double <code>getFrequency()</code>
Returns the reference frequency. Returns the reference frequency of the image. Return double

double getFrequency()

The reference frequency of the image.

double getFrequency(Frequency freq)
--

Returns the reference frequency.

Returns the reference frequency of the image.

Argument

Frequency freq [INPUT, MANDATORY]
--

Return

double

The reference frequency of the image.

setFrequency(double frequency)

Sets the reference frequency.

Set the reference frequency of the image in gigahertz.
--

Argument

double frequency [INPUT, MANDATORY]
--

setFrequency(double frequency, Frequency unit)

Sets the reference frequency.


Set the reference frequency of the image.

Arguments

double frequency [INPUT, MANDATORY]
--

Frequency unit [INPUT, MANDATORY]
--

2.383. SmoothBaseline

Full Name:	herschel.ia.toolbox.spectrum.standingwaves.SmoothBaseline
Type:	Java Task - 
Import:	from herschel.ia.toolbox.spectrum.standingwaves import SmoothBaseline

Description

Class that makes a smooth and clipped version of the flux.

Example

Example 1: Description
<pre> from herschel.ia.toolbox.spectrum.standingwaves import FitFringeData from herschel.ia.toolbox.spectrum.standingwaves import SmoothBaseline myFreq = Double1d(freq_column.data) myFlux = Double1d(flux_column.data) ffData = FitFringeData(myFreq, myFlux) sm = SmoothBaseline() baseline = sm(data=ffData, midcycle=1700000.0) mask = sm.mask sm2 = SmoothBaseline() u = [(598.0,598.1), (599.0,599.1), (600.0,600.1)] baseline2 = sm2(data=ffData, midcycle=1700000.0, box=200, usermask=u, plot=True) mask2 = sm2.mask </pre>

API Summary

Jython Syntax
<pre> sm = SmoothBaseline() baseline = sm(data=myData, midcycle=1700000.0) mask = f.mask </pre>
Properties
FitFringeData data [INPUT, MANDATORY, default=no default value]
Double midcycle [INPUT, MANDATORY, default=no default value]
Integer box [INPUT, MANDATORY, default=no default value]
Object usermask [INPUT, MANDATORY, default=no default value]
Integer plot [INPUT, MANDATORY, default=no default value]
FitFringeData baseline [OUTPUT, MANDATORY, default=no default value]
Double1d mask [OUTPUT, MANDATORY, default=no default value]
Boolean mhz [INPUT, MANDATORY, default=no default value]

API details

Properties

FitFringeData data [INPUT, MANDATORY, default=no default value]
Input FitFringeData which contains: - 1d array of wavelength, 1d array of flux, 1d array of flag (optional), 1d array of weight (optional).

Double midcycle [INPUT, MANDATORY, default=no default value]

- input mid of fringe search range (per inverse wavenumber in micron).

Integer box [INPUT, MANDATORY, default=no default value]

- input smooth box.

Object usermask [INPUT, MANDATORY, default=no default value]

- Input frequencies of a set of wavelengths to disregard during fringe fitting (eg. at emission lines).

Integer plot [INPUT, MANDATORY, default=no default value]

- Plot the result.

FitFringeData baseline [OUTPUT, MANDATORY, default=no default value]

- Smoothed background.


DoubleId mask [OUTPUT, MANDATORY, default=no default value]

- Output mask.

Boolean mhz [INPUT, MANDATORY, default=no default value]

- Unit in mhz.

2.384. SmoothingTask


Full Name:	herschel.ia.toolbox.image.SmoothingTask
Type:	Java Task - 
Import:	from herschel.ia.toolbox.image import SmoothingTask

Description

An abstract Task for smoothing.

An abstract Task for smoothing images.

2.385. SmoothSpectrum

Full Name:	herschel.ia.toolbox.spectrum.SmoothSpectrum
Type:	Java Task - 
Import:	from herschel.ia.toolbox.spectrum import SmoothSpectrum

Description

Task for smoothing the segments included in a spectrum container according to a selected filter.

The other attributes found in the spectra are not processed but just copied to the result container.

Flags and weights can be included in the processing by setting the 'variant'-parameter accordingly.

API Summary

Properties
SpectrumContainer ds [INPUT, MANDATORY, default=no default value.]
SpectrumContainer result [OUTPUT, MANDATORY, default=no default value.]
String filter [INPUT, OPTIONAL, default=no default value.]
Double width [INPUT, MANDATORY, default=no default value.]
String unit [INPUT, OPTIONAL, default="pixels".]
String variant [INPUT, OPTIONAL, default=no default value.]
String edge [INPUT, OPTIONAL, default="REPEAT".]
Boolean center [INPUT, OPTIONAL, default=True.]
Boolean overwrite [INPUT, OPTIONAL, default=False.]

API details

Properties

SpectrumContainer ds [INPUT, MANDATORY, default=no default value.]
Input container with the spectra to be smoothed.
SpectrumContainer result [OUTPUT, MANDATORY, default=no default value.]
Output container with the smoothed spectra.
String filter [INPUT, OPTIONAL, default=no default value.]
Filter (convolution) to be applied. Available are: "Box" and "Gaussian".
Double width [INPUT, MANDATORY, default=no default value.]
Smoothing width parameter to configure the filter/convolution kernel. The width is expressed in units as specified as the unit parameter.

String `unit` [INPUT, OPTIONAL, default="pixels".]

Specify the unit the width is expressed in. The default is pixels, i.e. specify the width in number of pixels. Other values should be compatible with the unit specified for the wave scale (eg width in GHz and wave scale unit given in MHz is ok but width in "cm" not).

String `variant` [INPUT, OPTIONAL, default=no default value.]

Specify the variant of processing mode you would like to run (including flags / weights / ...). HCSS defaults include: "flux" / "flux-weight" / "flux-flag-weight"

String `edge` [INPUT, OPTIONAL, default="REPEAT".]

Parameter to configure the behavior at the edges: "ZEROES", "CIRCULAR", "REPEAT", "CUT".

Boolean `center` [INPUT, OPTIONAL, default=True.]

Parameter to configure the behavior of the convolution: If set to true the smoothed value is assigned to the center of the smoothing interval. Otherwise, it is assigned at the left edge of the smoothing interval.


Boolean `overwrite` [INPUT, OPTIONAL, default=False.]

Specify whether the input data container can be reused - the values found therein are overwritten.

History

- 2008-03-19 - meli: initial.

2.386. SORT

Full Name:	herschel.ia.numeric.toolbox.basic.Sort
Alias:	SORT
Type:	Java Class - 
Import:	from herschel.ia.numeric.toolbox.basic import Sort

Description

Sorts the specified array into ascending natural order.

Index sorting is available as a special case of SORT i.e. SORT.BY_INDEX which returns an index array. The SORT.IS_SORTED function tests if the given array is sorted.

Examples

Example 1: Apply SORT on a DoubleId

```
x=DoubleId([-1,4,2,7,2,-6,-8,3,2])
print SORT(x) # [-8.0,-6.0,-1.0,2.0,2.0,2.0,3.0,4.0,7.0]
```

Example 2: Apply SORT.BY_INDEX

```
x=DoubleId([-1,4,2,7,2,-6,-8,3,2])
print SORT.BY_INDEX(x) # [6,5,0,2,4,8,7,1,3]
```

Example 3: Apply SORT.IS_SORTED

```
x=DoubleId([-1,4,2,7,2,-6,-8,3,2])
print SORT.IS_SORTED(x) # 0 (false)
```

API Summary

Jython Syntax

```
<y>=SORT(<x>)
```

Properties

[any array **x** \[INPUT, MANDATORY, default=no default value\]](#)

[an array **y** \[OUTPUT, MANDATORY, default=no default value\]](#)

API details

Properties


any array **x [INPUT, MANDATORY, default=no default value]**

The input array can be an array of rank 1

an array **y [OUTPUT, MANDATORY, default=no default value]**

Returns an array with the same number of elements as the input array and sorted in the ascending natural order.

2.387. SourceExtractorDaophotTask

Full Name:	herschel.ia.toolbox.srcext.SourceExtractorDaophotTask
Type:	Java Task - 
Import:	from herschel.ia.toolbox.srcext import SourceExtractorDaophotTask
Category:	task

Description

This task extracts point sources from an HCSS SimpleImage image using DAOPHOT.

The task returns a SourceListProduct listing the sources found in the input image

The DAOPHOT algorithm is described in:

- DAOPHOT: [Stetson \(1987\) "DAOPHOT - A computer program for crowded-field stellar photometry"](#)

Note that the image must have units specified, and these must be $\langle \text{mJy}, \text{Jy} \text{ or } \text{MJy} \rangle / \langle \text{pixel}, \text{sr} \text{ or } \text{beam} \rangle$. Use `print myImage.getUnit()` to view the image units, and `myImage.setUnit(herschel.share.unit.Unit.parse("MJy/sr"))` to set the units of the image to MJy/sr (Jy/beam, Jy/pixel, mJy/pixel etc. also work).

Example

Example 1: Extract sources from a SimpleImage using DAOPHOT

```
# myImage is a SimpleImage
disp = Display(myImage)
#
# Extract a list of sources from the image
beamArea = Math.PI * 18.0 ** 2 -/ 4 -/ LOG(2)
sourceList = sourceExtractorDaophot( \
    image = myImage,          # Image name \
    detThreshold = 10,        # Threshold in signal-to-noise ratio (or log
evidence) \
    fwhm = 18.0,              # FWHM of PRF (arcsec) \
    beamArea = beamArea,     # Area of the beam \
    -)
#
# How many sources found?
print -"Found", len(sourceList), -"sources."
#
# Display each source on the image (or drag and drop the sourceList onto the
image)
disp.addPositionList(sourceList)
```

API Summary

Properties
SimpleImage image [INPUT, MANDATORY, default=no default value]
Double detThreshold [INPUT, MANDATORY, default=no default value]
Double fwhm [INPUT, MANDATORY, default=no default value]
SimpleImage prf [INPUT, OPTIONAL, default=no default value]
Double beamArea [INPUT, OPTIONAL, default=no default value]
Double pixelRegion [INPUT, MANDATORY, default=default value: 1.5]

Properties
SourceListProduct inputSourceList [INPUT, OPTIONAL, default=no default value]
Boolean returnPixelCoordinates [INPUT, OPTIONAL, default=default value: False]
Boolean useSignalToNoise [INPUT, OPTIONAL, default=default value: True]
Double radiusArcsec [INPUT, OPTIONAL, default=default value: 1*FWHM]
Double innerArcsec [INPUT, OPTIONAL, default=default value: 1.25*FWHM]
Double outerArcsec [INPUT, OPTIONAL, default=default value: 3*FWHM]
Double roundnessMin [INPUT, OPTIONAL, default=default value: -1.0]
Double roundnessMax [INPUT, OPTIONAL, default=default value: 1.0]
Double sharpnessMin [INPUT, OPTIONAL, default=default value: 0.2]
Double sharpnessMax [INPUT, OPTIONAL, default=default value: 1.0]
Double corner1Ra [INPUT, OPTIONAL, default=default value: Ra of image pixel (0,0)]
Double corner1Dec [INPUT, OPTIONAL, default=default value: Dec of image pixel (0,0)]
Double corner2Ra [INPUT, OPTIONAL, default=default value: Ra of image pixel (width-1, height-1)]
Double corner2Dec [INPUT, OPTIONAL, default=default value: Dec of image pixel (width-1, height-1)]
Boolean getFilteredMap [INPUT, OPTIONAL, default=default value: False]
Boolean getPrf [INPUT, OPTIONAL, default=default value: False]

API details

Properties

SimpleImage image [INPUT, MANDATORY, default=no default value]
Image map to search for point sources
Double detThreshold [INPUT, MANDATORY, default=no default value]
Threshold at which a local maximum is detected. The meaning of this depends on the useSignalToNoise parameter and may be set to either signal-to-noise or DAOPHOT H-units for DAOPHOT extractor.
Double fwhm [INPUT, MANDATORY, default=no default value]
Full-width half maximum, in arcsec, of a default Gaussian point response function (PRF).
SimpleImage prf [INPUT, OPTIONAL, default=no default value]
SimpleImage of a custom point response function (PRF). If defined, this will be used in preference to the default Gaussian. The PRF should be of odd dimension, with the peak in the centre, and is interpreted as the response of the central pixels of a 1 Jy source in a map with units of the input map.

Double beamArea [INPUT, OPTIONAL, default=no default value]
<p>Solid angle of the beam in square arcsec. For use when the input map has units of Jy/beam. For example, for a Gaussian beam profile with a given full-width half maximum, fwhm, beamArea is given by</p> $\text{Math.PI} * \text{fwhm}^2 / 4 / \text{LOG}(2)$ <p>For PACS and SPIRE, the values given in the Observer's Manuals for the FWHM are:</p> <ul style="list-style-type: none"> • PACS-70: 5.2 • PACS-100: 7.7 • PACS-160: 12 • SPIRE-PSW: 18 • SPIRE-PMW: 25 • SPIRE-PLW: 36

Double pixelRegion [INPUT, MANDATORY, default=default value: 1.5]
Radius around each pixel to consider when searching for local maxima. Units: pixels.

SourceListProduct inputSourceList [INPUT, OPTIONAL, default=no default value]
SourceListProduct containing a list of "known" sourceList positions, at which the fluxes will be extracted.

Boolean returnPixelCoordinates [INPUT, OPTIONAL, default=default value: False]
Return x, y coordinates of sources in ra, dec columns. If image has a valid WCS, FWHM is assumed to be in arcsec; otherwise FWHM is assumed to be in pixels

Boolean useSignalToNoise [INPUT, OPTIONAL, default=default value: True]
Threshold to use. If set to False, the threshold will be DAOPHOT H units. This will be returned as the 'quality' in the SourceListDataset.

Double radiusArcsec [INPUT, OPTIONAL, default=default value: 1*FWHM]
Source aperture radius in arcsec

Double innerArcsec [INPUT, OPTIONAL, default=default value: 1.25*FWHM]
Inner radius of sky aperture annulus in arcsec

Double outerArcsec [INPUT, OPTIONAL, default=default value: 3*FWHM]
Outer radius of sky aperture annulus in arcsec

Double roundnessMin [INPUT, OPTIONAL, default=default value: -1.0]

Minimum value for DAOPHOT roundness filter

Double roundnessMax [INPUT, OPTIONAL, default=default value: 1.0]

Maximum value for DAOPHOT roundness filter

Double sharpnessMin [INPUT, OPTIONAL, default=default value: 0.2]

Minimum value for DAOPHOT sharpness filter

Double sharpnessMax [INPUT, OPTIONAL, default=default value: 1.0]

Maximum value for DAOPHOT sharpness filter

Double corner1Ra [INPUT, OPTIONAL, default=default value: Ra of image pixel (0,0)]

These "corner..." parameters locate two points on the image in world coordinates, defining opposite corners of a rectangle. Only sources within this rectangle will be returned in the output SourceListProduct. If these parameters are not used then the entire map will be searched. Otherwise, if at least one of these parameters are defined, then all four must be.

Double corner1Dec [INPUT, OPTIONAL, default=default value: Dec of image pixel (0,0)]

See description for corner1Ra.

Double corner2Ra [INPUT, OPTIONAL, default=default value: Ra of image pixel (width-1, height-1)]

See description for corner1Ra.

Double corner2Dec [INPUT, OPTIONAL, default=default value: Dec of image pixel (width-1, height-1)]

See description for corner1Ra.


Boolean getFilteredMap [INPUT, OPTIONAL, default=default value: False]

Return the input map filtered by the DAOPHOT kernel. If this is set to True, the return value for the task will be a tuple: (sourceList,)

Boolean getPrf [INPUT, OPTIONAL, default=default value: False]

If this is set to True, the return value for the task will be a tuple: (sourceList,)

2.388. SourceExtractorSussextractorTask

Full Name:	herschel.ia.toolbox.srcext.SourceExtractorSussextractorTask
Type:	Java Task - 
Import:	from herschel.ia.toolbox.srcext import SourceExtractorSussextractorTask
Category:	task

Description

This task extracts point sources from an HCSS SimpleImage image using SUSSExtractor.

The task returns a SourceListProduct listing the sources found in the input image

The SUSSExtractor algorithm is described in:

- SUSSExtractor: [Savage & Oliver \(2007\) Bayesian Methods of Astronomical Source Extraction](#)

Note that the image must have units specified, and these must be $< \text{mJy}, \text{Jy} \text{ or } \text{MJy} > / < \text{pixel}, \text{sr} \text{ or } \text{beam} >$. Use `print myImage.getUnit()` to view the image units, and `myImage.setUnit(herschel.share.unit.Unit.parse("MJy/sr"))` to set the units of the image to MJy/sr (Jy/beam, Jy/pixel, mJy/pixel etc. also work).

Example

Example 1: Extract sources from a SimpleImage using SUSSExtractor

```
# myImage is a SimpleImage
disp = Display(myImage)
#
# Extract a list of sources from the image
beamArea = Math.PI * 18.0 ** 2 -/ 4 -/ LOG(2)
sourceList = sourceExtractorSussextractor( \
    image = myImage,          # Image name \
    detThreshold = 10,       # Threshold in signal-to-noise ratio (or log
evidence) \
    fwhm = 18.0,             # FWHM of PRF (arcsec) \
    beamArea = beamArea,    # Area of the beam \
    -)
#
# How many sources found?
print -"Found", len(sourceList), -"sources."
#
# Display each source on the image (or drag and drop the sourceList onto the
image)
disp.addPositionList(sourceList)
```

API Summary

Properties
SimpleImage image [INPUT, MANDATORY, default=no default value]
Double detThreshold [INPUT, MANDATORY, default=no default value]
Double fwhm [INPUT, MANDATORY, default=no default value]
SimpleImage prf [INPUT, OPTIONAL, default=no default value]
Double beamArea [INPUT, OPTIONAL, default=no default value]
Double fluxPriorsLambda [INPUT, OPTIONAL, default=default value: 1.0]

Properties
Boolean fitBackground [INPUT, OPTIONAL, default=default value: False]
Double pixelRegion [INPUT, MANDATORY, default=default value: 1.5]
SourceListProduct inputSourceList [INPUT, OPTIONAL, default=no default value]
Boolean returnPixelCoordinates [INPUT, OPTIONAL, default=default value: False]
Boolean useSignalToNoise [INPUT, OPTIONAL, default=default value: True]
Double fluxPriorsMin [INPUT, OPTIONAL, default=default value: 1.0e-4]
Double fluxPriorsMax [INPUT, OPTIONAL, default=default value: 1.0e8]
Double backgroundPriorsMin [INPUT, OPTIONAL, default=no default value]
Double backgroundPriorsMax [INPUT, OPTIONAL, default=no default value]
Boolean subtractMedianBackground [INPUT, OPTIONAL, default=default value: False]
Double corner1Ra [INPUT, OPTIONAL, default=default value: Ra of image pixel (0,0)]
Double corner1Dec [INPUT, OPTIONAL, default=default value: Dec of image pixel (0,0)]
Double corner2Ra [INPUT, OPTIONAL, default=default value: Ra of image pixel (width-1, height-1)]
Double corner2Dec [INPUT, OPTIONAL, default=default value: Dec of image pixel (width-1, height-1)]
Boolean getFilteredMap [INPUT, OPTIONAL, default=default value: False]
Boolean getPrf [INPUT, OPTIONAL, default=default value: False]

API details

Properties

SimpleImage image [INPUT, MANDATORY, default=no default value]
Image map to search for point sources
Double detThreshold [INPUT, MANDATORY, default=no default value]
Threshold at which a local maximum is detected. The meaning of this depends on the useSignalToNoise parameter and may be set to either signal-to-noise or log(evidence).
Double fwhm [INPUT, MANDATORY, default=no default value]
Full-width half maximum, in arcsec, of a default Gaussian point response function (PRF).
SimpleImage prf [INPUT, OPTIONAL, default=no default value]
SimpleImage of a custom point response function (PRF). If defined, this will be used in preference to the default Gaussian. The PRF should be of odd dimension, with the peak in the centre,

SimpleImage prf [INPUT, OPTIONAL, default=no default value]

and is interpreted as the response of the central pixels of a 1 Jy source in a map with units of the input map.

