

User's Reference Manual

Herschel Data Processing

version 0.2.401 , Document Number: HERSCHEL-HSC-DOC-0935
21 November 2008



User's Reference Manual: Herschel Data Processing

Table of Contents

I. Categorized view of Commands	x
II. The Herschel Common Science System and Data Processing (DP)	xii
II.1. Brief Overview	xii
II.2. Availability of DP and Operating Systems	xii
II.3. Related Documentation	xii
II.4. Versioning	xiii
1. Numeric	1
1.1. Introduction	1
1.2. Quick Start	1
1.3. Import statements	1
1.4. Arrays	2
1.5. Toolboxes	13
2. Product Access Layer	19
2.1. Introduction	19
2.2. Example Usage	19
2.3. Querying	20
2.4. Product Pools	22
2.5. What is a URN ?	22
2.6. Context Products	23
2.7. Deep Copy or Cloning of Products	23
2.8. Interface Definitions	23
3. DP Commands	25
3.1. Introduction	25
3.2. ABS	26
3.3. AddSpectrum	27
3.4. ALL	30
3.5. AllPresent	31
3.6. AmoebaFitter	33
3.7. AnnotationToolbox	34
3.8. AnnularSkyAperturePhotometryExplorer	36
3.9. AnnularSkyAperturePhotometryPanel	38
3.10. AnnularSkyAperturePhotometryProduct	40
3.11. AnnularSkyAperturePhotometryTask	43
3.12. ANY	45
3.13. AnyPresent	47
3.14. AperturePhotometryExplorer	49
3.15. AperturePhotometryPanel	53
3.16. AperturePhotometryProduct	59
3.17. AperturePhotometryTask	67
3.18. ARCCOS	69
3.19. ARCSIN	70
3.20. ARCTAN	71
3.21. asciiTableReader	72
3.22. asciiTableWriter	76
3.23. AttribQuery	78
3.24. AutomaticContourTask	79
3.25. AverageFilteringTask	80
3.26. AverageNoiseTask	81
3.27. AverageSpectrum	82
3.28. bg	85
3.29. Bilinear	87
3.30. BinCentres	89
3.31. Bool1d	91
3.32. Bool2d	92
3.33. Bool3d	93

3.34. Bool4d	94
3.35. Bool5d	95
3.36. BoxCarFilter	96
3.37. Byte1d	98
3.38. Byte2d	99
3.39. Byte3d	100
3.40. Byte4d	101
3.41. Byte5d	102
3.42. CachedPool	103
3.43. CEIL	104
3.44. ChannelMapPlotting	105
3.45. CircleHistogramExplorer	108
3.46. CircleHistogramPanel	110
3.47. CircleHistogramProduct	111
3.48. CircleHistogramTask	114
3.49. clampTask	116
3.50. clear	118
3.51. clear	120
3.52. Complex1d	122
3.53. Complex2d	123
3.54. Complex3d	124
3.55. Complex4d	125
3.56. Complex5d	126
3.57. compose	127
3.58. compute	129
3.59. CONCATENATE	131
3.60. Condense	132
3.61. ConnectorBox	136
3.62. Contour	137
3.63. ContourLevel	138
3.64. ContourPlotting	139
3.65. ContourTask	142
3.66. Convolution	143
3.67. Correlate	145
3.68. CorrelateMatrix	147
3.69. COS	148
3.70. COSH	150
3.71. cropTask	151
3.72. CubeSpectrumAnalysis	153
3.73. CubeSpectrumAnalysisToolbox	155
3.74. CubicSplineInterpolator	167
3.75. cutLevels	169
3.76. database	171
3.77. DataFlow	173
3.78. DataFlowManager	174
3.79. DbPool	175
3.80. DETERMINANT	177
3.81. DFT2dTask	178
3.82. Display	179
3.83. displaylog	209
3.84. displayQCLog	211
3.85. displayQCLog	212
3.86. DivideSpectrum	213
3.87. Double1d	216
3.88. Double2d	217
3.89. Double3d	218
3.90. Double4d	219
3.91. Double5d	220

3.92. EigenvalueDecomposition	221
3.93. EllipseHistogramExplorer	222
3.94. EllipseHistogramPanel	224
3.95. EllipseHistogramProduct	225
3.96. EllipseHistogramTask	228
3.97. Erfc	230
3.98. Erf	231
3.99. EXP10	232
3.100. EXP	233
3.101. ExpN	234
3.102. ExtractFreqRanges	235
3.103. ExtractRegionPixelSpectrumTask	237
3.104. ExtractSinglePixelSpectrumTask	239
3.105. FFT2dTask	241
3.106. FFT	242
3.107. FilteringTask	244
3.108. FilterNoiseTask	245
3.109. FitsArchive	246
3.110. FitterFunction	250
3.111. Fitter	252
3.112. FitterTask	253
3.113. FixedMask	257
3.114. Flag	258
3.115. FlagSaturatedPixelsCubeTask	264
3.116. FlagSaturatedPixelsTask	265
3.117. Float1d	266
3.118. Float2d	267
3.119. Float3d	268
3.120. Float4d	269
3.121. Float5d	270
3.122. FLOOR	271
3.123. FullQuery	272
3.124. GAMMALN	273
3.125. GammaP	274
3.126. GammaQ	275
3.127. GaussFitTask	276
3.128. GaussianFilter	280
3.129. HAMMING	282
3.130. HANNING	285
3.131. HduHeaders	288
3.132. help	290
3.133. Histogram	292
3.134. Histogram	296
3.135. HistogramPanel	298
3.136. HistogramTask	303
3.137. IaProcessImpl	304
3.138. IFFT	306
3.139. ImageAbsTask	308
3.140. ImageAddTask	309
3.141. ImageAnalysis	310
3.142. ImageAnalysisToolbox	312
3.143. ImageArithmeticsTask	319
3.144. ImageAxis	320
3.145. ImageCeilTask	327
3.146. ImageContour	328
3.147. ImageDivideTask	331
3.148. ImageExp10Task	332
3.149. ImageExpNTask	333