Double beamArea [INPUT, OPTIONAL, default=no default value]

Solid angle of the beam in square arcsec. For use when the input map has units of Jy/beam. For example, for a Gaussian beam profile with a given full-width half maximum, fwhm, beamArea is given by

$$\text{Math.PI} * \text{fwhm}^2 / 4 / \text{LOG}(2)$$

For PACS and SPIRE, the values given in the Observer's Manuals for the FWHM are:

- PACS-70: 5.2
- PACS-100: 7.7
- PACS-160: 12
- SPIRE-PSW: 18
- SPIRE-PMW: 25
- SPIRE-PLW: 36

Double fluxPriorsLambda [INPUT, OPTIONAL, default=default value: 1.0]

Lambda value for SUSSExtractor flux priors: 0 = tophat, 1 = Jeffreys, >1 = deboost

Boolean fitBackground [INPUT, OPTIONAL, default=default value: False]

SUSSExtractor flag to fit background as a free parameter

Double pixelRegion [INPUT, MANDATORY, default=default value: 1.5]

Radius around each pixel to consider when searching for local maxima. Units: pixels.

SourceListProduct inputSourceList [INPUT, OPTIONAL, default=no default value]

SourceListProduct containing a list of "known" source positions, at which the fluxes will be extracted.

Boolean returnPixelCoordinates [INPUT, OPTIONAL, default=default value: False]


Return x, y coordinates of sources in ra, dec columns. If image has a valid WCS, FWHM is assumed to be in arcsec; otherwise FWHM is assumed to be in pixels

Boolean useSignalToNoise [INPUT, OPTIONAL, default=default value: True]

Threshold to use. If set to False, the threshold will be $\log(\text{evidence})$. This will be returned as the 'quality' in the SourceListDataset.

Double fluxPriorsMin [INPUT, OPTIONAL, default=default value: 1.0e-4]
Prior on minimum flux for SUSSEXtractor, in mJy
Double fluxPriorsMax [INPUT, OPTIONAL, default=default value: 1.0e8]
Prior on maximum flux for SUSSEXtractor, in mJy
Double backgroundPriorsMin [INPUT, OPTIONAL, default=no default value]
Prior on minimum background level for SUSSEXtractor, in units of the input map
Double backgroundPriorsMax [INPUT, OPTIONAL, default=no default value]
Prior on maximum background level for SUSSEXtractor, in units of the input map
Boolean subtractMedianBackground [INPUT, OPTIONAL, default=default value: False]
SUSSEXtractor flag to subtract median background from image
Double corner1Ra [INPUT, OPTIONAL, default=default value: Ra of image pixel (0,0)]
These "corner..." parameters locate two points on the image in world coordinates, defining opposite corners of a rectangle. Only sources within this rectangle will be returned in the output SourceListProduct. If these parameters are not used then the entire map will be searched. Otherwise, if at least one of these parameters are defined, then all four must be.
Double corner1Dec [INPUT, OPTIONAL, default=default value: Dec of image pixel (0,0)]
See description for corner1Ra.
Double corner2Ra [INPUT, OPTIONAL, default=default value: Ra of image pixel (width-1, height-1)]
See description for corner1Ra.
Double corner2Dec [INPUT, OPTIONAL, default=default value: Dec of image pixel (width-1, height-1)]
See description for corner1Ra.
Boolean getFilteredMap [INPUT, OPTIONAL, default=default value: False]
Return the input map filtered by the PRF, in units of mJy. If this is set to True, the return value for the task will be a tuple: (sourceList,). The filtered map is $\sum(d_i P_i / \sigma_i^2) / \sum(P_i^2 / \sigma_i^2)$, with the error map as $1 / \sqrt{\sum(P_i^2 / \sigma_i^2)}$, where the sum is over the PRF pixels, d_i is the data value, P_i is the PRF value, and σ_i is the image uncertainty.
Boolean getPrf [INPUT, OPTIONAL, default=default value: False]
If this is set to True, the return value for the task will be a tuple: (sourceList,)

2.389. SourceFitTask

Full Name:	herschel.ia.toolbox.fit.SourceFitTask
Alias:	SourceFitTask
Type:	Java Task - 
Import:	from herschel.ia.toolbox.fit import SourceFitTask
Category:	generic task

Description

generic module: SourceFitTask

Task shell around the package herschel.ia.numeric.toolbox.fit. Both a command line and a GUI interface are provided. With the GUI interface there is more of the Fitter package accessible than on this tasks command line. However not everything possible within the fitter package is accessible via this task, GUI or otherwise. Most text fields in the GUI accept either numerics or JIDE variables or JIDE expressions.

Example

Example 1: SourceFitTask

```
from herschel.hifi.generic.task import SourceFitTask
# Assume that `tt` and/or `data` are DoubleIeld's
# usage on command line:
ft = SourceFitTask()( x=tt, y=data -)
ft.fitter = -"LevenbergMarquardtFitter"
ft() # performs the execute method
print ft.result # parameters of the fit
print ft.stdev # standard deviations
print ft.fitter # name of the fitter
# Etcetera, etc.
# use the GUI.
ft = SourceFitTask()
ft.gui = 1 # start the GUI
# from the GUI select the data, model, fitter, properties etc and run.
print ft.result # parameters of the fit
print ft.stdev # standard deviations
print ft.fitter # name of the fitter
# Etcetera as before.
```

API Summary

Jython Syntax

see example below

Properties

[NumericData](#) **x** [INPUT, OPTIONAL, default=null]

[DoubleArray](#) **y** [INPUT, MANDATORY, default=no]

[Boolean](#) **indexview** [INPUT, OPTIONAL, default=true]

[DoubleArray](#) **weights** [INPUT, OPTIONAL, default=null]

[AbstractModel](#) **model** [INPUT, OPTIONAL, default=null]

[String](#) **modelname** [INPUT, OPTIONAL, default=no]

[ArrayIeldData](#) **modelarg** [INPUT, OPTIONAL, default=null]

Properties
Fitter fitter [INPUT, OPTIONAL, default=null]
String fittername [INPUT, true, default=no]
String autoinit [INPUT, OPTIONAL, default=""]
Integer smooth [INPUT, OPTIONAL, default=1]
DoubleId result [OUTPUT, OPTIONAL, default=n/a]
DoubleId parameters [OUTPUT, OPTIONAL, default=n/a]
Double chisq [OUTPUT, OPTIONAL, default=n/a]
DoubleId prior [INPUT, OPTIONAL, default=null]
Double scale [OUTPUT, OPTIONAL, default=n/a]
IterationPlotable plotter [INPUT, OPTIONAL, default=null]
Integer plotfreq [INPUT, OPTIONAL, default=10]
Integer printfreq [INPUT, OPTIONAL, default=0]
Boolean auto [INPUT, OPTIONAL, default=false]
Double fixed [INPUT, OPTIONAL, default=1.0]
Double mixed [INPUT, OPTIONAL, default=0.0]
Double tolerance [INPUT, OPTIONAL, default=0.01]
Integer iterations [INPUT, OPTIONAL, default=10000]
Double temperature [INPUT, OPTIONAL, default=0.0]
Double cooling [INPUT, OPTIONAL, default=0.95]
Integer tempsteps [INPUT, OPTIONAL, default=100]
DoubleId initialpars [INPUT, OPTIONAL, default=from model]
DoubleId highlimits [INPUT, OPTIONAL, default=null]
DoubleId lowlimits [INPUT, OPTIONAL, default=null]
IntId keepfixed [INPUT, OPTIONAL, default=null]
DoubleId fixedvalues [INPUT, OPTIONAL, default=null]
Boolean gui [INPUT, OPTIONAL, default=false]

Limitations

Not everything which is possible using the package directly is accessible via this task.

Miscellaneous

In case of problems look at the [trouble shooting](#) section.

API details

Properties

NumericData x [INPUT, OPTIONAL, default=null]
The independent variable(s) of the fit problem. If not provided, it will use indices ranging from 0 to len(y). If y is more-dimensional the proper 2-d indices are provided.
DoubleArray y [INPUT, MANDATORY, default=no]
The data to be fitted. It can be more-dimensional. E.g. a 2-dimensional map.


Boolean <code>indexview</code> [INPUT, OPTIONAL, default=true]
Applicable parameters appear in the same order as in the <code>y</code> DoubleArray This parameter is only active when <code>y</code> has 2 dimensions or more.
DoubleArray <code>weights</code> [INPUT, OPTIONAL, default=null]
The weights to be used in the fit. Can be more-dimensional. When provided it should be the same size as parameter <code>y</code> .
AbstractModel <code>model</code> [INPUT, OPTIONAL, default=null]
The model to be fitted. Default: GaussModel (for 1d data) or Gauss2DModel (for 2d data)
String <code>modelname</code> [INPUT, OPTIONAL, default=no]
The model to be fitted. Default: GaussModel (for 1d data) or Gauss2DModel (for 2d data)
Array1dData <code>modelarg</code> [INPUT, OPTIONAL, default=null]
Arguments, if any, needed in the Constructor of the model.
Fitter <code>fitter</code> [INPUT, OPTIONAL, default=null]
Fitter to be used. Default: LevenbergMarquardtFitter
String <code>fittername</code> [INPUT, true, default=no]
Fitter to be used. Default: LevenbergMarquardtFitter
String <code>autoinit</code> [INPUT, OPTIONAL, default=""]
Automated search for initial params. options: "high", "low", "center"
Integer <code>smooth</code> [INPUT, OPTIONAL, default=1]
Smooth data with BoxCarFilter before auto-search. <code>smooth</code> > 1.
Double1d <code>result</code> [OUTPUT, OPTIONAL, default=n/a]
The parameters of the fit.
Double1d <code>parameters</code> [OUTPUT, OPTIONAL, default=n/a]
The parameters of the fit. (same as <code>result</code>)
Double <code>chisq</code> [OUTPUT, OPTIONAL, default=n/a]
Chi-squared of the fit.
Double1d <code>prior</code> [INPUT, OPTIONAL, default=null]
Prior ranges for the parameters, needed when the evidence is requested.
Double <code>scale</code> [OUTPUT, OPTIONAL, default=n/a]
Noise scale of the data (when auto or mixed is selected)
IterationPlotable <code>plotter</code> [INPUT, OPTIONAL, default=null]
make a plot of the fit at each "plotfreq" iteration. A plotter is provided at <code>herschel.ia.toolbox.fit.IterationPlotter</code>
Integer <code>plotfreq</code> [INPUT, OPTIONAL, default=10]
to be used in conjunction with <code>plotter</code> .

Integer <code>printfreq</code> [INPUT, OPTIONAL, default=0]
Report about the present parameter settings every printfreq-th iteration.
Boolean <code>auto</code> [INPUT, OPTIONAL, default=false]
Select automatic noise scaling.
Double <code>fixed</code> [INPUT, OPTIONAL, default=1.0]
Fixed noise scale.
Double <code>mixed</code> [INPUT, OPTIONAL, default=0.0]
Automatic noise scaling with a minimum.
Double <code>tolerance</code> [INPUT, OPTIONAL, default=0.01]
Stopping criterion for iterative fitting.
Integer <code>iterations</code> [INPUT, OPTIONAL, default=10000]
Maximum number of iterations.
Double <code>temperature</code> [INPUT, OPTIONAL, default=0.0]
Starting temperature in annealing amoeba fitting.
Double <code>cooling</code> [INPUT, OPTIONAL, default=0.95]
Cooling step in annealing amoeba fitting.
Integer <code>tempsteps</code> [INPUT, OPTIONAL, default=100]
Number of exploratory steps at each temperature.
DoubleId <code>initialpars</code> [INPUT, OPTIONAL, default=from model]
Initial parameters in iterative fitters.
DoubleId <code>highlimits</code> [INPUT, OPTIONAL, default=null]
Upper limits of the parameters (only in AmoebaFitter)
DoubleId <code>lowlimits</code> [INPUT, OPTIONAL, default=null]
Lower limits of the parameters (only in AmoebaFitter)
IntId <code>keepfixed</code> [INPUT, OPTIONAL, default=null]
Keep these parameters fixed. (Parameter numbering starts at 0)
DoubleId <code>fixedvalues</code> [INPUT, OPTIONAL, default=null]
Values at which the parameters should be kept.
Boolean <code>gui</code> [INPUT, OPTIONAL, default=false]
Allows to handle this task using a gui.

History

- 15-10-2006 DK

2.390. SourceFittingExplorer

Full Name:	herschel.ia.gui.image.SourceFittingExplorer
Type:	Java Class - 
Import:	from herschel.ia.gui.image import SourceFittingExplorer

Description

An explorer for SourceFittingProducts.

An explorer for SourceFittingProducts.

API Summary

Constructors
SourceFittingExplorer() The construction of a new SourceFittingExplorer.
SourceFittingExplorer(Object object) The construction of a new SourceFittingExplorer associated with the given object.

Methods
String getName() Returns the name.
String getDescription() Returns the description.
boolean canHandle(Class className) Checks whether this SourceFittingExplorer can handle objects of the given class.
setObject(Object object) Sets the object.
SourceFittingProduct getObject() Returns the object.
Class getVariableType() Returns the expected variable type for this SoureFittingExplorer.
JComponent getComponent() Returns the component that is responsible for displaying the data object.
JScrollPane getParameterTable() Returns the parameter table.
addDataObjectListener(DataObjectListener listener) Adds the given listener.
removeDataObjectListener(DataObjectListener listener) Removes the given listener.

API details

Constructors

SourceFittingExplorer()
The construction of a new SourceFittingExplorer. The construction of a new SourceFittingExplorer.
SourceFittingExplorer(Object object)
The construction of a new SourceFittingExplorer associated with the given object. The construction of a new SourceFittingExplorer associated with the given object. Argument <code>Object object</code> [INPUT, MANDATORY]

Methods

String getName()
Returns the name. Returns the name for this SourceFittingExplorer. Return <code>String</code> Returns the name for this SourceFittingExplorer.
String getDescription()
Returns the description. Returns the description for this SourceFittingExplorer. Return <code>String</code> Returns the description for this SourceFittingExplorer.
boolean canHandle(Class className)
Checks whether this SourceFittingExplorer can handle objects of the given class. Returns true if this SourceFittingExplorer can handle objects of the given class; false otherwise. Argument <code>Class className</code> [INPUT, MANDATORY] Return <code>boolean</code> Returns true of this SourceFittingExplorer can handle objects of the given class; false otherwise.

setObject(Object object)

Sets the object.

Sets the object for this SourceFittingExplorer to the given object.

Argument

[Object](#) **object** [INPUT, MANDATORY]

SourceFittingProduct getObject()

Returns the object.

Returns the object for this SourceFittingExplorer.

Return

[SourceFittingProduct](#)

Returns the object for this SourceFittingExplorer.

Class getVariableType()

Returns the expected variable type for this SourceFittingExplorer.

Returns the expected variable type for this SourceFittingExplorer.

Return

[Class](#)

Returns the expected variable type for this SourceFittingExplorer.

JComponent getComponent()

Returns the component that is responsible for displaying the data object.

Returns the component that is responsible for displaying the data object for this SourceFittingExplorer.

Return

[JComponent](#)

Returns the component that is responsible for displaying the data object for this SourceFittingExplorer.

JScrollPane getParameterTable()

Returns the parameter table.

Returns the parameter table for this SourceFittingExplorer.

Return

[JScrollPane](#)

Returns the parameter table for this SourceFittingExplorer.

addDataObjectListener(DataObjectListener listener)

Adds the given listener.

addDataObjectListener(DataObjectListener listener)

Adds the given listener to this SourceFittingExplorer to receive data object events from it.

Argument

DataObjectListener **listener** [INPUT, MANDATORY]

removeDataObjectListener(DataObjectListener listener)


Removes the given listener.

Removes the given listener for this SourceFittingExplorer, so that it no longer receives data object events by this explorer.

Argument

DataObjectListener **listener** [INPUT, MANDATORY]

2.391. SourceFittingPanel

Full Name:	herschel.ia.toolbox.image.gui.SourceFittingPanel
Type:	Java Class - 
Import:	from herschel.ia.toolbox.image.gui import SourceFittingPanel

Description

A panel for the SourceFittingTask.

A panel to serve as GUI for the SourceFittingTask.

API Summary

Constructor	
SourceFittingPanel()	The constructor of a new SourceFittingPanel.
Methods	
JPanel getImagePanel()	Returns a display of the image.
JPanel getButtonPanel()	Returns the button panel.
Action getRunAction()	Returns the run action for this SourceFittingPanel.
Action getResetAction()	Returns the reset action for this SourceFittingPanel.
setSiteEventHandler(SiteEventHandler handler)	Sets the site event handler.
setTask(TaskApi task)	Associates the task.
setVariableSelection(VariableSelection selection)	Sets the variable selection.
Display getDisplay()	Returns the display of the image.
Image getImage()	Returns the associated image.
TaskApi getTask()	Returns the associated task.
Map getMap()	Returns the associated map.
SiteEventHandler getHandler()	Returns the associated site event handler
updateFigure()	Updates the rectangle.

Methods
setPoint(MouseEvent me) Stores the location of the given mouse event in pixel coordinates.
setPoint(Double point) Stores the given location in pixel coordinates.

API details

Constructor

<code>SourceFittingPanel()</code>
The constructor of a new SourceFittingPanel.
The constructor of a new SourceFittingPanel.

Methods

JPanel <code>getImagePanel()</code>
Returns a display of the image.
Returns a panel that displays the image for this SourceFittingPanel.
Return
JPanel
Returns a panel that displays the image for this SourceFittingPanel.

JPanel <code>getButtonPanel()</code>
Returns the button panel.
Returns the button panel for this SourceFittingPanel.
Return
JPanel
Returns the button panel for this SourceFittingPanel.


Action <code>getRunAction()</code>
Returns the run action for this SourceFittingPanel.
Returns the run action for this SourceFittingPanel.
Return
Action
Returns the run action for this SourceFittingPanel.

Action <code>getResetAction()</code>
Returns the reset action for this SourceFittingPanel.
Returns the reset action for this SourceFittingPanel.

Action <code>getResetAction()</code>
<p>Return</p> <p>Action</p> <p>Returns the reset action for this SourceFittingPanel.</p>
setSiteEventHandler(SiteEventHandler handler)
<p>Sets the site event handler.</p> <p>Sets the site event handler for this SourceFittingPanel to the given site event handler.</p> <p>Argument</p> <p>SiteEventHandler handler [INPUT, MANDATORY]</p>
setTask(TaskApi task)
<p>Associates the task.</p> <p>Associates the given task with this SourceFittingPanel.</p> <p>Argument</p> <p>TaskApi task [INPUT, MANDATORY]</p>
setVariableSelection(VariableSelection selection)
<p>Sets the variable selection.</p> <p>Sets the variable selection for this SourceFittingPanel to the given variable selection.</p> <p>Argument</p> <p>VariableSelection selection [INPUT, MANDATORY]</p>
Display <code>getDisplay()</code>
<p>Returns the display of the image.</p> <p>Returns the display for this SourceFittingPanel.</p> <p>Return</p> <p>Display</p> <p>Returns the display for this SourceFittingPanel.</p>
Image <code>getImage()</code>
<p>Returns the associated image.</p> <p>Returns the image associated with this SourceFittingPanel.</p> <p>Return</p> <p>Image</p> <p>Returns the image associated with this SourceFittingPanel.</p>
TaskApi <code>getTask()</code>
<p>Returns the associated task.</p>

TaskApi <code>getTask()</code>
Returns the task associated with this SourceFittingPanel
Return
TaskApi
Returns the task associated with this SourceFittingPanel.
Map <code>getMap()</code>
Returns the associated map.
Returns the map associated with this SourceFittingPanel.
Return
Map
Returns the map associated with this SourceFittingPanel.
SiteEventHandler <code>getHandler()</code>
Returns the associated site event handler
Returns the site event handler associated with this SourceFittingPanel.
Return
SiteEventHandler
Returns the site event handler associated with this SourceFittingPanel.
updateFigure()
Updates the rectangle.
Updates the rectangle associated with this SourceFittingPanel.
setPoint (MouseEvent <code>me</code>)
Stores the location of the given mouse event in pixel coordinates.
Sets the position of the given mouse event at the appropriate place (i.e. at the index that corresponds with the number of times there was clicked before) in the list of clicked points for this SourceFittingPanel in pixel coordinates.
Argument
MouseEvent <code>me</code> [INPUT, MANDATORY]
setPoint (Double <code>point</code>)
Stores the given location in pixel coordinates.
Sets the given screen coordinates at the appropriate place (i.e. at the index that corresponds with the number of times there was clicked before) in the list of clicked points for this SourceFittingPanel in pixel coordinates.
Argument
Double <code>point</code> [INPUT, MANDATORY]

2.392. SourceFittingProduct

Full Name:	herschel.ia.dataset.image.SourceFittingProduct
Type:	Java Class - 
Import:	from herschel.ia.dataset.image import SourceFittingProduct

Description

A class to handle the result of the SourceFittingTask.

A class to handle the result of the SourceFittingTask.