3.150. ImageExpTask	334
3.151. ImageFloorTask	335
3.152. ImageHistogramExplorer	336
3.153. ImageHistogramProduct	339
3.154. ImageHistogramTask	342
3.155. ImageLog10Task	343
3.156. ImageLogNTask	344
3.157. ImageLogTask	345
3.158. ImageModuloTask	346
3.159. ImageMultiplyTask	347
3.160. ImagePowerTask	348
3.161. ImageRoundTask	349
3.162. imageSaverTask	350
3.163. ImageSqrtTask	351
3.164. ImageSquareTask	352
3.165. ImageSubtractTask	353
3.166. importCubeTask	354
3.167. importImageTask	355
3.168. info	356
3.169. Int1d	358
3.170. Int2d	359
3.171. Int3d	360
3.172. Int4d	361
3.173. Int5d	362
3.174. Integrator	363
3.175. InverseDFT2dTask	365
3.176. InverseFFT2dTask	366
3.177. INVERSE	367
3.178. IS_FINITE	368
3.179. IS_INFINITE	369
3.180. IS_NAN	370
3.181. KURTOSIS	371
3.182. LevenbergMarquardtFitter	373
3.183. LinearInterpolator	374
3.184. ListContext	376
3.185. localStoreWriter	378
3.186. LOG10	379
3.187. LOG	380
3.188. LogN	381
3.189. Long1d	383
3.190. Long2d	384
3.191. Long3d	385
3.192. Long4d	386
3.193. Long5d	387
3.194. ManualContourTask	388
3.195. MapContext	389
3.196. MapMakingTask	391
3.197. MATMUL	392
3.198. MatrixMultiply	393
3.199. MatrixSolve	395
3.200. MAX	396
3.201. MEAN	398
3.202. MEDIANDEV	399
3.203. MedianFilteringTask	400
3.204. MEDIAN	401
3.205. MedianNoiseTask	402
3.206. MetaQuery	403
3.207. MIN	404




3.208. MosaicTask	406
3.209. MultiplySpectrum	407
3.210. NaiveMapMakingTask	410
3.211. NearestNeighborInterpolator	411
3.212. NoiseTask	413
3.213. Normalize	414
3.214. NotPresent	417
3.215. NumberedDataset	419
3.216. ObservationContext	420
3.217. OpDayGenerator	422
3.218. PackedMask	423
3.219. PacketSequence	424
3.220. pause	425
3.221. pointHistoryDisplay	426
3.222. Polygon	428
3.223. PolygonHistogramExplorer	430
3.224. PolygonHistogramPanel	432
3.225. PolygonHistogramProduct	433
3.226. PolygonHistogramTask	435
3.227. Polynomial	437
3.228. poolDataReader	439
3.229. poolDataWriter	441
3.230. PowerSpectrum	444
3.231. Pow	445
3.232. ProcessDistributor	446
3.233. PRODUCT	448
3.234. ProductRef	450
3.235. ProductStorage	451
3.236. ProfileExplorer	456
3.237. Profile	459
3.238. ProfilePanel	462
3.239. ProfilePlotting	466
3.240. ProfileTask	470
3.241. Query	472
3.242. RadialVelocity	473
3.243. RandomGauss	474
3.244. RandomPoisson	475
3.245. RandomUniform	476
3.246. RangeExtractionGui	478
3.247. RangeExtractionTask	480
3.248. RealFunction	481
3.249. REBIN	482
3.250. RectangleHistogramExplorer	484
3.251. RectangleHistogramPanel	486
3.252. RectangleHistogramProduct	487
3.253. RectangleHistogramTask	490
3.254. RectangularSkyAperturePhotometryExplorer	492
3.255. RectangularSkyAperturePhotometryPanel	494
3.256. RectangularSkyAperturePhotometryProduct	496
3.257. RectangularSkyAperturePhotometryTask	499
3.258. ReplaceFreqRanges	502
3.259. ResampleFrequency	504
3.260. RESHAPE	506
3.261. restore	508
3.262. resume	510
3.263. REVERSE	511
3.264. Rotate	512
3.265. rotateTask	515

3.266. ROUND	518
3.267. RowByRowAverageSpectrum	520
3.268. save	521
3.269. scaleTask	523
3.270. SelectSpectrum	526
3.271. SerialArchive	528
3.272. SerialClientPool	531
3.273. SHIFT	533
3.274. Short1d	535
3.275. Short2d	536
3.276. Short3d	537
3.277. Short4d	538
3.278. Short5d	539
3.279. SIGCLIP	540
3.280. SimpleCube	542
3.281. simpleFitsReader	550
3.282. simpleFitsWriter	552
3.283. SimpleImage	554
3.284. SimplePool	563
3.285. SIN	565
3.286. SINH	567
3.287. SKEWNESS	568
3.288. SmoothSpectrum	570
3.289. SOLVE	572
3.290. SORT	573
3.291. SourceExtractorTask	574
3.292. Spectrum1d	576
3.293. Spectrum2d	578
3.294. SpectrumPlotting	580
3.295. SpectrumStatistics	584
3.296. SpectrumTask	586
3.297. SpireDataFrameFactoryImpl	589
3.298. SpireDataFrameFactoryImpl	590
3.299. SQRT	591
3.300. SQUARE	592
3.301. STDDEV	593
3.302. String1d	594
3.303. SubtractSpectrum	595
3.304. SUM	598
3.305. TAN	600
3.306. TANH	602
3.307. TaskWrapper	603
3.308. ThreadDynamicProcessImpl	604
3.309. tiledImage	606
3.310. Transform2dTask	607
3.311. translateTask	608
3.312. TRANSPOSE	610
3.313. transposeTask	611
3.314. UNIQ_SORTED	613
3.315. UNIQ	615
3.316. VARIANCE	617
3.317. VelocityPosMapComputeTask	618
3.318. VelocityPosMapPlotting	620
3.319. Wcs	623
3.320. WeightedMean	642

Categorized view of Commands

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

Datasets

-  [NumberedDataset](#)
-  [Spectrum1d](#)
-  [Spectrum2d](#)

Display

-  [ImageAxis](#)

Image

-  [AnnotationToolbox](#)
-  [Display](#)
-  [Flag](#)
-  [SimpleCube](#)
-  [SimpleImage](#)
-  [Wcs](#)

binstruct

-  [PacketSequence](#)

class

-  [FitsArchive](#)
-  [HduHeaders](#)
-  [SerialArchive](#)

developer

-  [clear](#)

generic task

-  [FitterTask](#)
-  [GaussFitTask](#)

task

-  [bg](#)
-  [clear](#)
-  [compose](#)
-  [compute](#)
-  [database](#)
-  [displaylog](#)
-  [displayQCLog](#)
-  [displayQCLog](#)

- help
- localStoreWriter
- pause
- restore
- resume
- save
- simpleFitsReader
- simpleFitsWriter
- SourceExtractorTask

• **general**

- info

• **image**

- clampTask
- cropTask
- cutLevels
- imageSaverTask
- importCubeTask
- importImageTask
- rotateTask
- scaleTask
- tiledImage
- translateTask
- transposeTask

• **toolbox**

- RadialVelocity

• **utility task**

- asciiTableReader
- asciiTableWriter
- pointHistoryDisplay
- poolDataReader
- poolDataWriter

The Herschel Common Science System and Data Processing (DP)



Warning

A number of the examples provided in this document are not rendered correctly when viewed from the JIDE help facility. Please use either the PDF or HTML versions of this document to view the examples correctly.

II.1. Brief Overview

The Herschel Common Science System (HCSS) is being developed by the Herschel Science Center (HSC) and Herschel Instrument Control Centers (ICCs) to provide the complete software system for the Herschel Observatory mission. The intention is to provide a common system that is able to handle test data, observation planning, mission planning and instrument data from observations within one common development. An important element of this common development is Data Processing (DP).

DP handles computed, stored or simulated data and has access to much of the software developed for other purposes within the HCSS (e.g., Quick Look Analysis, which runs on real-time data or replayed data streams from a database).

Branches of the HCSS have also been developed for handling Herschel instrument-specific tasks. So software packages for HIFI, PACS and SPIRE also reside within the HCSS framework and are available within DP.

Since the Herschel DP uses Java programming, it is very flexible and Java programs can be imported into a session. However, the basic DP system is a fully-fledged standalone system that is being developed to specifically deal with data from the Herschel spacecraft.

II.2. Availability of DP and Operating Systems

DP is available free of charge as part of the HCSS and can be downloaded for use on networked or individual desktop/laptop machines. Current operating systems supported by DP include

- Solaris 2.8+
- LINUX (Red Hat 8.0+, SuSE 9.1)
- Windows (2000, XP)

For download and installation instructions see ????

II.3. Related Documentation

In earlier versions of the DP User's Manual, 'HowTo' documents were available in parallel. Earlier HowTo documents for users are now incorporated into the current User's Manual. Developers of DP packages have also produced [Javadocs](ftp://ftp.rssd.esa.int/pub/HERSCHEL/csd/releases/doc/api/index.html) (at <ftp://ftp.rssd.esa.int/pub/HERSCHEL/csd/releases/doc/api/index.html>) which currently provide some basic information on some of the underlying libraries and programs that comprise the DP system.



Note

Users should be aware that these are NOT fully fledged help documents and are probably most useful to system developers or advanced users only.

II.4. Versioning

DP is still very much a system under development and it is intended that this manual will be updated with the regular user release updates of the system. The first version of this manual is associated with User Release v0.3 of the HCSS. Version numbering of the manual will be matched to that of the user release (it is hoped). So the first manual is version 0.3.

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.
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 where users access the library components from within a JIDE session or a JConsole session as 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. 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 behavior 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__# herchel.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 behaviors:

Implicit Conversion

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

Examples:

```
x=DoubleId.range(3) # [0.0,1.0,2.0]
x[0]=1              # implicitly convert '1' to '1.0'
y=ByteId.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 `FloatId` array, but your data is stored in an `IntId` 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=FloatId.range(3)+0.5 # [0.5,1.5,2.5]
d=DoubleId(f)          # widening!
i=IntId(d)              # narrowing! [0,1,2]
z=ComplexId(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.:

```
Selection([1,3,7,200])
```

slice A slice is specified as: `begin:end[:step]`, where:

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 *excluded*. 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.

Note, that `[:]`,`[:,:]`,`[:end:]`,`[start:]`, `[:,:step]` are all valid statements, as each field is optional.

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  # returns a data1d of the same type as array
value=array[i0:i1,j]
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
cube[i,j0:j1,k0:k1]=value    # returns a data2d of the same type as cube
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	<code>x & y</code>	<code>[true,false,false,false]</code>
	inclusive or	<code>x y</code>	<code>[true,false,true,true]</code>
^	exclusive or	<code>x ^ y</code>	<code>[false,false,true,true]</code>

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	<code>x < y</code>	<code>[true,true,false]</code>
<=	less than or equals	<code>x <= y</code>	<code>[true,true,false]</code>
==	equals	<code>x == y</code>	<code>[false,true,false]</code>
!=	not equals	<code>x != y</code>	<code>[true,false,true]</code>
>=	greater than or equals	<code>x >= y</code>	<code>[false,true,true]</code>
>	greater than	<code>x > y</code>	<code>[false,false,true]</code>

Methods

This section is under construction.

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 <code>+</code> , <code>-</code> , <code>...</code> and potential benefits.
Methods	Expanding 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 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 x 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=DoubleId( [1,2,3,4,5,6,7,8,9,10] )

# Applying a manually created mask
q=x.where(BoolId( [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=ald.where(IS_FINITE)
print "Selected indices:",q # [0,1,3]

# --- filtering your data ---
print ald[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 ---
ald=Int1d( [4,1,3,9,2,7,4,5] )
a2d=Double2d([ [1,2,3,4], [4,3,2,1] ])

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

# --- query used as long index ---
print ald[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 [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", etc...

Relational Operators

All operators such as "==", "!=", "<", etc... have equivalent methods named "eq", "ne", "lt", etc...

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)           # contents of 'a' is altered!
r=a.copy().and(b).or(c) # contents of 'a' is 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!

1.5. 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 SIN, SQUARE, MIN, ...
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

- [LOG10](#)
- [AnyPresent](#)
- [ABS](#)
- [SHIFT](#)
- [UNIQ](#)
- [IS_INFINITE](#)
- [SQRT](#)
- [IS_FINITE](#)
- [LOG](#)
- [LogN](#)
- [EXP10](#)
- [SIGCLIP](#)
- [UNIQ_SORTED](#)
- [EXP](#)
- [IS_NAN](#)
- [Polynomial](#)
- [Rotate](#)

- SORT
- MEDIANDEV
- SQUARE
- REVERSE
- ExpN
- CONCATENATE
- RESHAPE
- NotPresent
- Pow

Trigonometric Functions

- SINH
- SIN
- TANH
- COSH
- COS
- ARCSIN
- TAN
- ARCCOS
- ARCTAN

Nearest Integer Functions

- FLOOR
- CEIL
- ROUND

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

```
x = | 1 2 |
    | 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](#)
- [AllPresent](#)
- [MAX](#)
- [ANY](#)
- [SUM](#)
- [PRODUCT](#)
- [ALL](#)