API Summary

Constructors
SourceFittingProduct() The construction of a new SourceFittingProduct.
SourceFittingProduct(double peak, double centerX, double centerY, double sigmaPixels, double background, Wcs wcs, Unit unit) Construction of a new SourceFittingProduct.
SourceFittingProduct(double peak, double centerX, double centerY, double sigmaXPixels, double sigmaYPixels, double rotationAngle, double background, Wcs wcs, Unit unit) Construction of a new SourceFittingProduct.

Methods
double getPeak() Returns the peak intensity for this SourceFittingProduct.
double getCenterX() Returns the x-pixel-coordinate for the center for this SourceFittingProduct.
double getCenterY() Returns the y-pixel-coordinate for the center for this SourceFittingProduct.
double getCenterRA() Returns the right ascension for the center for this SourceFittingProduct.
double getCenterDec() Returns the declination for the center for this SourceFittingProduct.
double getSigmaPixels() Returns the width for this SourceFittingProduct in pixels.
double getSigmaXPixels() Returns the width in the x-direction for this SourceFittingProduct in pixels.
double getSigmaYPixels() Returns the width in the y-direction for this SourceFittingProduct in pixels.
double getSigmaArcsec() Returns the width for this SourceFittingProduct in arcsec.

Methods
double getSigmaXArcsec() Returns the width in the x-direction for this SourceFittingProduct in arcsec.
double getSigmaYArcsec() Returns the width in the y-direction for this SourceFittingProduct in arcsec.
double getRotationAngle() Returns the rotation angle for this SourceFittingProduct in radians.
double getRotationAngleDegrees() Returns the rotation angle for this SourceFittingProduct in degrees.
double getRotationAngleRadians() Returns the rotation angle for this SourceFittingProduct in radians.
double getPositionAngleDegrees() Returns the position angle for this SourceFittingProduct in degrees.
double getPositionAngleRadians() Returns the position angle for this SourceFittingProduct in radians.
double getBackground() Returns the background intensity for this SourceFittingProduct.
String getUnit() Returns the unit of the image for which the fit was made.
setUnit(Unit unit) Sets the unit for this SourceFittingProduct.

API details

Constructors

<code>SourceFittingProduct()</code>
The construction of a new SourceFittingProduct.
The construction of a new SourceFittingProduct.

<code>SourceFittingProduct(double peak, double centerX, double centerY, double sigmaPixels, double background, Wcs wcs, Unit unit)</code>
Construction of a new SourceFittingProduct.
Construction of a new SourceFittingProduct with the given peak intensity, pixel coordinates of the center, width and background.
Arguments
double peak [INPUT, MANDATORY]
double centerX [INPUT, MANDATORY]
double centerY [INPUT, MANDATORY]
double sigmaPixels [INPUT, MANDATORY]
double background [INPUT, MANDATORY]
Wcs wcs [INPUT, MANDATORY]
Unit unit [INPUT, MANDATORY]

```
SourceFittingProduct(double peak, double centerX, double centerY,
double sigmaXPixels, double sigmaYPixels, double rotationAngle,
double background, Wcs wcs, Unit unit)
```

Construction of a new SourceFittingProduct.

Construction of a new SourceFittingProduct with the given peak intensity, pixel coordinates of the center, rotation angle, widths and background.

Arguments

```
double peak [INPUT, MANDATORY]
double centerX [INPUT, MANDATORY]
double centerY [INPUT, MANDATORY]
double sigmaXPixels [INPUT, MANDATORY]
double sigmaYPixels [INPUT, MANDATORY]
double rotationAngle [INPUT, MANDATORY]
double background [INPUT, MANDATORY]
Wcs wcs [INPUT, MANDATORY]
Unit unit [INPUT, MANDATORY]
```

Methods

```
double getPeak()
```

Returns the peak intensity for this SourceFittingProduct.

Returns the value for the peak parameter.

Return

double

Returns the peak intensity for this SourceFittingProduct.

```
double getCenterX()
```

Returns the x-pixel-coordinate for the center for this SourceFittingProduct.

Returns the value for the centerX parameter.

Return

double

Returns the x-pixel-coordinate for the center for this SourceFittingProduct.

```
double getCenterY()
```

Returns the y-pixel-coordinate for the center for this SourceFittingProduct.

Returns the value of the centerY parameter.

Return

double

Returns the y-pixel-coordinate for the center for this SourceFittingProduct.

double getCenterRA()

Returns the right ascension for the center for this SourceFittingProduct.

Returns the value of the centerRA parameter.

Return

double

Returns the right ascension for the center for this SourceFittingProduct.

double getCenterDec()

Returns the declination for the center for this SourceFittingProduct.

Returns the value of the centerDec parameter.

Return

double

Returns the declination for the center for this SourceFittingProduct.

double getSigmaPixels()

Returns the width for this SourceFittingProduct in pixels.

Returns the value of the sigmaPixels parameter.

Return

double

Returns the width for this SourceFittingProduct in pixels.

double getSigmaXPixels()

Returns the width in the x-direction for this SourceFittingProduct in pixels.

Returns the value of the sigmaXPixels parameter.

Return

double

Returns the width in the x-direction for this SourceFittingProduct in pixels.

double getSigmaYPixels()

Returns the width in the y-direction for this SourceFittingProduct in pixels.

Returns the value of the sigmaYPixels parameter.

Return

double

Returns the width in the y-direction for this SourceFittingProduct in pixels.

double getSigmaArcsec()

Returns the width for this SourceFittingProduct in arcsec.

```
double getSigmaArcsec()
```

Returns the value of the sigmaArcsec parameter.

Return

double

Returns the width for this SourceFittingProduct in arcsec.

```
double getSigmaXArcsec()
```

Returns the width in the x-direction for this SourceFittingProduct in arcsec.

Returns the value of the sigmaXArcsec parameter.

Return

double

Returns the width in the x-direction for this SourceFittingProduct in arcsec.

```
double getSigmaYArcsec()
```

Returns the width in the y-direction for this SourceFittingProduct in arcsec.

Returns the value of the sigmaYArcsec parameter.

Return

double

Returns the width in the y-direction for this SourceFittingProduct in arcsec.

```
double getRotationAngle()
```

Returns the rotation angle for this SourceFittingProduct in radians.

Returns the value for the "rotationAngle" parameter.

Return

double

Returns the rotation angle for this SourceFittingProduct in radians.

```
double getRotationAngleDegrees()
```

Returns the rotation angle for this SourceFittingProduct in degrees.

Returns the rotation angle for this SourceFittingProduct in degrees.

Return

double

Returns the rotation angle for this SourceFittingProduct in degrees.


```
double getRotationAngleRadians()
```

Returns the rotation angle for this SourceFittingProduct in radians.

Returns the rotation angle for this SourceFittingProduct in radians.

double <code>getRotationAngleRadians()</code>
Return double Returns the rotation angle for this SourceFittingProduct in radians.
double <code>getPositionAngleDegrees()</code>
Returns the position angle for this SourceFittingProduct in degrees. Returns the position angle for this SourceFittingProduct in degrees. Return double Returns the position angle for this SourceFittingProduct in degrees.
double <code>getPositionAngleRadians()</code>
Returns the position angle for this SourceFittingProduct in radians. Returns the position angle for this SourceFittingProduct in radians. Return double Returns the position angle for this SourceFittingProduct in radians.
double <code>getBackground()</code>
Returns the background intensity for this SourceFittingProduct. Returns the value of the background parameter. Return double Returns the background intensity for this SourceFittingProduct.
String <code>getUnit()</code>
Returns the unit of the image for which the fit was made. Returns the unit of the image for which the fit was made for this SourceFittingProduct. Return String Returns the unit of the image for which the fit was made for this SourceFittingProduct.
setUnit(Unit unit)
Sets the unit for this SourceFittingProduct. Sets the unit for this SourceFittingProduct. Argument Unit unit [INPUT, MANDATORY]

2.393. SourceFittingTask

Full Name:	herschel.ia.toolbox.image.SourceFittingTask
Type:	Java Task - 
Import:	from herschel.ia.toolbox.image import SourceFittingTask

Description

A Task to fit a two-dimensional gaussian to a source in a specified region on an image.

A Task to fit a two-dimensional gaussian combined with a constant to a source in a specified region on an image. The user can assume a circular source or an elongated source.

API Summary

Properties
Image image [INPUT, MANDATORY, default=No default value]
Double minX [INPUT, MANDATORY, default=Default value : Double.NaN]
Double minY [INPUT, MANDATORY, default=Default value : Double.NaN]
Double width [INPUT, MANDATORY, default=Default value : 0.0]
Double height [INPUT, MANDATORY, default=Default value : 0.0]
Boolean elongated [INPUT, OPTIONAL, default=Default value : true]
Boolean slope [INPUT, OPTIONAL, default=Default value : 0.0]
SourceFittingProduct parameters [OUTPUT, MANDATORY, default=No default value]

Miscellaneous

In order to improve the convergence of the fitter, upscaling of the data has been provided. If the intensity peak in the cropped image is below 1000.0, the image is multiply such that the new peak is at 1000.0.

API details

Properties

Image image [INPUT, MANDATORY, default=No default value]
The input image.
Double minX [INPUT, MANDATORY, default=Default value : Double.NaN]
The x-pixel-coordinate of the lower left corner of the rectangle.
Double minY [INPUT, MANDATORY, default=Default value : Double.NaN]
The y-pixel-coordinate of the lower left corner of the rectangle.
Double width [INPUT, MANDATORY, default=Default value : 0.0]
The width of the rectangle in pixels.

Double height [INPUT, MANDATORY, default=Default value : 0.0]

The height of the rectangle in pixels.

Boolean elongated [INPUT, OPTIONAL, default=Default value : true]

Indicates whether the source is elongated (or circular).


Boolean slope [INPUT, OPTIONAL, default=Default value : 0.0]

Indicates whether the background has a slope (or is constant).

SourceFittingProduct parameters [OUTPUT, MANDATORY, default=No default value]

The fitting parameters.


2.394. SpectralProjectionAlgorithm

Full Name:	herschel.ia.toolbox.spectrum.projection.SpectralProjectionAlgorithm
Type:	Java Class - 
Import:	from herschel.ia.toolbox.spectrum.projection import SpectralProjectionAlgorithm

History

- 2009-11-22 - DS: [HCSS SCR-8470] Added methods to propagate units and metadata. Changed Column parameters to Double1d.

2.395. Spectrum1d

Full Name:	herschel.ia.dataset.spectrum.Spectrum1d
Type:	Java Class - 
Import:	from herschel.ia.dataset.spectrum import Spectrum1d
Category:	Datasets

Description

Spectrum1d is an extension of AbstractSpectrumDataset which in turn

is an extension of (Strict)TableDataset. It is a 1-dimensional incarnation of a SpectrumDataset.

Spectrum1d implements an iterator over spectral segments when these are defined.

Within a Spectrum1d there are provisions for 4 (predefined) columns.

1. flux, a Double1d array. This flux column is more or less obligatory, otherwise there can't be much talk of a spectrum. However see below for a caveat.
2. weight. A Double1d array of the same dimensionality as flux.
 - weight = 0, means that the corresponding samples are irrelevant.
 - weight = 1 / stdev², if such a value is available for the fluxes.
3. flag, an Int1d array of the same size as flux. The definition of flags is of course problem dependent. It is suggested to store the meaning of the flags in MetaData.
4. segment, an Int1d array of the same size as flux. The values within this array indicate to which segment the corresponding flux/weight/flag/wave belong. If this column is not present it is assumed the Spectrum1d contains only one {[SpectralSegment](#) spectral segment}.

The other part that a spectrum needs is a frequency or wavelength scale. See {[herschel.ia.dataset.spectrum.AbstractSpectrumDataset](#)}

None of the predefined columns really needs to exist. There can be reasons for any of them to be absent. If the column is not present a `IndexOutOfBoundsException` will be thrown. So unless you really know always use a `try{ }catch()` or use the `hasFlux()`, `hasWeight()` etc methods.

Example

Example 1: In Jide:

```
#flux is a Double1d
#segment is a Int1d of the same length containing [1,1,...1,2,2,...2,3,3...]
F = DataFormatter()
s1 = Spectrum1d()
s1.setFlux( flux -)
s1.setSegment( segment -)
it = s1.iterator()
seg = s1.getSpectralSegment()
while it.hasNext() -:
    print F.p( seg.getFlux() -)          # gets the 1's, the 2's etc.
    seg = it.next()
```


Limitations

Spectrum1d still is a TableDataset and can be addressed as such. If you do so, the special methods of Spectrum1d (and its predecessors, AbstractSpectrumDataset and StrictTableDataset) are **not** guaranteed to work.

History

- 06-03-2006 DK.

2.396. Spectrum2d

Full Name:	herschel.ia.dataset.spectrum.Spectrum2d
Type:	Java Class - 
Import:	from herschel.ia.dataset.spectrum import Spectrum2d
Category:	Datasets

Description

Spectrum2d is an extension of AbstractSpectrumDataset which in turn is an extension of (Strict)TableDataset. It is a 2-dimensional incarnation of a SpectrumDataset.

Within a Spectrum2d there are provisions for 3 (predefined) columns.

1. flux, a Double2d array, where the first axis runs over the spectral dimension and the second axis runs over e.g. time. In the following this time index is called sequential index or sequindex.

There is no requirement that either axis projects monotonically or equidistantly on the frequency/sequindex scale. It is not even necessary that they are independent, ie. frequency can change over sequindex. This flux column is more or less obligatory, otherwise there can't be much talk of a spectrum.

2. weight. An Double2d array of the same dimensionality as flux.

weight = 0, means that the corresponding samples are irrelevant.

weight = 1 / stdev², if such a value is available for the fluxes.

3. flag, an Int1d array of the same size as flux. The definition of flags is of course problem dependent. It is suggested to store the meaning of the flags in MetaData.

The other part that a spectrum needs is a frequency or wavelength scale. See {@link herschel.ia.dataset.spectrum.AbstractSpectrumDataset}

As a further extension of the Spectrum2d we introduce the concept of "subbands". Subbands are vertical splits in the flux (and weight, flag etc.) columns, equivalent to (functionality of) the segment column in {@link Spectrum1d}. The flux etc. columns are replaced with flux_1, flux_2, ... columns, depending on how many subbands were defined. The definition of the subbands is stored in small array MetaData subbandstart and subbandlength. {@link herschel.ia.dataset.spectrum.StrictTableDataset#setMeta(String name, Double1d data)} Operations of split and join are exact inverses of each other as long as the metadata of subbandstart and subbandlength is kept consistent and provided that the subband ranges cover the complete width of flux. Otherwise what is lost will stay so.

The {@link SpectralSegment SpectralSegment} interface is fully implemented on Spectrum2d, whether it has subbands or not. When there are no subbands, each time a next segment is requested, the next row is returned. When subbands are present, first a row from the next subband at the same sequential index is returned. When there are no more subbands, the row at the next sequindex in the first subband is returned. This behaviour can be overruled by {@link #setColumnFirst(boolean cf)}.

Example

Example 1: In Jide:

```
flux = Double2d()
for i in range(5) -: flux.append( Double1d( range(1000 -) -) + i, 0 -)
```

Example 1: In Jide:

```
flag = Int2d( 5, 1000 -)
s2 = Spectrum2d( flux, None, flag -)
F = DataFormatter()
print F.p( s2.getFlux( -), 6 -)
s2.setMeta( -"subbandstart", Int1d( [0,100,500] -) -)
s2.setMeta( -"subbandlength", Int1d( [100,200,400] -) -)
s2.splitInSegments( ["flux","flag"] -);
print F.p( s2.getFlux( 2 -) -)
print s2.getSegmentCount()           # 15 = 5 * 3
it = s2.iterator()
seg = s2.getSpectralSegment()
while it.hasNext() -:
    seg = it.next()
    print F.p( seg.getFlux() -)
```


Limitations

Spectrum2d still **is** a TableDataset and can be addressed as such. If you do so, the special methods of Spectrum2d (and its predecessors, AbstractSpectrumDataset and StrictTableDataset) are **not** guaranteed to work.

History

- 07-03-2006 DK.

2.397. SpectrumAvgPlotting

Full Name:	herschel.ia.gui.cube.SpectrumAvgPlotting
Type:	Java Class - 
Import:	from herschel.ia.gui.cube import SpectrumAvgPlotting

Description

A class to deal with SpectrumAvgPlotting.

API Summary


Methods
setUsedArea(int area)
setUsedRegion(int region)
Double2d getRectangleArray() Returns the selected region as ROI (region of interest) if the
__updatePolygon__() The real implementation of the updatePolygon method.

API details

Methods

setUsedArea(int area)
Argument int area [INPUT, MANDATORY]
setUsedRegion(int region)
Argument int region [INPUT, MANDATORY]
Double2d getRectangleArray() Returns the selected region as ROI (region of interest) if the selected region is bounded by a circle; null otherwise.
Return Double2d Returns the selected region as ROI (region of interest) if the selected region is bounded by a circle; null otherwise.
__updatePolygon__() The real implementation of the updatePolygon method. The real implementation of the updatePolygon method.

2.398. SpectrumPlotting

Full Name:	herschel.ia.gui.cube.SpectrumPlotting
Type:	Java Class - 
Import:	from herschel.ia.gui.cube import SpectrumPlotting

Description

SpectrumPlotting

A class to deal with CubeSpectrumAnalysis, the real computation of the spectrum is made in the task ExtractSpectrum()

API Summary

Constructor	
SpectrumPlotting(boolean useAsComponent, CubeSpectrumAnalysisToolbox toolbox)	Constructor for SpectrumPlotting. When the new SpectrumPlotting is
Methods	
setClicked(Double center)	Sets the center of the rectangle, associated with this
ImageFigure getRect()	Returns the straight line (ImageFigure), of which we wish to plot
Double getClicked()	Returns the clicked position in PixelCoordinates.
int getClicks()	Returns the number of valid clicks (i.e. in the image) you made.
String getSkyCoordinates()	Returns the String version of the sky coordinates
String getPixelCoordinates()	Returns the String version of the PixelCoordinates center of
increaseClicks()	Increases the number of valid clicks (i.e. in the image) you made.
drawRect()	Draws and returns a new rectangle on the image, we are analysing with
updatePlot()	Updates the plot in this ProfilePlotFrame, with the given plot
ReadValuesFilterFitter()	Clears the rectangle in the left part of the SpectrumPlotting.
ArrayList getImageFigures()	Returns all ImageFigures used for this SpectrumPlotting (the
DoubleId getSpectrum()	Returns a DoubleId at this date containing the spectrum values

Methods[SpectrumId](#) `getSpectrumId()`

Returns a SpectrumId containing the currently extracted spectrum

API details

Constructor

`SpectrumPlotting(boolean useAsComponent, CubeSpectrumAnalysis-Toolbox toolbox)`

Constructor for SpectrumPlotting. When the new SpectrumPlotting is component-based, a window for plotting the intensity is opened; otherwise nothing will be shown.

Argumentsboolean `useAsComponent` [INPUT, MANDATORY][CubeSpectrumAnalysisToolbox](#) `toolbox` [INPUT, MANDATORY]

Methods

`setClicked(Double center)`

Sets the center of the rectangle, associated with this

SpectrumPlotting to the given point (in UserCoordinates).

Argument[Double](#) `center` [INPUT, MANDATORY]`ImageFigure getRect()`

Returns the straight line ([ImageFigure](#)), of which we wish to plot the intensity in this SpectrumPlotting.

Return[ImageFigure](#)The straight line ([ImageFigure](#)), of which we wish to plot the intensity in this SpectrumPlotting.`Double getClicked()`

Returns the clicked position in PixelCoordinates.

Return[Double](#)

Returns the clicked position in PixelCoordinates.

`int getClicks()`

Returns the number of valid clicks (i.e. in the image) you made.

Return`int`

Returns the number of valid clicks (i.e. in the image) you made.

String getSkyCoordinates()
Returns the String version of the sky coordinates (WorldCoordinates) of the point Clicked. Return String Returns the String version of the sky coordinates (WorldCoordinates) of the point Clicked.
String getPixelCoordinates()
Returns the String version of the PixelCoordinates center of the rectangle which is also the point analysed. Return String Returns the String version of the PixelCoordinates.
increaseClicks()
Increases the number of valid clicks (i.e. in the image) you made.
drawRect()
Draws and returns a new rectangle on the image, we are analysing with the ImageAnalysisToolbox, associated with this SpectrumPlotting.
updatePlot()
Updates the plot in this ProfilePlotFrame, with the given plot points, which represent the intensity along the given straight line (CanvasFigure) and updates the SkyCoordinates (if available) and the PixelCoordinates.
ReadValuesFilterFitter()
Clears the rectangle in the left part of the SpectrumPlotting.
ArrayList getImageFigures()
Returns all ImageFigures used for this SpectrumPlotting (the rectangle on the image). Return ArrayList Returns all ImageFigures used for this SpectrumPlotting (the rectangle on the image).
Double1d getSpectrum()
Returns a Double1d at this date containing the spectrum values Return Double1d

[Double1d](#) `getSpectrum()`

Returns a Double1d at this date containing the spectrum values

[Spectrum1d](#) `getSpectrum1d()`


Returns a Spectrum1d containing the currently extracted spectrum

Return

[Spectrum1d](#)

Returns a Spectrum1d containing the currently extracted spectrum

2.399. SpectrumStatistics

Full Name:	herschel.ia.toolbox.spectrum.SpectrumStatistics
Type:	Java Task - 
Import:	from herschel.ia.toolbox.spectrum import SpectrumStatistics

Description

Task to compute statistical characteristics for the spectrum data included in one or several containers.