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
```

- [VARIANCE](#)
- [KURTOSIS](#)
- [Erf](#)
- [SKEWNESS](#)
- [GammaP](#)
- [MEDIAN](#)
- [GammaQ](#)
- [MEAN](#)
- [Correlate](#)
- [GAMMALN](#)
- [STDDEV](#)
- [CorrelateMatrix](#)
- [Erfc](#)

The Filter Toolbox

This toolbox provides filter functions.

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

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

Before proceeding it is wise to familiarize yourself with the background information about the Fitter classes.

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

The Integration Toolbox

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

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

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

The Interpolator Toolbox

This toolbox provides interpolation functions.

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

- [NearestNeighborInterpolator](#)
- [CubicSplineInterpolator](#)
- [Bilinear](#)
- [LinearInterpolator](#)

The Mask Handling Toolbox

This toolbox provides facilities for mask handling.

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

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

The Matrix Toolbox

This toolbox provides matrix functions.

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

- [SOLVE](#)
- [DETERMINANT](#)
- [INVERSE](#)
- [MATMUL](#)
- [MatrixSolve](#)
- [MatrixMultiply](#)
- [TRANSPOSE](#)
- [EigenvalueDecomposition](#)

The Random Functions Toolbox

This toolbox provides random functions.

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

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

The XFORM Toolbox

This toolbox provides transformation functions and utilities.

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

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

Chapter 2. Product Access Layer

2.1. Introduction

The Product Access Layer consists of several elements that allow you to save, retrieve and query Products that are stored in different locations whether that is on your local file system or remotely.

The **ProductStorage** class provides the user front-end interface that allows you to communicate with **Products** stored in multiple locations. Note that this is where the Product Access Layer fundamentally differs from the (versant) ObjectStore, that only allows to connect to one database in your session.

The locations mentioned above are what we call **Product Pools**; all Product pools adhere to the **ProductPool** interface. Example implementations of the ProductPool interface are local storage (on your laptop), and remote pools, such as the future Herschel Archive, accessing Products within a Versant database, a local storage made available by a fellow astronomer etcetera.

Simply by *registering* a pool to your storage, you can access the Products in that pool.

The ProductStorage provides mechanisms to load, save and query Products in the registered pools. Doing so you receive a reference to a Product (returned by the load() and save() commands) or a set of Product references (when querying). This functionality of a Product reference is provided by the **ProductRef** class; it allows to fetch information of the Product -such as meta data- without loading the Product in question in your memory completely.

The ProductStorage currently distincts three types of queries: 1) query on attributes that are available to all Products, 2) queries on meta data of the Product and 3) full data mining queries!

2.2. Example Usage

Creation and Registry

You can create a storage as follows:

```
storage=ProductStorage()
```

Then you have to assign the Product pools that you want to access. You have to register at least one pool:

```
storage.register(SimplePool.getInstance()) # simple storage on local file system
storage.register(SerialPoolClient("abc.xyz.org",123,"dummy"))
:
storage.register(poolN)
```

Save and Restoring Products

Saving a Product:

```
# creation of a dummy product
product=Product(creator="Me")
product["array"]=ArrayDataset(data=Int1d.range(5))

# save! returns reference to the saved product
reference=storage.save(product)
print reference.urn
# urn:simple.default:herschel.ia.dataset.Product:0
```


Loading a Product:

```
reference=storage.load("urn:simple.default:herschel.ia.dataset.Product:0")
```

A reference provides access parts of the product as well as access to the product itself:

```
print reference.urn
# urn:simple.default:herschel.ia.dataset.Product:0

print reference.type
# herchel.ia.dataset.Product

meta=reference.meta
print meta["creator"]
# Me

product=reference.product
print product.creator
# Me
```

Querying

To be written

2.3. Querying

The Product Access Layer allows to select product references from a [ProductStorage](#) by specifying a query.

Query types

We distinct three types of queries:

- [AttribQuery](#)

Allows searching on meta data values -so called attributes- that appear in any product. The attribute names are:

- creationDate : Date
- creator : String
- endDate : Date
- instrument : String
- modelName : String
- startDate : Date
- type : String

- [MetaQuery](#)

Allows searching on all meta data values and descriptions

- [FullQuery](#)

Allows searching on all aspects of Products exploiting the full interface of the Product family in question

Creation of a query

All queries are setup and executed on a [ProductStorage](#) as follows:

```
query=...Query(product-class,variable,where-expression)
results=storage.select(query)
```

,where:

- *product-class* is the product family. All products are derived from the Product class, but products may be categorized in sub-families such as PacsProduct.
- *variable* is a string holding the name of the product variable that is used in the *where-expression*.
- *where-expression* is a string that specifies the actual expression for which products of the *product-class* should be matched against. Note: the syntax is identical to the syntax of the JIDE command-line.

An example:

```
query=AttribQuery(SomeProduct, 'prod',
                  'prod.creator=="Me" and prod.modelName=="Fake"')
```

Query strategy

Typically an `AttribQuery` is faster than a `MetaQuery` which is in turn faster than a `FullQuery`. Depending on the product pools that are registered, a query can take some time; to avoid unnecessary waiting time one can adopt a strategy of staged queries.

For example, a query on attributes is executed first. If too many hits are found, you can refine your query by executing another query *using the hits returned from the previous query*. This process can be repeated until the number of hits have been reduced to a digestable amount:

```
results=storage.select(AttribQuery(...))      # 1000 hits
results=storage.select(MetaQuery(...),results) # 100 hits
results=storage.select(MetaQuery(...),results) # 50 hits
results=storage.select(FullQuery(...),results) # 3 hits
```

Querying for metadata in products

One thing you need to watch out when performing a meta or full query, is when you try to query for a metadata that does not exist in one or more products that you are applying the query to.

For example, consider the following `MetaQuery`:

```
query =MetaQuery(Product, 'p', 'p.meta["temperature"].value==10)
resultset=storage.select(query)
```

The query first starts creating a shortlist of all products in the storage matching type 'Product'. It then runs the query string on each product in that shortlist. If any of those products don't contain the information referenced in the query string, an error is raised.

There are two ways to avoid this:

- Be as specific as you can when it comes to specifying the product type in a query. If you know the product type you want to query is of type 'CalHrsQDCFull', then specify that. Running queries

using the most general product type of 'Product' is not recommended, unless the products you have saved are of this type only.

- Run a two-stage query, using the `containsKey()` operator to check whether a component exists first, eg

Get a sub-set of products that contain the metadata 'temperature':

```
queryOne= MetaQuery(Product, 'p', 'p.meta.containsKey("temperature")')
resultsetOne = storage.select(queryOne)
```

Run the original query on this subset:

```
queryTwo =MetaQuery(Product, 'p', 'p.meta["temperature"].value==10)
resultsetTwo = storage.select(queryTwo, resultSetOne)
```

2.4. Product Pools

Before you can do something useful with a created `ProductStorage`, you have to [register](#) one or more pools to that store.

Product pools are implementations that can load, save and query simple Products. All pools share the same features (the so-called `ProductPool` interface) such that they can be registered to a `ProductStorage`.

Typically a User sets up one `ProductStorage` and registers one or more Product pools to it. However the design permits to create multiple `ProductStorage` devices with a different registry of Product pools. Product pools can also be shared between two `ProductStorage` devices.

Depending of the Product Pool implementation you have selected, the products in such a pool can be stored on your local file system, talking to a object-oriented database, a relational database etcetera. You can even create your own `ProductPool` implementation (what about a memory pool, such that you can query all the Products available in memory of your JIDE session?).

Product Pool implementations

The following `ProductPool` implementations are available in the *Product Access Layer*:

<code>SimplePool</code>	This simple implementation stores Products on your local file system with reasonable query performance up to a few thousand Products.
<code>DbPool</code>	The <code>DbPool</code> stores Products in the Object (Versant) database.
<code>CachedPool</code>	A prototype implementation to cache communications with a <code>ProductPool</code> on your local file system. Useful when you wish to work in a disconnected environment.
<code>SerialClientPool</code>	A prototype implementation to set up a <code>ProductPool</code> server and an accompanying client.

2.5. What is a URN ?

'URN' stands for Uniform Resource Name, and is simply a string identifier to uniquely identify any product stored in any pool.

URNs are automatically generated and assigned to a product when that product is written to a storage. You can find out what the URN is for a product through its `ProductRef`. That `ProductRef` can in turn

be retrieved from the return value of the `ProductStorage.save()` operation that had initially saved that product:

```
ref = storage.save(myproduct) # Save the product and get the ref
print ref.urn                 # Get the URN
```

2.6. Context Products

Contexts are special types of products that contain references to other products stored.

This enables a means of building complex data structures in a storage.

There are two 'standard' types of context products provided: `ListContext` (for grouping products into sequences or lists) and `MapContexts` (for grouping products into containers with access to each by key).

2.7. Deep Copy or Cloning of Products

Say you had a context in one storage that referenced another product, and you wanted to copy that data tree to a different storage. How would you do that?

It is possible to do this using the usual `ProductStorage.save()` method. If you pass as an argument the context pointing to the 'head' of the data tree you want to clone, the whole data tree is cloned.

So for example, we have create a context with a child and store it in storage a:

```
l=ListContext()
p=Product()
l.refs.add(ProductRef(p))

storageA.save(l)
```

then we want to copy the context and child to a new storage, say storageB, all we do is as follows:

```
storageB.save(l)
```

The above cloning operation has one proviso: if a product within the data tree already exists in the destination product storage, it is not copied. A product can exist in the destination storage if for example, the original and destination storage happen to share a pool, and one of the products in the data tree being copied is in that common pool.

Note that a context may have older versions of it stored in a storage (a older version of a context may be saved when a context is saved, modified, then saved again). The older versions of the context specified in the `ProductStorage.save()` argument are also cloned (if that context has any descendents that are contexts, the local versions of those descendent contexts are not cloned, however).

2.8. Interface Definitions

Product Storage:

- [ProductStorage](#)
- [ProductRef](#)

- [MapContext](#)
- [ListContext](#)

ProductPool implementations:

- [SerialClientPool](#)
- [SimplePool](#)
- [CachedPool](#)
- [DbPool](#)

Query types:

- [MetaQuery](#)
- [AttribQuery](#)
- [FullQuery](#)

Context Products:

Product Browser:

Chapter 3. DP Commands

3.1. 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 description of each command follows its name. Beneath the general description of the command are a number of sections that describe the command in detail, its synopsis, its arguments, its limitations and examples.

Synopsis

The "Synopsis" section shows the proper syntax for calling 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 JIDE prompt. Others are code fragments or routines designed to serve as an examples for your own programs.

Arguments

The "Arguments" section describes each valid argument to the command. Note that most of these 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.

Limitations

The "Limitations" section describes the boundaries of applicability of the command.


Reference

The "Reference" section lists the names of related commands.

History

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

3.2. 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.

3.3. 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 synchronously 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.

In either mode, you can specify the segments you would like to process. See the 'segments' (for the scalar mode) or the 'segments1', 'segments2' parameters for the pair-wise mode.

For scalar mode, you have advanced selection functionality in place using the selection models as already discussed in the SpectrumTask.

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).

For further details see the (abstract) SpectrumTask in the same package.

Example

Example 1: from Jide:

```
from herschel.ia.toolbox.spectrum import AddSpectrum
add = AddSpectrum()
spectraOut = add(ds1=spectra1, ds2=spectra2)
spectraOut = add(ds=spectra, param=2.1)
```

API Summary

Properties
SpectrumContainer ds [INPUT, OPTIONAL, default=no default value.]
Double param [INPUT, OPTIONAL, default=no default value.]
SelectionModel selection [INPUT, OPTIONAL, default=no default value.]
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.]

Properties
SpectrumContainer ds1 [INPUT, OPTIONAL, default=no default value.]
SpectrumContainer ds2 [INPUT, OPTIONAL, default=no default value.]
SegmentSelection segments1 [INPUT, OPTIONAL, default=no default value.]
SegmentSelection 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 many-to-one operation (eg avg) or as scalar operation.
Double param [INPUT, OPTIONAL, default=no default value.]
The scalar parameter to be considered when the task is used as scalar operation.
SelectionModel selection [INPUT, OPTIONAL, default=no default value.]
Specify an instance of any kind of SelectionModel here.
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.
SegmentSelection segments1 [INPUT, OPTIONAL, default=no default value.]
SegmentSelection to be associated with 'ds1'.

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

SegmentSelection 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

3.4. 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

3.5. 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](#)

3.6. AmoebaFitter

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

Description

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.

Example


Example 1: The fit/demo directory contains worked