Mean, RMS and median are always reported - percentiles can be given on demand. Two modes are available: Compute the statistics over a set of PointSpectra on a channel by channel basis or compute the statistics for a range within a single point spectrum.

Example

Example 1: from Jide:

```
from herschel.ia.toolbox.spectrum import SpectrumStatistics
# ds: some spectrum container
statistics = SpectrumStatistics()
stats = statistics(ds=ds, mode="all") # stats is a product
stats = statistics(ds=ds, mode="perChannel") # stats is a product but
without -"summary" TableDataset
stats = statistics(ds=ds, mode="acrossChannels") # stats is a TableDataset
stats = statistics(ds=ds, mode = -"all", percentiles=[0.2,0.8], ranges =
Range(500,1500))
stats = statistics(ds=ds, mode = -"acrossChannels", percentiles=[0.2,0.8],
ranges = (4200,4500))
stats = statistics(ds=ds, mode = -"acrossChannels", percentiles=[0.2,0.8],
ranges = [(4200,4500),(7300,7600)])
```

API Summary

Properties
SpectrumContainer ds [INPUT, MANDATORY, default=no default value.]
String mode [INPUT, OPTIONAL, default=default value 'all'.]
Object stats [OUTPUT, OPTIONAL, default=no default value.]
Object ranges [INPUT, OPTIONAL, default=no default value.]
Object exclude [INPUT, OPTIONAL, default=no default value.]
double[] percentiles [INPUT, OPTIONAL, default=no default value.]
Boolean ignoreNaNs [INPUT, OPTIONAL, default=True.]

API details

Properties

SpectrumContainer ds [INPUT, MANDATORY, default=no default value.]

The input container(s) to be considered.

String mode [INPUT, OPTIONAL, default=default value 'all'.]

Contains the output with the statistics when the task is used in the normal mode. Possible values are 'acrossChannel', 'perChannel', 'all'. 'acrossChannel'-mode: Here, the statistics are computed separately for each spectrum and each segments. If a range has been set, this range is taken accordingly. 'perChannel'-mode: Here, the statistics are computed for all the spectra included in the container on a per channel basis. 'all': Both, the per channel and the across channel statistics are computed.

Object stats [OUTPUT, OPTIONAL, default=no default value.]

If the task is used in the per channel mode, a product is created which contains for each statistical characteristics and each sub-segment one SpectrumId. In case 'mode' is set to 'all', the product also contains the across channel statistics as a "summary". In case the task is used in the across channel mode, the statistics are written in a TableDataset (each point spectrum included in the input spectrum container corresponds a line in the table).

Object ranges [INPUT, OPTIONAL, default=no default value.]

Specify the ranges that should be considered within the spectra when computing the 'across channels'-statistics. You have various alternatives to do that:

- Specify a 'Range'-object: Then this range object is considered for all the sub-segments as the range of pixel numbers to select.
- Specify an array of 'Range'-objects: Here, each Range object in the array is associated with the Range of pixel number to select for each sub-segments. The number of ranges specified in the array must correspond to the number of segments.
- Specify a tuple with the lower and upper bound of a frequency interval to restrict to.
- Specify a (py-)array of tuples with the lower and upper bounds of frequency intervals to restrict the calculation of the statistics to.

Object exclude [INPUT, OPTIONAL, default=no default value.]

Specify the ranges that should be excluded when computing the 'across channels'-statistics. You have various alternatives to do that:

- Specify a 'Range'-object: Then this range of pixels is excluded in all sub-segments.
- Specify an array of such 'Range'-objects: Here, each range corresponds to one range of pixels for one sub-segment. The number of ranges specified in the array must correspond to the number of segments.
- Specify a tuple with the lower and upper bound of a frequency interval to exclude.
- Specify a (py-)array of tuples with the lower and upper bounds of frequency intervals to exclude in the calculation of the statistics.

double[] percentiles [INPUT, OPTIONAL, default=no default value.]

Specify what percentiles should be given in the output (a list of probabilities).


Boolean ignoreNaNs [INPUT, OPTIONAL, default=True.]

Flag to specify that NaN's found in the flux data should be ignored when computing the statistics.

History

- 2008-06-05 - meli: initial.
- 2008-06-14 - meli: more methods to specify ranges.
- 2009-06-05 - meli: ignoreNaNs added.

2.400. SpectrumTask

Full Name:	herschel.ia.toolbox.spectrum.SpectrumTask
Type:	Java Task - 
Import:	from herschel.ia.toolbox.spectrum import SpectrumTask

Description

Abstract base class for tasks processing { @link SpectrumContainer } data structures.


2.401. SpireFlags

Full Name:	herschel.ia.obs.quality.SpireFlags
Type:	Java Class - 
Import:	from herschel.ia.obs.quality import SpireFlags

History

- 03 Jun 09: Renamed several "get" methods to address SPR 6995.
- 16 Jul 09: Added more quality control flags (see SPR-7431).

2.402. SQRT

Full Name:	herschel.ia.numeric.toolbox.basic.Sqrt
Alias:	SQRT
Type:	Java Class - 
Import:	from herschel.ia.numeric.toolbox.basic import Sqrt

Description

Computes $x=\#x$, where x is a number or a numeric array.

Example

Example 1: Apply SQRT on a Int1d

```
x=Int1d( [0,4,9] -)
print SQRT(x) # [0,2.,3.]
```

API Summary

Jython Syntax

```
<y>=SQRT(<x>)
```

Properties

any number or array **x** [INPUT, MANDATORY, default=no default value]
 boolean array or a boolean **y** [OUTPUT, MANDATORY, default=no default value]

API details

Properties


any number or array **x** [INPUT, MANDATORY, default=no default value]

The input can be a number or an array

boolean array or a boolean **y** [OUTPUT, MANDATORY, default=no default value]

Returns an array (or a number if the input is not an array) where each element is the the # of the corresponding element of the input array

2.403. SQUARE

Full Name:	herschel.ia.numeric.toolbox.basic.Square
Alias:	SQUARE
Type:	Java Class - 
Import:	from herschel.ia.numeric.toolbox.basic import Square

Description

Computes $x=x^2$, where x is a number or a numeric array.

Example

Example 1: Apply SQUARE on a Int1d
<pre>x=Int1d([-1,0,2] -) print SQUARE(x) # [1,0,4]</pre>

API Summary


Jython Syntax
<code><y>=SQUARE (<x>)</code>
Properties
any number or array x [INPUT, MANDATORY, default=no default value]
boolean array or a boolean y [OUTPUT, MANDATORY, default=no default value]

API details

Properties

any number or array x [INPUT, MANDATORY, default=no default value]
The input can be a number or an array
boolean array or a boolean y [OUTPUT, MANDATORY, default=no default value]
Returns an array (or a number if the input is not an array) where each element is the the square of the corresponding element of the input array

2.404. Sso

Full Name:	herschel.ia.toolbox.pointing.Sso
Type:	Java Class - 
Import:	from herschel.ia.toolbox.pointing import Sso
Category:	toolbox

Description

To obtain an Horizons object you need an Ephemeris object and a directory with SSO files


Example

Example 1: Untitled
TODO

History

- 20100329 Changed implementation to just remove tracking (properly)
- Tools for SSO centered coordinate systems
- The origin of a object-centered reference frame is defined as the instantaneous ephemeris of the solar system object
- To reposition SKY(X,Y) as (OCX,OCY)
- For Attitude

2.405. STDDEV

Full Name:	herschel.ia.numeric.toolbox.basic.StdDev
Alias:	STDDEV
Type:	Java Class - 
Import:	from herschel.ia.numeric.toolbox.basic import StdDev

Description

Yields the standard deviation of the elements in the input array.

The standard deviation is equivalent to $\text{SQRT}(\text{VARIANCE}(x))$.

Example

Example 1: Apply STDDEV on a Float1d

```
x=Float1d([1,3,2,3,4])
print STDDEV(x) # 1.3
```

API Summary

Jython Syntax

```
<y>=STDDEV(<x>)
```

Properties

[any scalar array **x** \[INPUT, MANDATORY, default=no default value\]](#)

[double **y** \[OUTPUT, MANDATORY, default=no default value\]](#)

API details

Properties

any scalar array **x [INPUT, MANDATORY, default=no default value]**

The input array integral or floating-point arrays; the former implicitly transformed to a double array.


double **y [OUTPUT, MANDATORY, default=no default value]**

Returns a double

See also

- [MEAN](#)
- [MEDIAN](#)
- [VARIANCE](#)

2.406. StitchSpectrum

Full Name:	herschel.ia.toolbox.spectrum.StitchSpectrum
Type:	Java Task - 
Import:	from herschel.ia.toolbox.spectrum import StitchSpectrum

Description

Task for stitching the spectra (segments) included in a SpectrumContainer.

The stitching is performed on a per point spectrum (scan) basis - except for the variant 'stitchAll' which tries to combine all segments. The result is again a point spectrum that still may consist of several segments - in case gaps are found between consecutive segments. The way how the stitching is done is defined by the parameter 'variant' (see below). Once the stitching is done, the resulting point spectra are included in a new spectrum container. Here, in general, resampling is needed so that all the resulting point spectra have the same shape (which is a requirement so that the point spectra can be included in a common container (dataset)). In case no or a zero stepsize (which is the channel width for the resampled spectra) is specified, resampling is avoided if all the stitched point spectra happen to have all the same shape.

Example

Example 1: from Jide:

```
# all defaults -- cut at crossover points and concatenate
stitched11 = stitch(ds=spectra)
# cut at crossover points and concatenate
stitched12 = stitch(ds=spectra, variant="crossoverPoints")
# cut at crossover points and concatenate, consider crossover points only by
10 percent off the border to of the overlapping region
stitched13 = stitch(ds=spectra, variant="crossoverPoints", edgeTolerance=0.1)
# cut at crossover points and concatenate, resample the resulting spectra to
1MHz channel width
stitched14 = stitch(ds=spectra, variant="crossoverPoints", edgeTolerance=0.1,
stepsize=1.0, unit="MHz")
# cut at mid points of the overlapping regions and concatenate
stitched21 = stitch(ds=spectra, variant="midPoints")
# cut at mid points of the overlapping regions and concatenate, resample to
1.0 channel width (in units of the underlying frequency scale)
stitched22 = stitch(ds=spectra, variant="midPoints", stepsize=1.0)
# cut at predefined split points and concatenate
stitched31 = stitch(ds=spectra, variant="splitPoints", splitPoints = [5000.0,
6000.0, 7000.0])
# cut at predefined split points and concatenate, resample to 1.0 channel
width (in units of the underlying frequency scale)
stitched32 = stitch(ds=spectra, variant="splitPoints", splitPoints = [5000.0,
6000.0, 7000.0], stepsize=1.0)
# cut at predefined split points and concatenate, resample to 0.001 GHz
channel width
stitched33 = stitch(ds=spectra, variant="splitPoints", splitPoints = [5.0,
6.0, 7.0], stepsize=0.001, unit="GHz")
# average the parts of the spectra that are overlapping, resample to a 1.0
channel width (in units of the underlying frequency scale)
stitched4 = stitch(ds=spectra, variant="average", stepsize=1.0)
# average the parts of the spectra that are overlapping, resample to a 1.0
MHz channel width, do not consider flags and weights
# when computing teh average (but propagate them)
stitched41 = stitch(ds=spectra, variant="average", stepsize=1.0, unit="MHz",
avg_variant="flux")
# average the parts of the spectra that are overlapping, avoid resampling as
far as possible
stitched42 = stitch(ds=spectra, variant="average", avg_variant="flux")
```

API Summary

Properties
<code>SpectrumContainer ds</code> [INPUT, MANDATORY, default=no default value.]
<code>SpectrumContainer result</code> [OUTPUT, MANDATORY, default=no default value.]
<code>String variant</code> [INPUT, OPTIONAL, default="crossoverPoints".]
<code>double edgeTolerance</code> [INPUT, OPTIONAL, default=0.01.]
<code>Object splitPoints</code> [INPUT, OPTIONAL, default=no default value]
<code>double stepsize</code> [INPUT, OPTIONAL, default=no default value.]
<code>String unit</code> [INPUT, OPTIONAL, default=no default value.]
<code>String avg_variant</code> [INPUT, OPTIONAL, default=no default value.]

API details

Properties

<code>SpectrumContainer ds</code> [INPUT, MANDATORY, default=no default value.]
The input container with the spectra to be stitched.
<code>SpectrumContainer result</code> [OUTPUT, MANDATORY, default=no default value.]
The output container with the stitched spectra.
<code>String variant</code> [INPUT, OPTIONAL, default="crossoverPoints".]
<p>The parameter that specifies the way how the stitching should be done. Possible values:</p> <ul style="list-style-type: none"> "crossoverPoints": In overlapping ranges of the spectra a crossover point is taken to split the spectra. In case no crossover point is found a point is taken where the spectra come closest. In case many crossover points are found the one closest to the midpoint of the overlapping range is taken. The range where the crossover points are searched for is reduced by setting a non-zero edge tolerance (see parameter <code>edgeTolerance</code> below). This option only works if at most two segments 'participate' in the same overlapping range. "midPoint": In the overlapping ranges of the spectra the mid point is taken to split the spectra. Similar to the above, this option only works if at most two segments 'participate' in the same overlapping range. "splitPoints": In the overlapping ranges of the spectra previously specified points are taken to split the spectra (parameter <code>splitPoints</code>). the mid point is taken. Similar to the above, this option only works if at most two segments 'participate' in the same overlapping range. "average": In the overlapping ranges the average of the involved spectra is taken. This option allows to stitch spectra in which more than two spectra contribute to the same overlapping part. "stitchAll": Try to stitch all the segments found in the <code>SpectrumContainer</code> - irrespective of what point spectrum the segments stem from. In the overlapping ranges, take the average of the involved spectra.

double edgeTolerance [INPUT, OPTIONAL, default=0.01.]

Reduce the range (an overlapping range in the spectra) in which cross over points are searched. With `length` denoting the length of the original overlapping range $[n, m]$ (in number of pixels) the reduced range is defined by $[n+p*length, m-p*length]$ where p is the edge tolerance.

Object splitPoints [INPUT, OPTIONAL, default=no default value]

Specify the list of split points explicitly. This parameter is used only in combination with the option `variant="splitPoints"`. The split points are specified as a Python list. Note that the number of split points should correspond to the number of overlapping ranges (per point spectrum).

double stepsize [INPUT, OPTIONAL, default=no default value.]

Specify the stepsize used to define a linear frequency the spectra are resampled to. The resampling is important in the following situations:

- When including the stitched point spectra in a common container. In case a `stepsize>0` is specified, a frequency grid is constructed with the given step size. The upper and lower bounds are determined from the minimum and maximum frequencies found in all the stitched point spectra (for a give segment index). In case a `stepsize=0` is specified, the frequency grid found in the first (stitched) point spectrum is taken as the output grid.
- When using `variant="average"` the spectra contributing to common overlapping frequency ranges are resampled to a common frequency grid.

When searching for crossover points, one of the spectra also needs to be resampled to a common frequency scale - here, the frequency scale of the other spectrum is used, though so that the `stepsize` parameter is not needed here.

String unit [INPUT, OPTIONAL, default=no default value.]

The unit the stepsize and/or split points are expressed in. In case no unit is specified the stepsize and/or the split points are assumed to be in the same units as the underlying frequency grid.


String avg_variant [INPUT, OPTIONAL, default=no default value.]

The variant of the average to be used: whether to include flags and weights in the processing.

History

- 2009-11-21 - meli: initial.

2.407. String1d


Full Name:	herschel.ia.numeric.String1d
Type:	Java Class - 
Import:	from herschel.ia.numeric import String1d

Description

A rectangular numeric String array of rank 1.

Numeric arrays are part of the Numeric library. An explanation on how to use them, you can find in chapter **Numeric** -> **Arrays**. (Yes, it should be a link)

2.408. SubtractSpectrum

Full Name:	herschel.ia.toolbox.spectrum.SubtractSpectrum
Type:	Java Task - 
Import:	from herschel.ia.toolbox.spectrum import SubtractSpectrum

Description

Task for subtracting a scalar from the flux data included in a spectrum container or for subtracting two spectrum containers from each other on a spectrum-by-spectrum basis.

For the scalar mode, 'ds' and 'param' need to be specified - for the pair-wise mode, 'ds1' and 'ds2' need to be set. In the pairwise mode, the task iterates through the two containers and computes the pair-operation. In case the size of the two containers is different, the iteration stops at the minimum size. Hence, the remaining spectra in the larger container are ignored. In case one of the datasets only contains a single spectrum, the task always refers to this single spectrum while iterating over all the spectra in the other dataset.

Different selection schemes are available for selecting the point spectra or the segments to be processed. The most simple scheme is to specify lists of point spectrum indices (`selection=[1, 3, 4, 2]`). In this case, the operation (scalar- or pair-operation) is just taken over the point spectra with corresponding indices. Instead of specifying a list of indices (which requires the knowledge of these) other recipes to specify a selection are available. Typical recipes test attribute values to match certain values or to be included in a given interval. These recipes we refer to as selection models. The different options to specify selections can be combined. Here, an AND logic is adopted.

Similarly, you can specify what segments to consider. In the most simple case, you can just specify the indices of segments to be considered (for the scalar operation `segments=[1, 3, 4]` and for the pair operation `segments1=[1, 3, 4]`, `segments2=[3, 6, 5]`). In more advanced situations you can use a `SegmentSelection` object. Note that the pairs of segments to be processed need to be consistent - otherwise the processing will fail.

You can specify whether the original datasets should be overwritten with the 'overwrite' flag. By default no data is overwritten. In the pair-wise mode, it is not always possible to overwrite the data - in particular if non-trivial segment selections are specified.

For the pair-wise mode, there are further options that can be selected by the 'variant' (e.g. whether to use flags and weights).

Example

Example 1: from Jide:

```
from herschel.ia.toolbox.spectrum import SubtractSpectrum
subtract = SubtractSpectrum()
subtractConstant = subtract(ds=spectra, param=2.1)
subtractedFactor = subtract(ds=spectra, selection={"bbtype": [6031]}, param=2.1)
subtractedFactor = subtract(ds=spectra, selection=[0,1,2,3], param=2.1)
subtractedFactor = subtract(ds=spectra, selection={"Chopper":
([-4.4,5.9],0.2), -"bbtype": [6031]}, param=2.1)
subtractedFactor = subtract(ds=spectra, selection={"LoFrequency":
(4000.0,5000.0)}, param=2.1)
subtractedPairWise = subtract(ds1=spectral1, ds2=spectra2)
subtractedPairWise = subtract(ds1=spectral1, ds2=spectra2, selection={"bbtype":
[6031]})
subtractedPairWise = subtract(ds1=spectral1, ds2=spectra2, selection=[0,1,2,3])
subtractedPairWise = subtract(ds1=spectral1, ds2=spectra2, selection={"Chopper":
([-4.4,5.9],0.2), -"bbtype": [6031]})
```

Example 1: from Jide:

```
subtractedPairWise = subtract(ds1=spectral1, ds2=spectra2,
selection={"LoFrequency":(4000.0,5000.0)})
```

API Summary

Properties
SpectrumContainer ds [INPUT, OPTIONAL, default=no default value.]
Double param [INPUT, OPTIONAL, default=no default value.]
Object selection [INPUT, OPTIONAL, default=None.]
PyDictionary Map<T> selection_lookup [INPUT, OPTIONAL, default=no default value.]
PyList selection_index [INPUT, OPTIONAL, default=No default value.]
Boolean overwrite [INPUT, OPTIONAL, default=False.]
SpectrumContainer ds1 [INPUT, OPTIONAL, default=no default value.]
SpectrumContainer ds2 [INPUT, OPTIONAL, default=no default value.]
Object segments [INPUT, OPTIONAL, default=no default value.]
Object segments1 [INPUT, OPTIONAL, default=no default value.]
Object segments2 [INPUT, OPTIONAL, default=no default value.]
String variant [INPUT, OPTIONAL, default=no default value.]
SpectrumContainer result [OUTPUT, OPTIONAL, default=no default value.]

API details

Properties

SpectrumContainer ds [INPUT, OPTIONAL, default=no default value.]
Input container to be processed by the task in case the task is used as a scalar operation.
Double param [INPUT, OPTIONAL, default=no default value.]
The scalar parameter to be considered when the task is used as scalar operation.
Object selection [INPUT, OPTIONAL, default=None.]
Specify what point spectra the operation should be applied to. Different ways to specify these selections are possible: <ul style="list-style-type: none"> • Specify a list of indices (in jython) of the point spectra for which the add should be applied. • Use a dictionary (in jython) to specify a selection model (lookup for attributes, filter attributes to be included in ranges or intervals). • Pass a string representation of a jython list or a jython dictionary. The string will be parsed and the processing delegated to the two cases above. • Pass any java instance that implements the SelectionModel interface (for the advanced user).

PyDictionary|Map selection_lookup [INPUT, OPTIONAL, default=no default value.]

Specify a PyDictionary with keys given by the attribute name to look up and a list of admissible values. Behind the scenes, a DiscreteValueSelectionModel is instantiated.

PyList selection_index [INPUT, OPTIONAL, default=No default value.]

Specify a PyList with the indices of the point spectra to be considered.

Boolean overwrite [INPUT, OPTIONAL, default=False.]

Specify whether the input data container can be reused - the values found therein is overwritten.

SpectrumContainer ds1 [INPUT, OPTIONAL, default=no default value.]

First input container for pair-wise operations.

SpectrumContainer ds2 [INPUT, OPTIONAL, default=no default value.]

Second input container for pair-wise operations.

Object segments [INPUT, OPTIONAL, default=no default value.]

Specify what segments the operation should be applied to. There are two options available:

- Either pass an instance of a SegmentSelection which gives the information on what segments for each point spectrum included in the container.
- Specify a PyList of segment indices.

Object segments1 [INPUT, OPTIONAL, default=no default value.]

Specify the segment selection to be associated with 'ds1'.

Object segments2 [INPUT, OPTIONAL, default=no default value.]

Specify the segment selection to be associated with 'ds2'.

String variant [INPUT, OPTIONAL, default=no default value.]

Specify the variant of processing mode you would like to run (including flags / weights / ...). HCSS defaults include: "flux" / "flux-wave" / "flux-wave-weight" / "flux-flag-wave" / "flux-weight-flag-wave"

SpectrumContainer result [OUTPUT, OPTIONAL, default=no default value]

Result object containing the results of the operation applied.

See also


- [SpectrumTask](#)

History

- 2007-08-17 - meli: initial.

- 2008-03-16 - meli: major refactoring of the toolbox.
- 2008-10-03 - meli: segments, javadoc

2.409. SUM

Full Name:	herschel.ia.numeric.toolbox.basic.Sum
Alias:	SUM
Type:	Java Class - 
Import:	from herschel.ia.numeric.toolbox.basic import Sum

Description

Yields the sum of all elements in the input array.

Returns a scalar of the same type as the type of the input array.

Example

Example 1: Apply SUM on a Int2d

```
x=Int2d( [ [1,2], [-1,3] -])
print SUM(x)      # 5
print SUM(x, 0) # [0,5]
print SUM(x, 1) # [3,2]
```

API Summary

Jython Syntax

```
<y>=SUM(<x>, [ ,<dim>])
```

Properties

[any scalar array **x** \[INPUT, MANDATORY, default=no default value\]](#)

[integer **dim** \[INPUT, MANDATORY, default=no default value\]](#)

[float or double array **y** \[OUTPUT, MANDATORY, default=no default value\]](#)

API details

Properties

any scalar array **x [INPUT, MANDATORY, default=no default value]**

The input array can be any scalar array.

integer **dim [INPUT, MANDATORY, default=no default value]**

The dimension to compute the calculation along


float or double array **y [OUTPUT, MANDATORY, default=no default value]**

Returns a scalar of the same type as the type of the input array.

See also

- [PRODUCT](#)

2.410. TablePlotter

Full Name:	herschel.ia.gui.explorer.table.TablePlotter
Type:	Java Class - 
Import:	from herschel.ia.gui.explorer.table import TablePlotter

Description

TablePlotter is a GUI tool to view and analyze TableDataset. It can be invoked in JIDE and HIPE by right clicking on a TableDataset. It can also be invoked by a Jython command line in JIDE or Hipe through jython script. By default, the second column is plotted as y data and first column as x data unless x and y index are specified. All the default can be changed vs a group of buttons and selectors.

Example

Example 1: Invoke TablePlotter from command line TablePlotter is a pluggable

```

component which can be plugged in any GUI containers. Example: This
example shows how to use TablePlotter in command line
<pre>
from herschel.ia.io.fits import FitsArchive
from herschel.ia.dataset import TableDataset
from herschel.ia.dataset import Product
from herschel.ia.gui.explorer.table import TablePlotter
from herschel.ia.gui.explorer.table import OverPlotter
from herschel.ia.gui.explorer.table import *
from herschel.share.component import *
#####
# Load your data
#####
fits=FitsArchive();
path=&quot;/home/zhang/Development/data/&quot;;
filename = &quot;loadCurve.fits&quot;;
try:
fits.reader = FitsArchive.HCSS_READER
p = fits.load(path+filename)
except java.io.IOException:
fits.reader = FitsArchive.STANDARD_READER
p = fits.load(path+filename)
#####
# Load TablePlotter with a TableDataset only
#####
tbs =p.default
tpl=TablePlotter(tbs)
wm=WindowManager.getDefault()
#Load TablePlotter
wm.addWindow('test', tpl.component, 1)
#get extracted table
extractedTable = tpl.activeLayerStruct.extractedTableDataset
#get flags
flags = tpl.activeLayerStruct.flags
print flags.size
print flags.dimensions
#####
# Load a TablePlotter with a 2d flags
#####
tpl1=TablePlotter(tbs, flags, TablePlotter.ALL_COLUMNS)
wm=WindowManager.getDefault()
print tbs.rowCount
#Load TablePlotter
wm.addWindow('test 2d fkags', tpl1.component, 1)
#####
# Load TablePlotter with a 1d flag

```


Example 1: Invoke TablePlotter from command line TablePlotter is a pluggable

```
#####
flag=flags.get(Range(0, tbs.rowCount), 1)
tpl2=TablePlotter(tbs, flag, TablePlotter.ALL_COLUMNS)
wm=WindowManager.getDefault()
print tbs.rowCount
#Load TablePlotter
wm.addWindow('test 1d flags', tpl2.component, 1)
flagsId = tpl2.activeLayerStruct.flags
print flagsId.size
</pre>

```

API Summary

Constructors	
TablePlotter()	A default constructor This is a default constructor which will be
TablePlotter(TableDataset tds)	A constructor This constructor will initialize and create an
TablePlotter(TableDataset tds, integer xIndex, integer yIndex)	A constructor This constructor will initiate an instance of an
TablePlotter(TableDataset tds, integer xIndex, integer yIndex, boolean initialColumnMode)	A constructor This constructor will initiate an instance of an
TablePlotter(TableDataset tds, integer xIndex, integer yIndex, boolean initialColumnMode, boolean dataExtractionAllowed)	A constructor This constructor will initiate an instance of an
TablePlotter(TablePlotter tds, BoolId flags)	This constructor will initiate an instance of TablePlotter with
TablePlotter(TableDataset tds, BoolId flags, integer xIndex, int yIndex)	A constructor This constructor will initiate an instance of
TablePlotter(TableDataset tds, BoolId flags, integer xIndex, int yIndex, boolean dataExtractionAllowed)	A constructor This constructor will initiate an instance of
TablePlotter(TableDataset tds, Bool2d flags, boolean initial-ColumnState)	A constructor This constructor will initiate an instance of
TablePlotter(TableDataset tds, Bool2d flags, boolean initial-ColumnState, boolean dataExtractionAllowed)	
Methods	
JComponent GetComponent()	Inherit from explorer This method will return a TablePlotter as a
String getDescription()	
String getName()	
setObject(Object data)	
LayerStruct getActiveLayerStruct()	

Methods

This is a command line API. It is a getter to get the active

Limitations

The TableDataset has to be one dimensional numeric array such as Double1d, Float1d, Long1d, Short1d and Complex1d.

Miscellaneous

All the examples here are given in Jython script

API details**Constructors****TablePlotter()**

A default constructor This is a default constructor which will be called in DatasetInspector It initializes the DataObjectLinstersSupport for data extraction.

TablePlotter(TableDataset tds)

A constructor This constructor will initialize and create an instance of TablePlotter for the input dataset.

Argument

TableDataset **tds** [INPUT, MANDATORY, default=no default value A] TableDataset to be plotted. It contains one or more columns.

Example

Create a TablePlotter instance with a tds dataset from

```
hersechl.ia.gui.explorer.table import TablePlotter #import
TablePlotter tpl=TablePlotter(tds) #tds is a dataset defined
somewhere else
```

TablePlotter(TableDataset tds, integer xIndex, integer yIndex)

A constructor This constructor will initiate an instance of an TablePlotter with three input, a table, the xIndex and the yIndex.

Arguments

TableDataset **tds** [INPUT, MANDATORY, default=no default value A] TableDataset to be plotted and viewed. It contains one or more columns.

integer **xIndex** [INPUT, MANDATORY, default=no default value The xIndex]

is an integer which indicates the x column data

integer **yIndex** [INPUT, MANDATORY, default=no default value The]

yIndex is an integer which indicates the y column data

Example

TablePlotter(TableDataset tds, integer xIndex, integer yIndex)

create an instance of TablePlotter where column 3 is x and

```
column 10 is y from hersechl.ia.gui.explorer.table import
TablePlotter #import the TablePlotter #plot the 10th column vs
the first column from tds table tablePlotter = TablePlotter
(tds, 0, 10) #tds is a dataset defined somewhere else
```

TablePlotter(TableDataset tds, integer xIndex, integer yIndex, boolean initialColumnMode)

A constructor This constructor will initiate an instance of an

TablePlotter with three input, a table, the xIndex and the yIndex.

Arguments

TableDataset **tds** [INPUT, MANDATORY, default=no default value A]

TableDataset to be plotted and viewed. It contains one or more columns.

integer **xIndex** [INPUT, MANDATORY, default=no default value The xIndex]

is an integer which indicates the x column data

integer **yIndex** [INPUT, MANDATORY, default=no default value The]

yIndex is an integer which indicates the y column data

boolean **initialColumnMode** [INPUT, MANDATORY]

Example

create an instance of TablePlotter where column 3 is x and

```
column 10 is y from hersechl.ia.gui.explorer.table import
TablePlotter #import the TablePlotter #plot the 10th column vs
the first column from tds table tablePlotter = TablePlotter
(tds, 0, 10) #tds is a dataset defined somewhere else
```

TablePlotter(TableDataset tds, integer xIndex, integer yIndex, boolean initialColumnMode, boolean dataExtractionAllowed)

A constructor This constructor will initiate an instance of an

TablePlotter with three input, a table, the xIndex and the yIndex.

Arguments

TableDataset **tds** [INPUT, MANDATORY, default=no default value A]

TableDataset to be plotted and viewed. It contains one or more columns.

integer **xIndex** [INPUT, MANDATORY, default=no default value The xIndex]

is an integer which indicates the x column data

integer **yIndex** [INPUT, MANDATORY, default=no default value The]

yIndex is an integer which indicates the y column data

boolean **initialColumnMode** [INPUT, MANDATORY]

boolean **dataExtractionAllowed** [INPUT, MANDATORY]

Example

create an instance of TablePlotter where column 3 is x and

```
column 10 is y from hersechl.ia.gui.explorer.table import
TablePlotter #import the TablePlotter #plot the 10th column vs
```

```
TablePlotter(TableDataset tds, integer xIndex, integer yIndex,
boolean initialColumnMode, boolean dataExtractionAllowed)
```

```
the first column from tds table tablePlotter = TablePlotter
(tds, 0, 10) #tds is a dataset defined somewhere else
```

```
TablePlotter(TableDataset tds, BoolId flags)
```

This constructor will initiate an instance of TablePlotter with

pre selected and de-selected columns. The BoolId flag indicate which rows are selected and which rows are de-selected.

Arguments

TableDataset **tds** [INPUT, MANDATORY, default=no default value The tds]

is the TableDataset to be plotted. It contains one or more columns.

BoolId flags [INPUT, MANDATORY, default=no default value The flags]

specifies which rows are de-selected and which rows are selected

```
TablePlotter(TableDataset tds, BoolId flags, integer xIndex, int
yIndex)
```

A constructor This constructor will initiate an instance of

TablePlotter with specified x and y data.

Arguments

TableDataset **tds** [INPUT, MANDATORY, default=no default value A] TableDataset to plot and view. The tds contains one or more columns.

BoolId flags [INPUT, MANDATORY, default=no default value The flags]

specify which rows are de-selected and selected for all columns

integer **xIndex** [INPUT, MANDATORY, default=no default value The] xIndex specifies the x axis data

int **yIndex** [INPUT, MANDATORY, default=no default value The yIndex] specifies the y axis data

Example

create an TablePlotter object with tds, flags and x and y

```
column indices from hersechl.ia.gui.explorer.table import
TablePlotter #import the TablePlotter #Plot the fourth column
vs the second column tpl = TablePlotter(tds, flags, 1, 3) #tds,
flags are defined somewhere else
```

```
TablePlotter(TableDataset tds, BoolId flags, integer xIndex, int
yIndex, boolean dataExtractionAllowed)
```

A constructor This constructor will initiate an instance of

TablePlotter with specified x and y data.

Arguments

TableDataset **tds** [INPUT, MANDATORY, default=no default value A]

TablePlotter(TableDataset tds, Bool1d flags, integer xIndex, int yIndex, boolean dataExtractionAllowed)

TableDataset to plot and view. The tds contains one or more columns.

Bool1d flags [INPUT, MANDATORY, default=no default value The flags]

specify which rows are de-selected and selected for all columns

integer **xIndex** [INPUT, MANDATORY, default=no default value The]

xIndex specifies the x axis data

int **yIndex** [INPUT, MANDATORY, default=no default value The yIndex]

specifies the y axis data

boolean **dataExtractionAllowed** [INPUT, MANDATORY]

Example

create an TablePlotter object with tds, flags and x and y

```
column indices from hersechl.ia.gui.explorer.table import
TablePlotter #import the TablePlotter #Plot the fourth column
vs the second column tpl = TablePlotter(tds, flags, 1, 3) #tds,
flags are defined somewhere else
```

TablePlotter(TableDataset tds, Bool2d flags, boolean initial-ColumnState)

A constructor This constructor will initiate an instance of

TablePlotter with two inputs, a TableDataset and its flags. The flags tell which data points are selected and de-selected. TablePlotter will display two layer plots and one of which is for selected data (blue color) and the other is for de-selected data (red color).

Arguments

TableDataset **tds** [INPUT, MANDATORY, default=no default value A]

TableDataset to be plotted and viewed. It contains one or more columns.

Bool2d flags [INPUT, MANDATORY, default=no default value A flags to]

indicate which data points are selected and de-selected.

boolean **initialColumnState** [INPUT, MANDATORY, default=no default value. This argument]

indicates the initial column selector's state

Example

create a TablePlotter instance with a tds dataset and flags

```
<pre>
from hersechl.ia.gui.explorer.table import TablePlotter #import TablePlotter
tpl=TablePlotter(tds, flags, TablePlotter.ALL_COLUMNS) #tds and flags are
defined somewhere else
</pre>
```

TablePlotter(TableDataset tds, Bool2d flags, boolean initial-ColumnState, boolean dataExtractionAllowed)

Arguments

TableDataset **tds** [INPUT, MANDATORY, default=no default value A]

TableDataset to be plotted and viewed. It contains one or more columns.

TablePlotter(TableDataset tds, Bool2d flags, boolean initial-ColumnState, boolean dataExtractionAllowed)

Bool2d flags [INPUT, MANDATORY, default=no default value A flags to]

indicate which data points are selected and de-selected.

boolean **initialColumnState** [INPUT, MANDATORY, default=no default value. This argument]

indicates the initial column selector's state

boolean **dataExtractionAllowed** [INPUT, MANDATORY]

Methods

JComponent **getComponent**()

Inherit from explorer This method will return a TablePlotter as a component so that users can plug it in his/her own applications.

Return

JComponent

- return the TablePlotter as a component

Examples

Use TablePlotter as a plug-in

```
<pre>
from herschel.share.component import *
from javax.swing import *
from java.awt import *
from herschel.ia.io.fits import FitsArchive
from herschel.ia.dataset import TableDataset
from herschel.ia.dataset import Product
from herschel.ia.dataset.gui import *
from herschel.ia.gui.explorer.table import TablePlotter
fits=FitsArchive();
#Change to your data path
path=&quot;/home/zhang/Development/data/&quot;;
filename = &quot;/loadCurve.fits&quot;;
#load the table
try:
fits.reader = FitsArchive.HCSS_READER
p = fits.load(path+filename)
except java.io.IOException:
fits.reader = FitsArchive.STANDARD_READER
p = fits.load(path+filename)
#create a TableDataset
table =p.default
#create a container to hold the controls/components
pane=JPanel()
#set the layout, there are many different kinds users can choose according
to his/her need
pane.setLayout(BoxLayout(pane, BoxLayout.Y_AXIS))
#add control one, a lael
title = JLabel(&quot;Display TablePlotter&quot;);
pane.add(title)
#Add TablePlotter to the pane
tablePlotter=TablePlotter(table)
pane.add(tablePlotter.component)
pane.setPreferredSize(Dimension(tablePlotter.component.width,
tablePlotter.component.height))
#add to your application
frame =JFrame(&quot;TablePlotter as Plug-in demo&quot;);
frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE)
```

JComponent `getComponent()`

```
frame.getContentPane().add(pane, BorderLayout.CENTER)
frame.pack
frame.visible=1
</pre>
```

- example 2 from `herschel.ia.gui.explorer.table` import from

```
herschel.share.component import wm=WindowManager.getDefault()
tpl = TablePlotter(tds) wm.addWindow('My Application',
tpl.component, 1)
```

String `getDescription()`**Return**

String

- the description of this class

String `getName()`**Return**

String

- the nam of the application

setObject(Object data)**Argument**

Object **data** [INPUT, MANDATORY, default=no default value If data is a]

TableDataset, it will be processed and plotted. If data is not a TableDataset, nothing will be displayed.

Example

Pass the dataset to TablePlotter instance

```
<pre>
tpl = TablePlotter()
tpl.object = tds #tds defined somewhere else
</pre>
```

LayerStruct `getActiveLayerStruct()`

This is a command line API. It is a getter to get the active

LayerStruct object. Through the active LayerStruct object, extracted table and flags can be retrieved.

Return

LayerStruct

- return the current active LayerStruct

Example

Get the extracted dataset and flags `tpl = TablePlotter(tds)`

```
#tds defined somewhere else extracteDataset =
tpl.activeLayerStruct.extractedTableDataset flags =
```


```
LayerStruct getActiveLayerStruct()
```

```
tpl.activeLayerStruct.flgs
```

History

- 2006-10-06 - first: version of TablePlotter
- 2007-12-21 - Major: GUI changes

2.411. TAN

Full Name:	herschel.ia.numeric.toolbox.basic.Tan
Alias:	TAN
Type:	Java Class - 
Import:	from herschel.ia.numeric.toolbox.basic import Tan

Description

Computes the trigonometric tangent of a number or array

Gives the tangent of a number or array. The input is in radians, and may be of any type. Returns a float array for float input array, complex array for input complex array, and double array for any other numeric array type.

Example

Example 1: Apply TAN on a Float1d

```
x=Float1d([0,0.5])
print TAN(x) # [0.0,0.5463025]
```

API Summary

Jython Syntax

```
<y>=TAN(<x>)
```

Properties

[any type x \[INPUT, MANDATORY, default=no default value\]](#)

[any type y \[OUTPUT, NOT_MANDATORY, default=no default value\]](#)

API details

Properties

any type x [INPUT, MANDATORY, default=no default value]

The input is in radians, and may be of any type.


any type y [OUTPUT, NOT_MANDATORY, default=no default value]

Returns a float array for float input array, complex array for input complex array, and double array for any other numeric array type.

See also

- [ARCTAN](#)
- [COS](#)
- [SINH](#)
- [TANH](#)

2.412. TANH

Full Name:	herschel.ia.numeric.toolbox.basic.TanH
Alias:	TANH
Type:	Java Class - 
Import:	from herschel.ia.numeric.toolbox.basic import TanH

Description

Computes the trigonometric hyperbolic tangent of an number or array

Gives the hyperbolic tangent of a number or array. The input is in radians, and may be of any type. Returns a float array for float input array, complex array for input complex array, and double array for any other numeric array type.