```
<a href="../../../ia/numeric/toolbox/fit/demo/jython.html">examples</a>  
on almost all aspects of of the package.
```

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.

3.7. 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
<code>AnnotationToolbox(Display d)</code> Constructor for the annotation toolbox.
<code>AnnotationToolbox(Display d, boolean useAsComponent)</code> Constructs the annotation toolbox.
Methods
<code>getComponent()</code> Returns the component that contains the annotation toolbox.
<code>close()</code> Closes the annotation toolbox.
<code>saveJythonCode()</code> Saves the jython code to a python script.

API details

Constructors

<code>AnnotationToolbox(Display d)</code> Constructs the annotation toolbox.
Argument <code>Display d</code> [INPUT, MANDATORY]
<code>AnnotationToolbox(Display d, boolean useAsComponent)</code> Constructs the annotation toolbox. If useAsComponent is true, the annotation toolbox is component based.
Arguments <code>Display d</code> [INPUT, MANDATORY] boolean <code>useAsComponent</code> [INPUT, MANDATORY]


Methods

getComponent()
Returns the component that contains the annotation toolbox.

close()
Closes the annotation toolbox.

saveJythonCode()
Saves the jython code to reconstruct the annotations to a python script.

3.8. AnnularSkyAperturePhotometryExplorer

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

Description

An explorer for AnnularSkyAperturePhotometryProducts.

API Summary

Constructors
<code>AnnularSkyAperturePhotometryExplorer()</code> The constructor of a new AnnularSkyAperturePhotometryExplorer.
<code>AnnularSkyAperturePhotometryExplorer(Object object)</code> The constructor of a new AnnularSkyAperturePhotometryExplorer
Methods
<code>boolean canHandle(Class className)</code> Checks whether this AnnularSkyAperturePhotometryExplorer can
<code>AnnularSkyAperturePhotometryProduct getObject()</code> Returns the object for this AnnularSkyAperturePhotometryExplorer.
<code>JTable getParameterTable()</code> Constructs and returns the parameter table for this
<code>JPanel getPlotPanel()</code> Constructs and returns the plot panel for this
<code>JPanel getSkyIntensityPlotPanel()</code> Constructs and returns the plot panel for this
<code>JComponent getSkyIntensityPlot()</code> Constructs and returns the sky intensity plot for this

API details

Constructors

<code>AnnularSkyAperturePhotometryExplorer()</code>
<code>AnnularSkyAperturePhotometryExplorer(Object object)</code> associated with the given object.
Argument <code>Object object</code> [INPUT, MANDATORY]

Methods

<code>boolean canHandle(Class className)</code>
handle objects of the given class.

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

AnnularSkyAperturePhotometryProduct getObject()
Return AnnularSkyAperturePhotometryProduct Returns the object for this AnnularSkyAperturePhotometryExplorer.


JTable getParameterTable()
AnnularSkyAperturePhotometryExplorer. Return JTable Returns the parameter table for this AnnularSkyAperturePhotometryExplorer.

JPanel getPlotPanel()
AnnularSkyAperturePhotometryExplorer. Return JPanel Returns the plot panel for this AnnularSkyAperturePhotometryExplorer.

JPanel getSkyIntensityPlotPanel()
AnnularSkyAperturePhotometryExplorer. Return JPanel Returns the plot panel for this AnnularSkyAperturePhotometryExplorer.

JComponent getSkyIntensityPlot()
AnnularSkyAperturePhotometryExplorer. Return JComponent Returns the sky intensity plot for this AnnularSkyAperturePhotometryExplorer.

3.9. AnnularSkyAperturePhotometryPanel

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

Description

A panel for the AnnularSkyAperturePhotometryTask.

API Summary

Constructor
<p>AnnularSkyAperturePhotometryPanel()</p> <p>The constructor of a new AnnularSkyAperturePhotometryPanel.</p>
Methods
<p>JPanel getAperturesPanel()</p> <p>Returns the panel where to specify the radii for the</p>
<p>JComponent getSkyApertureComponent()</p> <p>Returns the component where to specify the inner and outer</p>
<p>transferFromModifierToSelectionMap()</p> <p>Transfers information from the modifier map to the</p>
<p>drawFigures()</p> <p>Draws the circles on the image for this</p>
<p>moveConfirmedFigures()</p> <p>Allows the user to move the circles bounding the</p>

API details

Constructor

AnnularSkyAperturePhotometryPanel()

Methods

<p>JPanel getAperturesPanel()</p> <p>apertures for this AnnularSkyAperturePhotometryPanel.</p> <p>Return</p> <p>JPanel</p> <p>Returns the panel where to specify the radii for the apertures for this AnnularSkyAperturePhotometryPanel.</p>
<p>JComponent getSkyApertureComponent()</p> <p>radius of the annular sky aperture for this AnnularSkyAperturePhotometryPanel.</p>

JComponent <code>getSkyApertureComponent()</code>
--

Return**JComponent**

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

transferFromModifierToSelectionMap()

selection map associated with this AnnularSkyAperturePhotometryPanel.


drawFigures()

AnnularSkyAperturePhotometryPanel.

moveConfirmedFigures()

apertures for this AnnularSkyAperturePhotometryPanel.

3.10. 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
<code>AnnularSkyAperturePhotometryProduct()</code> The constructor of a new AnnularSkyAperturePhotometryProduct.
Methods
<code>setRadii(double innerPixels, double outerPixels)</code> Sets the inner and outer radius in pixels for this
<code>setRadii(double innerPixels, double innerArcsec, double outerPixels, double outerArcsec)</code> Sets the inner and outer radius for this
<code>double getInnerRadiusPixels()</code> Returns the inner radius for this AnnularSkyAperturePhotometryProduct in pixels.
<code>double getInnerRadiusArcsec()</code> Returns the inner radius for this AnnularSkyAperturePhotometryProduct
<code>double getOuterRadiusPixels()</code> Returns the outer radius for this AnnularSkyAperturePhotometryProduct in pixels.
<code>double getOuterRadiusArcsec()</code> Returns the outer radius for this AnnularSkyAperturePhotometryProduct in arcsec.
<code>TableDataset getSkyIntensityPlot()</code> Returns the sky intensity plot for this
<code>DoubleId getSkyRadius()</code> Returns the sky radius in pixels for the sky intensity
<code>DoubleId getSkyIntensity()</code> 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()
```

Methods

```
setRadii(double innerPixels, double outerPixels)
```

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)
```

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()
```

Return

double

Returns the inner radius for this AnnularSkyAperturePhotometryProduct in pixels.

```
double getInnerRadiusArcsec()
```

in arcsec.

Return

double

Returns the inner radius for this AnnularSkyAperturePhotometryProduct in arcsec.

```
double getOuterRadiusPixels()
```

Return

double

Returns the outer radius for this AnnularSkyAperturePhotometryProduct in pixels.

```
double getOuterRadiusArcsec()
```

Return

double

Returns the outer radius for this AnnularSkyAperturePhotometryProduct in arcsec.