API Summary

Jython Syntax
<code><y>=TANH (<x>)</code>

Properties
<code>any type x [INPUT, MANDATORY, default=no default value]</code>
<code>any type y [OUTPUT, NOT_MANDATORY, default=no default value]</code>

Miscellaneous

Does not work for complex values.

API details

Properties


<code>any type x [INPUT, MANDATORY, default=no default value]</code>
The input is in radians, and may be of any type.

<code>any type y [OUTPUT, NOT_MANDATORY, default=no default value]</code>
Returns a float array for float input array, complex array for input complex array, and double array for any other numeric array type.

See also

- [TAN](#)
- [ARCTAN](#)

2.413. TaskWrapper

Full Name:	herschel.ia.dataflow.TaskWrapper
Type:	Java Class - 
Import:	from herschel.ia.dataflow import TaskWrapper

Description

Wraps a task into a process.

TaskWrapper is a special process which allows reuse of a Task instance within stream type environment. It automatically creates a connector for each parameter of the Task instance (as passed as an arg for the constructor of this class).

In default mode (= state mode) it will enable the user to build up an input array (if present) for each Task parameter. In practice it means it creates (a) an input connector which deals with instances of the array's component type and (b) a control connector allowing the user to select the number (default = 1) of instances to gather in an ArrayList before these can be passed to the tasks input parameter as an array.

I.e. for the input parameters of the type [] this class allows you to maintain a state: in case fifo = true (default) the array(list) as maintained by this object is using the fifo concept for the case the array size = "number as set by the related control". Otherwise (i.e fifo=false) the array will be emptied after the number (as set by the control) is reached. Everytime the array is filled it is passed to the task's parameter, the task's "perform()" method is called and the tasks output(s) are passed to the related output(s) of this TaskWrapper object.

Example

Example 1: how to use a TaskWrapper with AverageTask

```
<pre>
from herschel.ia.dataflow import *
from mytasks import AverageTask
tw = TaskWrapper("avg", AverageTask(), True, True, True)
df = DataFlow("mydf")
df.addProcess(tw)
# The rest is obvious...
</pre>
```

Limitations

no limitation


See also

- [reference](#)

History

- 26-5-2005 JCG: change javadoc format for help support.

2.414. tiledImage

Full Name:	herschel.ia.toolbox.image.TiledImageTask
Alias:	tiledImage
Type:	Java Task - 
Import:	from herschel.ia.toolbox.image import TiledImageTask
Category:	task/image

Description

A Task to convert an image to a TiledImage.

A Task to convert an image to a TiledImage, taking the flag into account. TiledImageTask is a task which returns the TiledImage of an Image.

Examples

Example 1: Return the TiledImage, taking into account the flag.

```
tiledImage(image = im, flag = True)
```

Example 2: Return the TiledImage, not taking into account the flag.

```
tiledImage(image = im, method=CutLevels.MEDIAN_FILTER)
```

API Summary

Jython Syntax

```
TiledImage(image, True)
```

Properties

[Image](#) **image** [INPUT, MANDATORY, default=No default value]

[boolean](#) **flag** [INPUT, MANDATORY, default=true]

[TiledImage](#) **result** [OUTPUT, MANDATORY, default=No default value]

API details

Properties

Image image [INPUT, MANDATORY, default=No default value]

The image to use for returning the TiledImage.


boolean flag [INPUT, MANDATORY, default=true]

True if the flag should be taken into account.

[TiledImage](#) result [OUTPUT, MANDATORY, default=No default value]

The final TiledImage.

2.415. Transform2dTask

Full Name:	herschel.ia.toolbox.image.Transform2dTask
Type:	Java Task - 
Import:	from herschel.ia.toolbox.image import Transform2dTask

Description

An abstract Task for two dimensional transforms.

An abstract Task that calculates the Fourier Transform and the power spectrum.

API Summary


Properties
Numeric2dData image [INPUT, MANDATORY, default=No default value]
Complex2d transform [OUTPUT, MANDATORY, default=No default value]
Double2d spectrum [OUTPUT, MANDATORY, default=No default value]

API details

Properties

Numeric2dData image [INPUT, MANDATORY, default=No default value]
The input image.
Complex2d transform [OUTPUT, MANDATORY, default=No default value]
The transform.
Double2d spectrum [OUTPUT, MANDATORY, default=No default value]
The spectrum.

2.416. translate

Full Name:	herschel.ia.toolbox.image.TranslateTask
Alias:	translate
Type:	Java Task - 
Import:	from herschel.ia.toolbox.image import TranslateTask
Category:	task/image

Description

A Task that translates an image.

A task which translates an Image. The Wcs is also adapted. The image is translated over the given parameters (if ra = 1h00m00s, the image will be moved 1h00m00s to the right. At the left, 0.0's will be added which will be masked out.)

Examples

Example 1: translating an image using image coordinates

```
translate(image = im, x = 5, y = 7)
```

Example 2: translating an image using sky coordinates

```
translate(image = im, ra = 0.03, dec = 0.03)
```

API Summary

Jython Syntax

```
translate()
translate(image, ra, dec)
translate(image, x, y)
```

Properties

[Image](#) **image** [INPUT, MANDATORY, default=No default value]

[double](#) **x** [INPUT, OPTIONAL, default=0.0]

[double](#) **y** [INPUT, OPTIONAL, default=0.0]

[Angle](#) **ra** [INPUT, OPTIONAL, default=Angle(0.0, ANGLE.DEGREES)]

[Angle](#) **dec** [INPUT, OPTIONAL, default=Angle(0.0, ANGLE.DEGREES)]

[Image](#) **translatedImage** [OUTPUT, MANDATORY, default=No default value]

API details


Properties

Image image [INPUT, MANDATORY, default=No default value]

The Image to translate

double x [INPUT, OPTIONAL, default=0.0]
The number of pixels to translate in x-direction
double y [INPUT, OPTIONAL, default=0.0]
The number of pixels to translate in y-direction
Angle ra [INPUT, OPTIONAL, default=Angle(0.0, ANGLE.DEGREES)]
The amount of degrees to move in right ascension
Angle dec [INPUT, OPTIONAL, default=Angle(0.0, ANGLE.DEGREES)]
The amount of degrees to move in declination
Image translatedImage [OUTPUT, MANDATORY, default=No default value]
Result of the translation of the Image

2.417. TranslateTaskSignatureComponent

Full Name:	herschel.ia.toolbox.image.gui.TranslateTaskSignatureComponent
Type:	Java Class - 
Import:	from herschel.ia.toolbox.image.gui import TranslateTaskSignatureComponent

Description

The task dialog for the TranslateTask.

A task dialog to serve as GUI for the TranslateTask.

API Summary

Constructor
TranslateTaskSignatureComponent() The construction of a new TranslateTaskSignatureComponent.
Methods
check() Validates the current modifications.
clear() Clears the current modifications.
getParameters() Returns a map of parameters and modifiers storing the values selected by the user.
setVariableSelection(VariableSelection selection) Setting the variable selection.
setSignature(SignatureApi sig) Setting the signature.
JComponent getComponent() Returns the component.

API details

Constructor


<code>TranslateTaskSignatureComponent()</code>
The construction of a new TranslateTaskSignatureComponent.
The construction of a new TranslateTaskSignatureComponent.

Methods

<code>check()</code>
Validates the current modifications.
Validates the current modifications for this TranslateTaskSignatureComponent.

clear()
<p>Clears the current modifications.</p> <p>Clears the current modifications for this TranslateTaskSignatureComponentt.</p>
getParameters()
<p>Returns a map of parameters and modifiers storing the values selected by the user.</p> <p>Returns a map of parameters and modifiers storing the values selected by the user for this TranslateTaskSignatureComponent.</p>
setVariableSelection(VariableSelection selection)
<p>Setting the variable selection.</p> <p>Sets the given selection as the variable selection for this TranslateTaskSignatureComponentt.</p> <p>Argument</p> <p>VariableSelection selection [INPUT, MANDATORY]</p>
setSignature(SignatureApi sig)
<p>Setting the signature.</p> <p>Sets the given signature as the signature for this TranslateTaskSignatureComponentt.</p> <p>Argument</p> <p>SignatureApi sig [INPUT, MANDATORY]</p>
JComponent getComponent()
<p>Returns the component.</p> <p>Returns the component for this TranslateTaskSignatureComponent.</p> <p>Return</p> <p>JComponent</p> <p>Returns the component for this TranslateTaskSignatureComponent.</p>

2.418. TRANSPOSE

Full Name:	herschel.ia.numeric.toolbox.matrix.MatrixTranspose
Alias:	TRANSPOSE
Type:	Java Class - 
Import:	from herschel.ia.numeric.toolbox.matrix import MatrixTranspose

Description

Transposes a matrix.

Example

Example 1: Transposing a 2D array

```
x=Int2d([ [1,2],[3,4],[5,6] -])
print TRANSPOSE(x) # [ [1,3,5],[2,4,6] -]
```

API Summary

Jython Syntax

```
<y>=TRANSPOSE(<x>)
```

Property

[Array2dData](#) **x** [INPUT, MANDATORY, default=no default value]

Miscellaneous

TRANSPOSE is an alias for MatrixTranspose.FUNCTION


API details

Property

[Array2dData](#) **x** [INPUT, MANDATORY, default=no default value]

Input must be a 2d array as defined by the numeric library.

2.419. transpose

Full Name:	herschel.ia.toolbox.image.TransposeTask
Alias:	transpose
Type:	Java Task - 
Import:	from herschel.ia.toolbox.image import TransposeTask
Category:	task/image

Description

A Task that transposes an image.

A task which transposes an image. The Wcs is also adapted. The following types of are possible :

- FLIP_VERTICAL : The image is flipped upside-down
- FLIP_HORIZONTAL : The image is flipped left-right
- FLIP_DIAGONAL : The image is mirrored around the diagonal
- FLIP_ANTIDIAGONAL : The image is mirrored around the antidiagonal
- ROTATE_90 : The image is rotated 90 degrees
- ROTATE_180 : The image is rotated 180 degrees
- ROTATE_270 : The mirror is rotated 270 degrees

Examples

Example 1: Flipping an image horizontal

```
transpose(image = im, type = TransposeTask.FLIP_HORIZONTAL)
```

Example 2: Flipping an image antidiagonal

```
transpose(im, TransposeTask.FLIP_ANTIDIAGONAL)
```

API Summary

Jython Syntax

```
transpose()  
transpose(image, TransposeTask.FLIP_VERTICAL)
```

Properties

[Image](#) **image** [INPUT, MANDATORY, default=No default value]

[TransposeType](#) **type** [INPUT, MANDATORY, default=Transpose.FLIP_VERTICAL]


[Image](#) **transposedImage** [OUTPUT, MANDATORY, default=No default value]

API details

Properties

Image image [INPUT, MANDATORY, default=No default value]
The original image.
TransposeType type [INPUT, MANDATORY, default=Transpose.FLIP_VERTICAL]
The type of transposition to apply (FLIP_VERTICAL, FLIP_HORIZONTAL, FLIP_DIAGONAL, FLIP_ANTIDIAGONAL, ROTATE_90, ROTATE_180 or ROTATE_270).
Image transposedImage [OUTPUT, MANDATORY, default=No default value]
The transposed image.

2.420. TransposeTaskSignatureComponent

Full Name:	herschel.ia.toolbox.image.gui.TransposeTaskSignatureComponent
Type:	Java Class - 
Import:	from herschel.ia.toolbox.image.gui import TransposeTaskSignatureComponent

Description

The task dialog for the TransposeTask.

A task dialog to serve as GUI for the TransposeTask.

API Summary

Constructor
TransposeTaskSignatureComponent() The construction of a new TransposeTaskSignatureComponent.
Methods
check() Validates the current modifications.
clear() Clears the current modifications.
getParameters() Returns a map of parameters and modifiers storing the values selected by the user.
setVariableSelection(VariableSelection selection) Setting the variable selection.
setSignature(SignatureApi sig) Setting the signature.
JComponent getComponent() Returns the component.

API details

Constructor


<code>TransposeTaskSignatureComponent()</code>
The construction of a new TransposeTaskSignatureComponent.
The construction of a new TransposeTaskSignatureComponent.

Methods

<code>check()</code>
Validates the current modifications.
Validates the current modifications for this TransposeTaskSignatureComponent.

clear()
Clears the current modifications. Clears the current modifications for this TransposeTaskSignatureComponentt.
getParameters()
Returns a map of parameters and modifiers storing the values selected by the user. Returns a map of parameters and modifiers storing the values selected by the user for this TransposeTaskSignatureComponent.
setVariableSelection(VariableSelection selection)
Setting the variable selection. Sets the given selection as the variable selection for this TransposeTaskSignatureComponentt. Argument VariableSelection selection [INPUT, MANDATORY]
setSignature(SignatureApi sig)
Setting the signature. Sets the given signature as the signature for this TransposeTaskSignatureComponentt. Argument SignatureApi sig [INPUT, MANDATORY]
JComponent getComponent()
Returns the component. Returns the component for this TransposeTaskSignatureComponent. Return JComponent Returns the component for this TransposeTaskSignatureComponent.

2.421. TruncatedLinearInterpolator

Full Name:	herschel.ia.toolbox.spectrum.utils.interp.TruncatedLinearInterpolator
Alias:	TruncatedLinearInterpolator
Type:	Java Class - 
Import:	from herschel.ia.toolbox.spectrum.utils.interp import TruncatedLinearInterpolator

Description

Given a set of knots (x/y data), this function computes values at arbitrary positions

by linearly interpolating the y value based on the knot lower and higher than the input x position. This can only be applied to double arrays of rank 1. This form of interpolator always allows extrapolation.

Example

Example 1: Create and apply a TruncatedLinearInterpolator on a DoubleId

```
# create some knots
x=DoubleId.range(10) # [0.0,1.0,2.0,3.0,4.0,5.0,6.0,7.0,8.0,9.0]
f=TruncatedLinearInterpolator(x,SQUARE(x))
u=DoubleId([1,1.5,2,2.5,3,3.5])
print f(u) # [1.0,2.5,4.0,6.5,9.0,12.5]
```

API Summary

Jython Syntax

```
<f>=TruncatedLinearInterpolator(<x>,<y>)
```

Properties

[DoubleId](#) **x** [INPUT, MANDATORY, default=no default value]

[DoubleId](#) **y** [INPUT, NOT_MANDATORY, default=false]

API details

Properties


[DoubleId](#) **x** [INPUT, MANDATORY, default=no default value]

The knots are DoubleId

[DoubleId](#) **y** [INPUT, NOT_MANDATORY, default=false]

The knots are DoubleId

2.422. UNIQ_SORTED

Full Name:	herschel.ia.numeric.toolbox.basic.UniqSorted
Alias:	UNIQ_SORTED
Type:	Java Class - 
Import:	from herschel.ia.numeric.toolbox.basic import UniqSorted

Description

Returns the unique elements of an array.

An index `uniq` is available as a special case of `UNIQ_SORTED` i.e. `UNIQ_SORTED.BY_INDEX` which returns an index array into the original array. `UNIQ_SORTED.COUNT_UNIQ_VALUES` returns the number of unique values in an array; although it works also with unsorted arrays, passing an already sorted array is much more efficient.

Examples

Example 1: Apply UNIQ_SORTED on a Double1d

```
x=Double1d([-1,4,2,7,2,-6,-8,3,2])
print UNIQ_SORTED(SORT(x))           # [-8.0,-6.0,-1.0,2.0,3.0,4.0,7.0]
```

Example 2: Apply UNIQ_SORTED.BY_INDEX

```
x=SORT(Double1d([-1,4,2,7,2,-6,-8,3,2]))
print UNIQ_SORTED.BY_INDEX(x)       # [0,1,2,3,6,7,8]
```

Example 3: Apply UNIQ_SORTED.COUNT_UNIQ_VALUES

```
x=Double1d([4, 5, 7, 4, 8, 8, 3, 1, 5])
print UNIQ_SORTED.COUNT_UNIQ_VALUES(SORT(x)) # 6
```

API Summary

Jython Syntax

```
<y>=UNIQ_SORTED(<x>)
```

Properties

[any sorted array **x** \[INPUT, MANDATORY, default=no default value\]](#)

[an array **y** \[OUTPUT, MANDATORY, default=no default value\]](#)

API details

Properties


any sorted array **x [INPUT, MANDATORY, default=no default value]**

The input array can only be an array of rank 1

an array **y [OUTPUT, MANDATORY, default=no default value]**

Returns an array containing only the unique elements from the input array.

2.423. UNIQ

Full Name:	herschel.ia.numeric.toolbox.basic.Uniq
Alias:	UNIQ
Type:	Java Class - 
Import:	from herschel.ia.numeric.toolbox.basic import Uniq

Description

Returns the unique elements of an array.

An index uniq is available as a special case of UNIQ i.e. UNIQ.BY_INDEX which returns an index array into the original array. UNIQ.COUNT_UNIQ_VALUES returns the number of unique values in an array; although it works also with unsorted arrays, passing an already sorted array is much more efficient. The UNIQ.IS_UNIQ function, which tests if the given array indeed contains unique elements, is deprecated. It is strongly recommended that you sort the array and use UNIQ_SORTED due to performance issues.

Examples

Example 1: Apply UNIQ on a DoubleId

```
x=DoubleId([-1,4,2,7,2,-6,-8,3,2])
print UNIQ(x) # [-8.0,-6.0,-1.0,2.0,3.0,4.0,7.0]
print UNIQ(SORT(x)) # [-8.0,-6.0,-1.0,2.0,3.0,4.0,7.0]
```

Example 2: Apply UNIQ.BY_INDEX

```
x=DoubleId([-1,4,2,7,2,-6,-8,3,2])
print UNIQ.BY_INDEX(x) # [0,1,2,3,5,6,7]
print UNIQ.BY_INDEX(SORT(x)) # [0,1,2,3,6,7,8] SORT(x) =
[-8.0,-6.0,-1.0,2.0,2.0,2.0,3.0,4.0,7.0]
```

Example 3: Apply UNIQ.COUNT_UNIQ_VALUES

```
x=DoubleId([4, 5, 7, 4, 8, 8, 3, 1, 5])
print UNIQ.COUNT_UNIQ_VALUES(x) # 6
```

Example 4: Apply UNIQ.IS_UNIQ

```
x=DoubleId([-1,4,7,-2,-6,-8,3,2])
print UNIQ.IS_UNIQ(x) # 0 (false) -- Note that this function is
deprecated
print UNIQ.IS_UNIQ(SORT(x)) # 1 (true) -- Note that this function is
deprecated
```

API Summary

Jython Syntax

```
<y>=UNIQ(<x>)
```

Properties

[any sorted/unsorted array x \[INPUT, MANDATORY, default=no default value\]](#)

Properties

[an array y \[OUTPUT, MANDATORY, default=no default value\]](#)

API details

Properties


any sorted/unordered array x [INPUT, MANDATORY, default=no default value]

The input array can only be an array of rank 1

an array y [OUTPUT, MANDATORY, default=no default value]

Returns an array containing only the unique elements from the input array.

2.424. untar

Full Name:	herschel.ia.toolbox.util.UntarTask
Alias:	untar
Type:	Java Task - 
Import:	from herschel.ia.toolbox.util import UntarTask
Category:	task

Description

Untars a (relative path) tar file in a user-chosen directory

Both arguments are mandatory. If the tar holds absolute paths it may fail.

Example

Example 1: Untarring in temp

```
untar("./mytar.tar", -"/tmp")
```

API Summary

Jython Syntax

```
untar(<tar>, <dirout>)
```

Properties

[String tar](#) [INPUT, MANDATORY, default=null]

[String dirout](#) [INPUT, MANDATORY, default=null]

API details

Properties

[String tar](#) [INPUT, MANDATORY, default=null]

Path of the file to be untared


[String dirout](#) [INPUT, MANDATORY, default=null]

Directory (that may not exist yet) where we want to untar the tar file

History

- 2010-05-25 - JDS: first release

2.425. VARIANCE

Full Name:	herschel.ia.numeric.toolbox.basic.Variance
Alias:	VARIANCE
Type:	Java Class - 
Import:	from herschel.ia.numeric.toolbox.basic import Variance

Description

Yields the variance value of the elements in the input array.

The standard deviation is equivalent to $\text{SQRT}(\text{VARIANCE}(x))$.

Example

Example 1: Apply VARIANCE on a Float1d
<pre>x=Float1d([1,3,2,3,4]) print VARIANCE(x) # 1.3</pre>

API Summary

Jython Syntax
<code><y>=VARIANCE (<x>)</code>
Properties
any scalar array x [INPUT, MANDATORY, default=no default value]
double y [OUTPUT, MANDATORY, default=no default value]

API details


Properties

any scalar array x [INPUT, MANDATORY, default=no default value]
The input array integral or floating-point arrays; the former implicitly transformed to a double array.
double y [OUTPUT, MANDATORY, default=no default value]
Returns a double

See also

- [MEAN](#)
- [MEDIAN](#)
- [STDDEV](#)

2.426. VelocityPosMapComputeTask

Full Name:	herschel.ia.toolbox.cube.VelocityPosMapComputeTask
Type:	Java Task - 
Import:	from herschel.ia.toolbox.cube import VelocityPosMapComputeTask

Description

VelocityPosMapComputeTask The task to compute Velocity position map from a Simplecube

API Summary

Properties
simpleCube simplecube [INPUT, MANDATORY, default=no default value]
axis unknown [INPUT, Boolean, default=MANDATORY]
coordSlitArray unknown [INPUT, Double2d, default=no default value]
widthSlit unknown [INPUT, Integer, default=no default value]
referenceLayer unknown [INPUT, Integer, default=no default value]
totalWeight unknown [INPUT, no default value, default=no default value]
velocityMap unknown [OUTPUT, Double3d, default=no default value]
velocityMapAxis unknown [OUTPUT, Double2d, default=no default value]
velocityMapAxisProd unknown [OUTPUT, SimpleImage, default=no default value]
cubeVelocityMap unknown [OUTPUT, SimpleCube, default=no default value]


API details

Properties

simpleCube simplecube [INPUT, MANDATORY, default=no default value]
The spectralCube as a SimpleCube containing the spectral and velocities information
axis unknown [INPUT, Boolean, default=MANDATORY]
Define the type of map to generate axis coordSlitArray widthSlit referenceLayer totalWeight
coordSlitArray unknown [INPUT, Double2d, default=no default value]
new version array of pixel to be read : manage the width of the slit the order of the information is index, X, Y , weight
widthSlit unknown [INPUT, Integer, default=no default value]
width of the slit

referenceLayer unknown [INPUT, Integer, default=no default value]
The layer considered as reference for the computation of the velocities
totalWeight unknown [INPUT, no default value, default=no default value]
The inside total weight of the array of pixel to be read
velocityMap unknown [OUTPUT, Double3d, default=no default value]
The 2D map of velocity
velocityMapAxis unknown [OUTPUT, Double2d, default=no default value]
The map of velocity along the axis to come soon:
velocityMapAxisProd unknown [OUTPUT, SimpleImage, default=no default value]
The map of velocity along the axis, stored in a SimpleCube
cubeVelocityMap unknown [OUTPUT, SimpleCube, default=no default value]
a cube containing the map of velocity at maximum of emission, the dispersion map, an error value ...

2.427. VelocityPosMapPlotting

Full Name:	herschel.ia.gui.cube.VelocityPosMapPlotting
Type:	Java Class - 
Import:	from herschel.ia.gui.cube import VelocityPosMapPlotting

Description

PositionVelocityDiagramPlotting.

A class to deal with PositionVelocityDiagramComputeTask.

API Summary

Constructor	
VelocityPosMapPlotting(boolean useAsComponent, CubeSpectrumAnalysisToolbox toolbox)	Constructor for VelocityPosMapPlotting.
Methods	
setBegin(Double begin)	Sets the begin of the straight line, associated with this
setEnd(Double end)	Sets the end of the straight line, associated with this
ImageFigure getLine()	Returns the straight line (ImageFigure), along which we wish to
ImageFigure setLine(ImageFigure img)	
Double getBegin()	Returns the begin of the drawn line in PixelCoordinates.
Double getEnd()	Returns the end of the drawn line in PixelCoordinates.
DoubleId IntensityVelocityPosMapPlotting()	Returns the intensity of the pixels along the straight line,
updatePlot()	Updates the plot in this ProfilePlotFrame, with the given plot
drawLine()	Draws and returns a new line on the image, we are analysing with
Point2D[] getPlotPoints()	Returns the intensity of the pixels along the straight line,
getCoordPoints()	Returns the intensity of the pixels along the straight line,
String getSkyCoordinates()	Returns the String version of the sky coordinates
clearPlot()	Clears the plot in this VelocityPosMapPlotting.

API details

Constructor

VelocityPosMapPlotting(boolean useAsComponent, [CubeSpectrumAnalysisToolbox](#) toolbox)

Constructor for VelocityPosMapPlotting.

When the new VelocityPosMapPlotting is component-based, a window for plotting the histogram is opened; otherwise nothing will be shown. jparameter useAsComponent Indication whether the new VelocityPosMapPlotting should be component-based.

Arguments

boolean **useAsComponent** [INPUT, MANDATORY]

[CubeSpectrumAnalysisToolbox](#) **toolbox** [INPUT, MANDATORY]

Methods

setBegin([Double](#) begin)

Sets the begin of the straight line, associated with this

Velocity position map to the given point (in UserCoordinates).

Argument

[Double](#) **begin** [INPUT, MANDATORY]

setEnd([Double](#) end)

Sets the end of the straight line, associated with this

Velocity position map to the given point (in UserCoordinates).

Argument

[Double](#) **end** [INPUT, MANDATORY]

[ImageFigure](#) **getLine**()

Returns the straight line (ImageFigure), along which we wish to

create the diagram velocity position based on the Bresenham's Line-Drawing Algorithm

Return

[ImageFigure](#)

The straight line (ImageFigure), along which we wish to create the diagram velocity position based on the Bresenham's Line-Drawing Algorithm

[ImageFigure](#) **setLine**([ImageFigure](#) img)

Argument

[ImageFigure](#) **img** [INPUT, MANDATORY]

Return

[ImageFigure](#)

The straight line (ImageFigure).

Double `getBegin()`

Returns the begin of the drawn line in PixelCoordinates.

Return

[Double](#)

Returns the begin of the drawn line in PixelCoordinates.

Double `getEnd()`

Returns the end of the drawn line in PixelCoordinates.

Return

[Double](#)

Returns the end of the drawn line in PixelCoordinates.

Double1d `IntensityVelocityPosMapPlotting()`

Returns the intensity of the pixels along the straight line, associated with the given Velocity position map.

Return

[Double1d](#)

Returns the intensity of the pixels along the drawn straight line, associated with the given Velocity position map.

updatePlot()

Updates the plot in this ProfilePlotFrame, with the given plot

points, which represent the intensity along the given straight line (CanvasFigure) and updates the SkyCoordinates (if available) and the PixelCoordinates.

drawLine()

Draws and returns a new line on the image, we are analysing with

the ImageAnalysisToolbox, associated with this Velocity position map.

Point2D[] `getPlotPoints()`

Returns the intensity of the pixels along the straight line, associated with the given Velocity position map.

Return

[Point2D\[\]](#)

Returns the intensity of the pixels along the drawn straight line, associated with the given Velocity position map.

getCoordPoints()

Returns the intensity of the pixels along the straight line,

associated with the given Velocity position map.

[String](#) getSkyCoordinates()

Returns the String version of the sky coordinates

(WorldCoordinates) of the begin and end of the drawn straight line.

Return


[String](#)

Returns the String version of the sky coordinates (WorldCoordinates) of the begin and end of the drawn straight line.

clearPlot()

Clears the plot in this VelocityPosMapPlotting.


2.428. VoigtModel

Full Name:	herschel.ia.numeric.toolbox.fit.VoigtModel
Type:	Java Class - 
Import:	from herschel.ia.numeric.toolbox.fit import VoigtModel

Example

Example 1: VoigtModel
<pre>voigt = VoigtModel() voigt.setParameters(DoubleIeld([5, 4, 1, 2] -) -) print voigt(DoubleIeld.range(41 -) -/ 5 -) # from [0,8] # ... fitter etc. see LevenbergMarquardtFitter</pre>

2.429. WcsExplorer

Full Name:	herschel.ia.gui.image.WcsExplorer
Type:	Java Class - 
Import:	from herschel.ia.gui.image import WcsExplorer

Description

An explorer to view a Wcs.

An explorer to view a Wcs.

API Summary

Constructors
WcsExplorer() The construction of a new WcsExplorer.
WcsExplorer(Object object) The construction of a new WcsExplorer associated with the given object.

Methods
String getName() Returns the name.
String getDescription() Returns the description.
boolean canHandle(Class className) Checks whether this WcsExplorer can handle objects of the given class.
setObject(Object object) Sets the object.
Wcs getObject() Returns the object.
Class getVariableType() Returns the expected variable type.
JComponent getComponent() Returns the component that is responsible for displaying the data object.
boolean makeEditorContent() Creates the editor content for this Explorer.
addDataObjectListener(DataObjectListener arg0) Adds the given listener.
removeDataObjectListener(DataObjectListener arg0) Removes the given listener.

API details

Constructors

WcsExplorer()
The construction of a new WcsExplorer.
The construction of a new WcsExplorer.

WcsExplorer(Object object)
The construction of a new WcsExplorer associated with the given object.
The construction of a new WcsExplorer associated with the given object.
Argument
<code>Object object</code> [INPUT, MANDATORY]

Methods

String getName()
Returns the name.
Returns the name for this WcsExplorer.
Return
<code>String</code>
Returns the name for this WcsExplorer.

String getDescription()
Returns the description.
Returns the description for this WcsExplorer.
Return
<code>String</code>
Returns the description for this WcsExplorer.

boolean canHandle(Class className)
Checks whether this WcsExplorer can handle objects of the given class.
Returns true if this WcsExplorer can handle objects of the given class; false otherwise.
Argument
<code>Class className</code> [INPUT, MANDATORY]
Return
<code>boolean</code>
Returns true of this WcsExplorer can handle objects of the given class; false otherwise.

setObject(Object object)

Sets the object.

Sets the object for this WcsExplorer to the given object.

Argument

[Object](#) **object** [INPUT, MANDATORY]

Wcs getObject()

Returns the object.

Returns the object for this WcsExplorer.

Return

[Wcs](#)

Returns the object for this WcsExplorer.

Class getVariableType()

Returns the expected variable type.

Returns the expected variable type for this WcsExplorer.

Return

[Class](#)

Returns the expected variable type for this WcsExplorer.

JComponent getComponent()

Returns the component that is responsible for displaying the data object.

Returns the component that is responsible for displaying the data object for this WcsExplorer.

Return

[JComponent](#)

Returns the component that is responsible for displaying the data object for this WcsExplorer.

boolean makeEditorContent()

Creates the editor content for this Explorer.

Returns true if the editor content for this WcsExplorer could be made; false otherwise.

Return

boolean

Returns true if the editor content for this WcsExplorer could be made; false otherwise.

addDataObjectListener(DataObjectListener arg0)

Adds the given listener.

Adds the given listener to this WcsExplorer to receive data object events from it.

Argument

addDataObjectListener(DataObjectListener arg0)

DataObjectListener arg0 [INPUT, MANDATORY]

removeDataObjectListener(DataObjectListener arg0)
--


Removes the given listener.

Removes the given listener for this WcsExplorer, so that it no longer receives data object events by this explorer.

Argument

DataObjectListener arg0 [INPUT, MANDATORY]

2.430. Wcs

Full Name:	herschel.ia.dataset.image.wcs.Wcs
Type:	Java Class - 
Import:	from herschel.ia.dataset.image.wcs import Wcs
Category:	Image

Description

A class to create a Wcs.

This class creates a Wcs. The Wcs describes the World coordinate system, which is used to convert between image coordinates and world coordinates.

Example

Example 1: A basic example on how to create a Wcs

```
wcs = Wcs(crval1=30.0, crval2=-89.50, crpix1=109, crpix2=109)
```

API Summary

Constructors	
Wcs()	The default constructor.
Wcs(int naxis)	Constructor for 2 or 3 dimensional Wcs.
Wcs(Wcs orig)	The copy constructor.
Methods	
Wcs copy()	Returns a copy of the Wcs object.
setParameter(String paramname, Object param, String description)	Adds a new parameter to the Wcs.
isCompleteWcs()	Checks whether the Wcs is complete.
Object getParameter(String paramname)	Returns a parameter from the Wcs.
Object[] getParameters()	Returns all parameters.
removeParameter(String paramname)	Removes a parameter.
setNAxis(int naxis)	Sets the number of axes.
int getNAxis()	Returns the number of axes.

Methods
setImageIndex(DoubleId index, Unit unit) Sets the image index for a non-equidistant 3rd dimension.
boolean hasImageIndex() Checks the image index
TableDataset getImageIndex() Returns the image index.
boolean isEquidistantInZ() Checks whether the Wcs is equidistant in the 3rd dimension.
setCrval1(double crval1) Sets crval1.
double getCrval1() Returns crval1
setCrval2(double crval2) Sets crval2.
double getCrval2() Returns crval2
setCrval3(double crval3) Sets crval3.
double getCrval3() Returns crval3
setCrpix1(double crpix1) Sets crpix1.
double getCrpix1() Returns crpix1
setCrpix2(double crpix2) Sets crpix2.
double getCrpix2() Returns crpix2
setCrpix3(double crpix3) Sets crpix3.
double getCrpix3() Returns crpix3
setCdelt1(double cdelt1) Sets cdelt1.
double getCdelt1() Returns cdelt1
setCdelt2(double cdelt2) Sets cdelt2.
double getCdelt2() Returns cdelt2
setCdelt3(double cdelt3)

Methods
Sets cdelt3.
double getCdelt3() Returns cdelt3
boolean checkCtypeValidity(String ctype) Checks the validity of the ctype parameter.
setCtype1(String ctype1) Sets ctype1.
String getCtype1() Returns ctype1
setCtype2(String ctype2) Sets ctype2.
String getCtype2() Returns ctype2
setCtype3(String ctype3) Sets ctype3.
String getCtype3() Returns ctype3
setCunit1(String cunit1) Sets cunit1.
String getCunit1() Returns cunit1
setCunit2(String cunit2) Sets cunit2.
String getCunit2() Returns cunit2
boolean hasParameter(String parameter) Checks the availability of a parameter.
setCunit3(String cunit3) Sets cunit3.
String getCunit3() Returns cunit3
setEpoch(double epoch) Sets the epoch.
double getEpoch() Returns the epoch
setEquinox(double equinox) Sets the equinox.
double getEquinox() Returns the equinox
setRadesys(String Radesys) Sets the reference frame.

Methods
String getRadesys() Returns the reference frame
setCd1_1(double cd1_1) Sets the cd1_1 element.
setCd1_2(double cd1_2) Sets the cd1_2 element.
setCd1_3(double cd1_3) Sets the cd1_3 element.
setCd2_1(double cd2_1) Sets the cd2_1 element.
setCd2_2(double cd2_2) Sets the cd2_2 element.
setCd2_3(double cd2_3) Sets the cd2_3 element.
setCd3_1(double cd3_1) Sets the cd3_1 element.
setCd3_2(double cd3_2) Sets the cd3_2 element.
setCd3_3(double cd3_3) Sets the cd3_3 element.
double getCd1_1() Returns cd1_1
double getCd1_2() Returns cd1_2
double getCd2_1() Returns cd2_1
double getCd2_2() Returns cd2_2
double getCd1_3() Returns cd1_3
double getCd2_3() Returns cd2_3
double getCd3_1() Returns cd3_1
double getCd3_2() Returns cd3_2
double getCd3_3() Returns cd3_3
setPc1_1(double pc1_1) Sets the pc1_1 element.
double getPc1_1()

Methods
Returns pc1_1
setPc1_2(double pc1_2) Sets the pc1_2 element.
double getPc1_2() Returns pc1_2
setPc1_3(double pc1_3) Sets the pc1_3 element.
double getPc1_3() Returns pc1_3
setPc2_1(double pc2_1) Sets the pc2_1 element.
double getPc2_1() Returns pc2_1
setPc2_2(double pc2_2) Sets the pc2_2 element.
double getPc2_2() Returns pc2_2
setPc2_3(double pc2_3) Sets the pc2_3 element.
double getPc2_3() Returns pc2_3
setPc3_1(double pc3_1) Sets the pc3_1 element.
double getPc3_1() Returns pc3_1
setPc3_2(double pc3_2) Sets the pc3_2 element.
double getPc3_2() Returns pc3_2
setPc3_3(double pc3_3) Sets the pc3_3 element.
double getPc3_3() Returns pc3_3
setProjection(String projection) Sets the projection.
String getProjection() Returns the projection.
Metadata getMeta() Return the Wcs as metadata.
String toString() Returns a string representation of the Wcs.

Methods
<p>double[] getWorldCoordinates(double row, double column) Returns the world coordinates of the given image coordinates.</p>
<p>double getZCoordinate(int depth) Returns the world coordinates of the given layer.</p>
<p>double[] getPixelCoordinates(double c1, double c2) Returns the pixel coordinates.</p>
<p>double[] getPixelCoordinates(Double p, double[] coordinates) Returns the pixel coordinates.</p>
<p>setCrota2(double crota2) Sets crota2</p>
<p>setNaxis1(int naxis1) Sets naxis1</p>
<p>int getNaxis1() Returns the number of pixels of the first axis.</p>
<p>setNaxis2(int naxis2) Sets naxis2</p>
<p>int getNaxis2() Returns the number of pixels of the second axis.</p>
<p>setNaxis3(int naxis3) Sets naxis3</p>
<p>int getNaxis3() Returns the number of pixels of the third axis.</p>
<p>double getCrota2() Returns crota2.</p>
<p>boolean isValid() Checks the possibility to convert from pixel to world coordinates.</p>
<p>setWavelength(double wavelength) Sets the wavelength in the Wcs.</p>
<p>getWavelength() Returns the wavelength which is stored in the Wcs</p>

API details

Constructors

Wcs ()
<p>The default constructor.</p>
<p>A constructor which creates a standard Wcs object. The standard Wcs sets naxis = 2 (NAXIS = 2)</p>
<p>Example</p> <p>Typical example on how to create a Wcs.</p> <pre>wcs = Wcs(crval1=30.0, crval2=-89.50, crpix1=109, crpix2=109)</pre>

Wcs(int naxis)

Constructor for 2 or 3 dimensional Wcs.

A constructor which creates a Wcs object with the chosen number of axes. The standard Wcs has only parameter naxis set

Argument

int **naxis** [INPUT, MANDATORY]

Example

Typical example on how to create a Wcs.

```
wcs = Wcs(3)
wcs.setCrvall(30.0)
```

Wcs(Wcs orig)

The copy constructor.

Constructor for Wcs A constructor which creates a copy of an existing Wcs object.

Argument

[Wcs](#) **orig** [INPUT, MANDATORY]

Methods

[Wcs](#) copy()

Returns a copy of the Wcs object.

Returns a copy of the Wcs object.

Return

[Wcs](#)

A copy of the Wcs.

setParameter([String](#) paramname, [Object](#) param, [String](#) description)

Adds a new parameter to the Wcs.

Adds a new parameter to the Wcs.

Arguments

[String](#) **paramname** [INPUT, MANDATORY]

[Object](#) **param** [INPUT, MANDATORY]

[String](#) **description** [INPUT, MANDATORY]

isCompleteWcs()

Checks whether the Wcs is complete.

Returns true if the Wcs has enough information to convert to and from World Coordinates.

[Object](#) getParameter([String](#) paramname)

Returns a parameter from the Wcs.

Object `getParameter(String paramname)`

Returns the requested parameter of the Wcs.

Argument

`String paramname` [INPUT, MANDATORY]

Return

`Object`

The requested parameter.

Object[] `getParameters()`

Returns all parameters.

Returns the list of all Wcs parameters.

Return

`Object[]`

The list of all Wcs parameters.

removeParameter(String paramname)

Removes a parameter.

Removes the requested parameter of the Wcs.

Argument

`String paramname` [INPUT, MANDATORY]

setNAxis(int naxis)

Sets the number of axes.

Sets the number of axes of the Wcs.

Argument

`int naxis` [INPUT, MANDATORY]

int `getNAxis()`

Returns the number of axes.

Returns the number of axes of the Wcs.

Return

`int`

The number of axes.

setImageIndex(Doubleld index, Unit unit)

Sets the image index for a non-equidistant 3rd dimension.

Sets the image index for a non-equidistant 3rd dimension.

Arguments

```
setImageIndex(DoubleId index, Unit unit)
```

```
DoubleId index [INPUT, MANDATORY]
```

```
Unit unit [INPUT, MANDATORY]
```

```
boolean hasImageIndex()
```

Checks the image index

Returns true if there is an image index.

Return

boolean

True if there is an image index.

```
TableDataset getImageIndex()
```

Returns the image index.

Returns the image index for a non-equidistant 3rd dimension.

Return

TableDataset

The image index for a non-equidistant 3rd dimension.

```
boolean isEquidistantInZ()
```

Checks whether the Wcs is equidistant in the 3rd dimension.

isEquidistantInZ returns true when the 3rd axis is equidistant. In this case, the crval3, crpix3 and cdelt3 keywords are used. If the 3rd axis is not equidistant, the ImageIndex is used.