TableDataset `getSkyIntensityPlot()`

AnnularSkyAperturePhotometryProduct.

Return

TableDataset

Returns the sky intensity plot for this AnnularSkyAperturePhotometryProduct.

Double1d `getSkyRadius()`

plot for this AnnularSkyAperturePhotometryProduct.

Return

Double1d

Returns the sky radius in pixels for the sky intensity plot for this AnnularSkyAperturePhotometryProduct.

Double1d `getSkyIntensity()`


AnnularSkyAperturePhotometryProduct.

Return

Double1d

Returns the sky intensity for the sky intensity plot for this AnnularSkyAperturePhotometryProduct.

3.11. 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, usgin 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.

3.12. 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](#)

3.13. 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](#)

3.14. AperturePhotometryExplorer

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

Description

An explorer for AperturePhotometryProducts.

API Summary

Constructors
<code>AperturePhotometryExplorer()</code> The constructor of a new AperturePhotometryExplorer.
<code>AperturePhotometryExplorer(Object object)</code> The constructor of a new AperturePhotometryExplorer associated
Methods
<code>String getName()</code> Returns the name for this AperturePhotometryExplorer.
<code>String getDescription()</code> Returns the description for this AperturePhotometryExplorer.
<code>boolean canHandle(Class className)</code> Checks whether this AperturePhotometryExplorer can
<code>setObject(Object object)</code> Sets the object for this AperturePhotometryExplorer to the given object.
<code>AperturePhotometryProduct getObject()</code> Returns the object for this AperturePhotometryExplorer.
<code>JComponent getComponent()</code> Returns the component that is responsible for displaying the
<code>JTable getParameterTable()</code> Constructs and returns the parameter table for this
<code>JTable getResultsTable()</code> Constructs and returns the results table for this AperturePhotometryExplorer.
<code>JPanel getPlotPanel()</code> Constructs and returns the plot panel for this
<code>JPanel getCurveOfGrowthPanel()</code> Constructs and returns the curve of growth for this
<code>JComponent getCurveOfGrowth()</code> Constructs and returns the curve of growth for this AperturePhotometryExplorer.
<code>addDataObjectListener(DataObjectListener listener)</code> Add the given listener to this AperturePhotometryExplorer
<code>removeDataObjectListener(DataObjectListener listener)</code>

Methods

Removes the given listener for this AperturePhotometryExplorer,

API details

Constructors

AperturePhotometryExplorer()

AperturePhotometryExplorer(Object object)

with the given object.

Argument

Object object [INPUT, MANDATORY]

Methods

String getName()

Return

String

Returns the name for this AperturePhotometryExplorer.

String getDescription()

Return

String

Returns the description for this AperturePhotometryExplorer.

boolean canHandle(Class className)

handle objects of the given class.

Argument

Class className [INPUT, MANDATORY]

Return

boolean

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

setObject(Object object)

Argument

Object object [INPUT, MANDATORY]

AperturePhotometryProduct getObject()

Return

AperturePhotometryProduct

AperturePhotometryProduct getObject()
Returns the object for this AperturePhotometryExplorer.
JComponent getComponent()
data object for this AperturePhotometryExplorer.
Return
JComponent
Returns the component that is responsible for displaying the data object for this AperturePhotometryExplorer.
JTable getParameterTable()
AperturePhotometryExplorer.
Return
JTable
Returns the parameter table for this AperturePhotometryExplorer.
JTable getResultsTable()
Return
JTable
Returns the results table for this AperturePhotometryExplorer.
JPanel getPlotPanel()
AperturePhotometryExplorer.
Return
JPanel
Returns the plot panel for this AperturePhotometryExplorer.
JPanel getCurveOfGrowthPanel()
AperturePhotometryExplorer.
Return
JPanel
Returns the curve of growth for this AperturePhotometryExplorer.
JComponent getCurveOfGrowth()
Return
JComponent
Returns the curve of growth for this AperturePhotometryExplorer.
addDataObjectListener(DataObjectListener listener)
to receive data object events from it.
Argument

addDataObjectListener(DataObjectListener listener)

<code>DataObjectListener listener</code> [INPUT, MANDATORY]


removeDataObjectListener(DataObjectListener listener)
--

so that it no longer receives data object events by this explorer.
--

Argument

<code>DataObjectListener listener</code> [INPUT, MANDATORY]

3.15. AperturePhotometryPanel

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

Description

A panel for aperture photometry.

API Summary

Constructor
AperturePhotometryPanel() The construction of a new AperturePhotometryPanel.
Methods
JPanel getImagePanel() Returns a panel that displays the image for this
JPanel getTargetCenterPanel() Returns the panel where to specify the target center
adjustTargetCenter() Allows the user to drag the circle indicating the target
boolean isInImage(MouseEvent me) Checks whether the given mouse event occurred inside the
boolean isInImage(double pixX, double pixY) Checks whether the given pixel coordinates lies
JPanel getAperturesPanel() Returns the panel where to specify the radius/radii for
JPanel getTargetAperturePanel() Returns the panel where to specify the target radius
JComponent getSkyApertureComponent() Returns the panel where to specify the parameters for
JPanel getAlgorithmPanel() Returns the panel where to specify the kind of pixels
JPanel getButtonPanel() Returns the button panel for this AperturePhotometryPanel
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) Sets the variable selection for this AperturePhotometryPanel

Methods	
<code>setSiteEventHandler(SiteEventHandler handler)</code>	Sets the site event handler for this AperturePhotometryPanel
<code>setTask(TaskApi task)</code>	Associates the given task with this AperturePhotometryPanel.
Display <code>getDisplay()</code>	Returns the display for this AperturePhotometryPanel.
Image <code>getImage()</code>	Returns the image associated with this
TaskApi <code>getTask()</code>	Returns the task associated with this
Map <code>getSelectionMap()</code>	Returns the map associated with this
Map <code>getModifierMap()</code>	Returns the map with the modifiers associated
SiteEventHandler <code>getHandler()</code>	Returns the site event handler associated with this
double <code>getTargetRadius()</code>	Returns the target radius for this
<code>trigger()</code>	Triggers the execution of the task associated
<code>transferFromModifierToSelectionMapCommon()</code>	Transfers the common information for this AperturePhotometryPanel
<code>transferFromModifierToSelectionMap()</code>	Transfers information from the modifier map to the
<code>drawFigures()</code>	Draws the figures bounding the apertures for this
<code>moveConfirmedFigures()</code>	Allows to move/resize the figures bounding the apertures

API details

Constructor

<code>AperturePhotometryPanel()</code>
--

Methods

<code>JPanel getImagePanel()</code>
PhotometryPanel.
Return
<code>JPanel</code>
Returns a panel that displays the image for this AperturePhotometryPanel.