Return

boolean

True if the Wcs is equidistant.

```
setCrval1(double crval1)
```

Sets crval1.

Sets the first coordinate of the center.

Argument

```
double crval1 [INPUT, MANDATORY]
```

```
double getCrval1()
```

Returns crval1

Returns the first coordinate of the reference pixel.

Return

double

The first coordinate of the reference pixel.

setCrval2(double crval2)

Sets crval2.

Sets the second coordinate of the center.

Argument

double **crval2** [INPUT, MANDATORY]

double getCrval2()

Returns crval2

Returns the second coordinate of the reference pixel.

Return

double

The second coordinate of the reference pixel.

setCrval3(double crval3)

Sets crval3.

Sets the third coordinate of the center.

Argument

double **crval3** [INPUT, MANDATORY]

double getCrval3()

Returns crval3

Returns the third coordinate of the reference pixel.

Return

double

The third coordinate of the reference pixel.

setCrpix1(double crpix1)

Sets crpix1.

Sets the reference pixel position of axis 1. WARNING: CRPIX1 should be given in x,y coordinates. The standard specifies that the image starts at pixel (1,1) and not at pixel (0,0). So to set the first pixel, you should use 1 as value for crpix1.

Argument

double **crpix1** [INPUT, MANDATORY]

double getCrpix1()

Returns crpix1

Returns the reference pixel position of axis 1.

Return

double

double getCrpix1()

The reference pixel position of axis 1.

setCrpix2(double crpix2)

Sets crpix2.

Sets the reference pixel position of axis 2. WARNING: CRPIX2 should be given in x,y coordinates. The standard specifies that the image starts at pixel (1,1) and not at pixel (0,0). So to set the first pixel, you should use 1 as value for crpix2.

Argument

double **crpix2** [INPUT, MANDATORY]

double getCrpix2()

Returns crpix2

Returns the reference pixel position of axis 2.

Return

double

The reference pixel position of axis 2.

setCrpix3(double crpix3)

Sets crpix3.

Sets the reference pixel position of axis 3.

Argument

double **crpix3** [INPUT, MANDATORY]

double getCrpix3()

Returns crpix3

Returns the reference pixel position of axis 3.

Return

double

The reference pixel position of axis 3.

setCdelt1(double cdelt1)

Sets cdelt1.

Sets the pixel scale of axis 1.

Argument

double **cdelt1** [INPUT, MANDATORY]

double getCdelt1()

Returns cdelt1

Returns the pixel scale of axis 1.

double getCdelt1()
Return double The pixel scale of axis 1.
setCdelt2(double cdelt2)
Sets cdelt2. Sets the pixel scale of axis 2. Argument double cdelt2 [INPUT, MANDATORY]
double getCdelt2()
Returns cdelt2 Returns the pixel scale of axis 2. Return double The pixel scale of axis 2.
setCdelt3(double cdelt3)
Sets cdelt3. Sets the pixel scale of axis 3. Argument double cdelt3 [INPUT, MANDATORY]
double getCdelt3()
Returns cdelt3 Returns the pixel scale of axis 3. Return double The pixel scale of axis 3.
boolean checkCtypeValidity(String ctype)
Checks the validity of the ctype parameter. Checks the validity of the ctype parameter. Argument String ctype [INPUT, MANDATORY] Return boolean true if the ctype parameter is valid.

setCtype1([String](#) ctype1)

Sets ctype1.

Sets the projection type of axis 1.

Argument[String](#) ctype1 [INPUT, MANDATORY][String](#) getCtype1()

Returns ctype1

Returns the projection type of axis 1.

Return[String](#)

The projection type of axis 1.

setCtype2([String](#) ctype2)

Sets ctype2.

Sets the projection type of axis 2.

Argument[String](#) ctype2 [INPUT, MANDATORY][String](#) getCtype2()

Returns ctype2

Returns the projection type of axis 2.

Return[String](#)

The projection type of axis 2.

setCtype3([String](#) ctype3)

Sets ctype3.

Sets the projection type of axis 3.

Argument[String](#) ctype3 [INPUT, MANDATORY][String](#) getCtype3()

Returns ctype3

Returns the projection type of axis 3.

Return[String](#)

The projection type of axis 3.

setCunit1([String](#) cunit1)

Sets cunit1.

Sets the unit of axis 1.

Argument[String](#) cunit1 [INPUT, MANDATORY][String](#) getCunit1()

Returns cunit1

Returns the unit of axis 1.

Return[String](#)

The unit of axis 1.

setCunit2([String](#) cunit2)

Sets cunit2.

Sets the unit of axis 2.

Argument[String](#) cunit2 [INPUT, MANDATORY][String](#) getCunit2()

Returns cunit2

Returns the unit of axis 2.

Return[String](#)

The unit of axis 2.

boolean hasParameter([String](#) parameter)

Checks the availability of a parameter.

Checks whether the parameter is available.

Argument[String](#) parameter [INPUT, MANDATORY]**Return****boolean**

True if the parameter is available.

setCunit3([String](#) cunit3)

Sets cunit3.

Sets the unit of axis 3.

```
setCunit3(String cunit3)
```

Argument

```
String cunit3 [INPUT, MANDATORY]
```

```
String getCunit3()
```

Returns cunit3

Returns the unit of axis 3.

Return

[String](#)

The unit of axis 3.

```
setEpoch(double epoch)
```

Sets the epoch.

Sets the epoch.

Argument

```
double epoch [INPUT, MANDATORY]
```

```
double getEpoch()
```

Returns the epoch

Returns the epoch.

Return

double

The epoch.

```
setEquinox(double equinox)
```

Sets the equinox.

Sets the equinox.

Argument

```
double equinox [INPUT, MANDATORY]
```

```
double getEquinox()
```

Returns the equinox

Returns the equinox.

Return

double

The equinox.

```
setRadesys(String Radesys)
```

Sets the reference frame.

setRadesys([String](#) Radesys)

Sets the reference frame.

Argument

[String](#) Radesys [INPUT, MANDATORY]

[String](#) getRadesys()

Returns the reference frame

Returns the reference frame.

Return

[String](#)

The reference frame.

setCd1_1(double cd1_1)

Sets the cd1_1 element.

Sets element (1, 1) of the corrected CD matrix $CD_{i,j}$.

Argument

double cd1_1 [INPUT, MANDATORY]

setCd1_2(double cd1_2)

Sets the cd1_2 element.

Sets element (1, 2) of the corrected CD matrix $CD_{i,j}$.

Argument

double cd1_2 [INPUT, MANDATORY]

setCd1_3(double cd1_3)

Sets the cd1_3 element.

Sets element (1, 3) of the corrected CD matrix $CD_{i,j}$.

Argument

double cd1_3 [INPUT, MANDATORY]

setCd2_1(double cd2_1)

Sets the cd2_1 element.

Sets element (2, 1) of the corrected CD matrix $CD_{i,j}$.

Argument

double cd2_1 [INPUT, MANDATORY]

setCd2_2(double cd2_2)

Sets the cd2_2 element.

Sets element (2, 2) of the corrected CD matrix $CD_{i,j}$.

setCd2_2(double cd2_2)
Argument double cd2_2 [INPUT, MANDATORY]
setCd2_3(double cd2_3)
Sets the cd2_3 element. Sets element (2, 3) of the corrected CD matrix CDi_j.
Argument double cd2_3 [INPUT, MANDATORY]
setCd3_1(double cd3_1)
Sets the cd3_1 element. Sets element (3, 1) of the corrected CD matrix CDi_j.
Argument double cd3_1 [INPUT, MANDATORY]
setCd3_2(double cd3_2)
Sets the cd3_2 element. Sets element (3, 2) of the corrected CD matrix CDi_j.
Argument double cd3_2 [INPUT, MANDATORY]
setCd3_3(double cd3_3)
Sets the cd3_3 element. Sets element (3, 3) of the corrected CD matrix CDi_j.
Argument double cd3_3 [INPUT, MANDATORY]
double getCd1_1()
Returns cd1_1 Returns element (1,1) of the corrected CD matrix CDi_j.
Return double Element (1,1) of the corrected CD matrix CDi_j.
double getCd1_2()
Returns cd1_2 Returns element (1,2) of the corrected CD matrix CDi_j.
Return double

double getCd1_2()

Element (1,2) of the corrected CD matrix CDi_j.

double getCd2_1()

Returns cd2_1

Returns element (2,1) of the corrected CD matrix CDi_j.

Return**double**

Element (2,1) of the corrected CD matrix CDi_j.

double getCd2_2()

Returns cd2_2

Returns element (2,2) of the corrected CD matrix CDi_j.

Return**double**

Element (2,2) of the corrected CD matrix CDi_j.

double getCd1_3()

Returns cd1_3

Returns element (1,3) of the corrected CD matrix CDi_j.

Return**double**

Element (1,3) of the corrected CD matrix CDi_j.

double getCd2_3()

Returns cd2_3

Returns element (2,3) of the corrected CD matrix CDi_j.

Return**double**

Element (2,3) of the corrected CD matrix CDi_j.

double getCd3_1()

Returns cd3_1

Returns element (3,1) of the corrected CD matrix CDi_j.

Return**double**

Element (3,1) of the corrected CD matrix CDi_j.

double getCd3_2()

Returns cd3_2

Returns element (3,2) of the corrected CD matrix CDi_j.

Return

double

Element (3,2) of the corrected CD matrix CDi_j.

double getCd3_3()

Returns cd3_3

Returns element (3,3) of the corrected CD matrix CDi_j.

Return

double

Element (3,3) of the corrected CD matrix CDi_j.

setPc1_1(double pc1_1)

Sets the pc1_1 element.

Sets element (1, 1) of the linear transformation matrix PCi_j..

Argument

double **pc1_1** [INPUT, MANDATORY]

double getPc1_1()

Returns pc1_1

Returns element (1,1) of the linear transformation matrix PCi_j.

Return

double

Element (1,1) of the linear transformation matrix PCi_j.

setPc1_2(double pc1_2)

Sets the pc1_2 element.

Sets element (1, 2) of the linear transformation matrix PCi_j..

Argument

double **pc1_2** [INPUT, MANDATORY]

double getPc1_2()

Returns pc1_2

Returns element (1,2) of the linear transformation matrix PCi_j.

Return

double

double getPc1_2()

Element (1,2) of the linear transformation matrix PCi_j.

setPc1_3(double pc1_3)

Sets the pc1_3 element.

Sets element (1, 3) of the linear transformation matrix PCi_j..

Argumentdouble **pc1_3** [INPUT, MANDATORY]**double getPc1_3()**

Returns pc1_3

Returns element (1,3) of the linear transformation matrix PCi_j.

Return**double**

Element (1,3) of the linear transformation matrix PCi_j.

setPc2_1(double pc2_1)

Sets the pc2_1 element.

Sets element (2, 1) of the linear transformation matrix PCi_j..

Argumentdouble **pc2_1** [INPUT, MANDATORY]**double getPc2_1()**

Returns pc2_1

Returns element (2,1) of the linear transformation matrix PCi_j.

Return**double**

Element (2,1) of the linear transformation matrix PCi_j.

setPc2_2(double pc2_2)

Sets the pc2_2 element.

Sets element (2, 2) of the linear transformation matrix PCi_j..

Argumentdouble **pc2_2** [INPUT, MANDATORY]**double getPc2_2()**

Returns pc2_2

Returns element (2,2) of the linear transformation matrix PCi_j.

```
double getPc2_2()
```

Return

double

Element (2,2) of the linear transformation matrix PCi_j.

```
setPc2_3(double pc2_3)
```

Sets the pc2_3 element.

Sets element (2, 3) of the linear transformation matrix PCi_j..

Argument

double pc2_3 [INPUT, MANDATORY]

```
double getPc2_3()
```

Returns pc2_3

Returns element (2,3) of the linear transformation matrix PCi_j.

Return

double

Element (2,3) of the linear transformation matrix PCi_j.

```
setPc3_1(double pc3_1)
```

Sets the pc3_1 element.

Sets element (3, 1) of the linear transformation matrix PCi_j..

Argument

double pc3_1 [INPUT, MANDATORY]

```
double getPc3_1()
```

Returns pc3_1

Returns element (3,1) of the linear transformation matrix PCi_j.

Return

double

Element (3,1) of the linear transformation matrix PCi_j.

```
setPc3_2(double pc3_2)
```

Sets the pc3_2 element.

Sets element (3, 2) of the linear transformation matrix PCi_j..

Argument

double pc3_2 [INPUT, MANDATORY]

```
double getPc3_2()
```

Returns pc3_2

double <code>getPc3_2()</code>
Returns element (3,2) of the linear transformation matrix PCi_j.
Return
double
Element (3,2) of the linear transformation matrix PCi_j.
setPc3_3(double pc3_3)
Sets the pc3_3 element.
Sets element (3, 3) of the linear transformation matrix PCi_j..
Argument
double pc3_3 [INPUT, MANDATORY]
double <code>getPc3_3()</code>
Returns pc3_3
Returns element (3,3) of the linear transformation matrix PCi_j.
Return
double
Element (3,3) of the linear transformation matrix PCi_j.
setProjection(String projection)
Sets the projection.
Sets the projection type.
Argument
String projection [INPUT, MANDATORY]
String <code>getProjection()</code>
Returns the projection.
Returns the projection type.
Return
String
The projection type.
MetaData <code>getMeta()</code>
Return the Wcs as metadata.
Returns the Wcs as metadata.
Return
MetaData
The Wcs as metadata.

<code>String toString()</code>
<p>Returns a string representation of the Wcs.</p> <p>Returns a string representation of the values of the Wcs object. The format of this string is undefined and subject to change.</p> <p>Return</p> <p><code>String</code></p> <p>a string representation of the values of the Wcs object.</p>
<code>double[] getWorldCoordinates(double row, double column)</code>
<p>Returns the world coordinates of the given image coordinates.</p> <p>Returns the world coordinates when the image coordinates are given.</p> <p>Arguments</p> <p><code>double row</code> [INPUT, MANDATORY]</p> <p><code>double column</code> [INPUT, MANDATORY]</p> <p>Return</p> <p><code>double[]</code></p> <p>The corresponding world coordinates as a 2 dimensional array. When the ctype1 and ctype2 keywords of the wcs are defined as "RA--TAN" and "DEC--TAN", the first coordinates describes the right ascension and the second the declination.</p>
<code>double getZCoordinate(int depth)</code>
<p>Returns the world coordinates of the given layer.</p> <p>Returns the world coordinates of the given layer.</p> <p>Argument</p> <p><code>int depth</code> [INPUT, MANDATORY]</p> <p>Return</p> <p><code>double</code></p> <p>The corresponding world coordinates of the Z axis.</p>
<code>double[] getPixelCoordinates(double c1, double c2)</code>
<p>Returns the pixel coordinates.</p> <p>Returns the pixelCoordinates of the given SkyCoordinates. The pixel- coordinates are the row and the column of the image!</p> <p>Arguments</p> <p><code>double c1</code> [INPUT, MANDATORY]</p> <p><code>double c2</code> [INPUT, MANDATORY]</p> <p>Return</p> <p><code>double[]</code></p> <p>A 2-dimensional array of doubles describing the pixel coordinates of the given world coordinates.</p>

```
double[] getPixelCoordinates(Double p, double[] coordinates)
```

Returns the pixel coordinates.

Returns the pixelCoordinates of the given SkyCoordinates. The pixel- coordinates are the row and the column of the image!

Arguments

[Double](#) p [INPUT, MANDATORY]

double[] **coordinates** [INPUT, MANDATORY]

Return

double[]

A 2-dimensional array of doubles describing the pixel coordinates of the given world coordinates.

```
setCrota2(double crota2)
```

Sets crota2

Sets the rotation angle in degrees.

Argument

double **crota2** [INPUT, MANDATORY]

```
setNaxis1(int naxis1)
```

Sets naxis1

Sets the number of pixels of the first axis.

Argument

int **naxis1** [INPUT, MANDATORY]

```
int getNaxis1()
```

Returns the number of pixels of the first axis.

Returns the number of pixels of the first axis of the Wcs.

Return

int

The number of pixels of the first axis.

```
setNaxis2(int naxis2)
```

Sets naxis2

Sets the number of pixels of the second axis.

Argument

int **naxis2** [INPUT, MANDATORY]

```
int getNaxis2()
```

Returns the number of pixels of the second axis.

Returns the number of pixels of the second axis of the Wcs.

```
int getNaxis2()
```

Return

```
int
```

The number of pixels of the second axis.

```
setNaxis3(int naxis3)
```

Sets naxis3

Sets the number of pixels of the third axis.

Argument

```
int naxis3 [INPUT, MANDATORY]
```

```
int getNaxis3()
```

Returns the number of pixels of the third axis.

Returns the number of pixels of the third axis of the Wcs.

Return

```
int
```

The number of pixels of the third axis.

```
double getCrota2()
```

Returns crota2.

Returns the rotation angle in degrees.

Return

```
double
```

The rotation angle in degrees.

```
boolean isValid()
```

Checks the possibility to convert from pixel to world coordinates.

Checks if the conversion from pixel to world coordinates is possible and returns true if this is the case.

Return

```
boolean
```

true is the conversion from pixel to world coordinates is possible.

```
setWavelength(double wavelength)
```

Sets the wavelength in the Wcs.


Sets the wavelength in the Wcs (in micrometers). The WAVELNTH and the WAVEUNIT parameter are set.

Argument

```
double wavelength [INPUT, MANDATORY]
```

getWavelength()
Returns the wavelength which is stored in the Wcs
Returns the wavenlength in micrometer.

2.431. WeightedMean

Full Name:	herschel.ia.numeric.toolbox.basic.WeightedMean
Type:	Java Class - 
Import:	from herschel.ia.numeric.toolbox.basic import WeightedMean

Description

This class calculates the quadratically weighted mean and the uncertainty of it, based on

the assumption that the distribution of the errors is Gaussian, of an array of numbers and its corresponding uncertainties in the same units.

Formula:

```

Let -"x" be a vector of n numbers for input and -"dx" be a vector of -"n" numbers
that represent the uncertainties of -"x" in the same units.
Let -"x[i]" be the i'st element of a vector -"x".
Let sum() be the sum over all indices i from 1 to n.
The weighted mean wmean is then:
wmean = sum( x[i]/(dx[i]^2) -) -/ sum( 1/(dx[i]^2) )
Standard deviation of the distribution (old name ewmean):
errorDistr = sqrt( sum( (x[i]-wmean)^2/(dx[i]^2) -) -/ sum( 1/(dx[i]^2) -) -)
Standard deviation of the mean:
errorMean = sqrt( (1/sum(1/dx[i]^2)) * (1/(N-1)) * sum((x[i]-wmean)^2/(dx[i]\
]^2)) -)
Standard deviation of the mean for trusted errors:
errorMeanTrusted = sqrt(1/sum(1/dx[i]^2))

```

NaN numbers: if ignoreNaN is set to 'true', a NaN value, either in values or uncertainties, is not used in the procedure. If ignoreNaN if set to 'false' (default value) and a NaN is found, a NaN value is returned. NOTE: NaN values can be used for double, float and complex arrays only.

Example

Example 1: Untitled
<pre> values = Double1d([1,3,5]) errors = Double1d([0.1,0.2,0.3]) wmean = WeightedMean(values,errors) print wmean.mean() print wmean.errorDistr() print wmean.errorMean() print wmean.errorMeanTrusted() values = Complex1d([1+1j,3+1j,5+1j]) errors = Complex1d([0.1+1j,0.2+1j,0.3+1j]) wmean = WeightedMean(values,errors) print wmean.mean() print wmean.errorDistr() print wmean.errorMean() print wmean.errorMeanTrusted() </pre> </pre>

API Summary

Jython Syntax
<pre> wm = WeightedMean(<values>, <errors>, <ignoreNaN>) wm.mean() </pre>

Jython Syntax

```

wm.errorDistr()
wm.errorMean()
wm.errorMeanTrusted()

```

Properties

[number array values](#) [INPUT, MANDATORY, default=no default value]

[number array errors](#) [INPUT, MANDATORY, default=no default value]

[boolean ignoreNaN](#) [INPUT, OPTIONAL, default=false]

API details

Properties

number array values [INPUT, MANDATORY, default=no default value]

The input numbers array can be integer, long, float, double and complex one dimensional array.

number array errors [INPUT, MANDATORY, default=no default value]

The input uncertainties array can be integer, long, float, double and complex one dimensional array.

boolean ignoreNaN [INPUT, OPTIONAL, default=false]

If 'true' a NaN value will be ignored. By default is 'false'.

See also

- [WeightedMean](#)