JPanel <code>getTargetCenterPanel()</code>
for this AperturePhotometryPanel.
Return
JPanel
Returns the panel where to specify the target center for this AperturePhotometryPanel.
adjustTargetCenter()
center for this AperturePhotometryPanel.
boolean <code>isInImage(MouseEvent me)</code>
image for this AperturePhotometryPanel.
Argument
MouseEvent me [INPUT, MANDATORY]
Return
boolean
Returns true if the given mouse event occurred inside the image for this AperturePhotometryPanel.
boolean <code>isInImage(double pixX, double pixY)</code>
inside the image for this AperturePhotometryPanel.
Arguments
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 <code>getAperturesPanel()</code>
the apertures for this AperturePhotometryPanel.
Return
JPanel
Returns the panel where to specify the radius/radii for the apertures for this AperturePhotometryPanel.
JPanel <code>getTargetAperturePanel()</code>
for this AperturePhotometryPanel.
Return
JPanel
Returns the panel where to specify the target radius for this AperturePhotometryPanel.
JComponent <code>getSkyApertureComponent()</code>
the annular/rectangular sky aperture for this AperturePhotometryPanel.

```
JComponent getSkyApertureComponent()
```

Return**JComponent**

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

```
JPanel getAlgorithmPanel()
```

to use and the sky estimation algorithm for this AperturePhotometryPanel.

Return**JPanel**

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

```
JPanel getButtonPanel()
```

Return**JPanel**

Returns the button panel for this AperturePhotometryPanel.

```
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)
```

Argument

String title [INPUT, MANDATORY]

Return**Border**

Returns a border with the given title.

```
setVariableSelection(VariableSelection selection)
```

to the given variable selection.

Argument

VariableSelection selection [INPUT, MANDATORY]

```
setSiteEventHandler(SiteEventHandler handler)
```

to the given site event handler.

Argument

```
setSiteEventHandler(SiteEventHandler handler)
```

```
SiteEventHandler handler [INPUT, MANDATORY]
```

```
setTask(TaskApi task)
```

Argument

```
TaskApi task [INPUT, MANDATORY]
```

```
Display getDisplay()
```

Return

```
Display
```

Returns the display for this AperturePhotometryPanel.

```
Image getImage()
```

AperturePhotometryPanel.

Return

```
Image
```

Returns the image associated with this AperturePhotometryPanel.

```
TaskApi getTask()
```

AperturePhotometryPanel.

Return

```
TaskApi
```

Returns the task associated with this AperturePhotometryPanel.

```
Map getSelectionMap()
```

AperturePhotometryPanel.

Return

```
Map
```

Returns the map associated with this AperturePhotometryPanel.

```
Map getModifierMap()
```

with this AperturePhotometryPanel.

Return

```
Map
```

Returns the map with the modifiers associated with this AperturePhotometryPanel.

```
SiteEventHandler getHandler()
```

AperturePhotometryPanel.

Return

```
SiteEventHandler
```

Returns the site event handler associated with this AperturePhotometryPanel.

double getTargetRadius()

AperturePhotometryPanel in pixels.

Return**double**

Returns the target radius for this AperturePhotometryPanel in pixels.

trigger()

with this AperturePhotometryPanel.

transferFromModifierToSelectionMapCommon()

from the modifier to the selection map.

transferFromModifierToSelectionMap()

variable selection map associated with this AperturePhotometryPanel.


drawFigures()

AperturePhotometryPanel on the image.

moveConfirmedFigures()

for this AperturePhotometryPanel.

3.16. 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
setAlgorithm(int index) Sets the sky estimation algorithm for this AperturePhotometryProduct to the
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 getAlgorithm()

Methods	
	Returns the String representation of the sky estimation algorithm for this
<code>String getPixels()</code>	Returns the String representation of the type of pixels (fractional / entire) used
<code>String getUnit()</code>	Returns the unit for this AperturePhotometryProduct.
<code>TableDataset getTable()</code>	Returns the results table for this AperturePhotometryProduct.
<code>Double2d getDouble2dTable()</code>	Returns the results table for this AperturePhotometryProduct as a Double2d.
<code>Double1d getTotal()</code>	Returns the total flux for the target, the sky and the target from
<code>Double1d getNbOfPixels()</code>	Returns the number of pixels for the target, the sky and the target from
<code>Double1d getIntensityPerPixel()</code>	Returns the intensity per pixel for the target, the sky and the
<code>Double1d getError()</code>	Returns the error for the target, the sky and the target from which the
<code>double getTargetPlusSkyTotal()</code>	Returns the total flux for the target (sky included) for this
<code>double getNbOfTargetPlusSkyPixels()</code>	Returns the number of pixels for the target (sky included) for this
<code>double getIntensityPerTargetPlusSkyPixel()</code>	Returns the intensity per pixel for the target (sky included) for this
<code>double getTargetPlusSkyError()</code>	Returns the error for the target (sky included) for this AperturePhotometryProduct.
<code>double getSkyTotal()</code>	Returns the total flux for this AperturePhotometryProduct.
<code>double getNbOfSkyPixels()</code>	Returns the number of pixels for the sky for this AperturePhotometryProduct.
<code>double getIntensityPerSkyPixel()</code>	Returns the intensity per pixel for the sky for this AperturePhotometryProduct.
<code>double getSkyError()</code>	Returns the error for the sky for this AperturePhotometryProduct.
<code>double getTargetTotal()</code>	Returns the total flux for the target (sky subtracted) for this
<code>double getNbOfTargetPixels()</code>	Returns the number of pixels for the target (sky subtracted) for this
<code>double getIntensityPerTargetPixel()</code>	Returns the intensity per pixel for the target (sky subtracted) for this
<code>double getTargetError()</code>	Returns the error for the target (sky subtracted) for this

Methods
TableDataset <code>getCurveOfGrowth()</code> Returns the curve of growth for this AperturePhotometryProduct.
DoubleId <code>getGrowthRadius()</code> Returns the growth radius for this AperturePhotometryProduct.
DoubleId <code>getGrowthFlux()</code> Returns the growth flux for this AperturePhotometryProduct,

Miscellaneous

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

API details

Constructor

<code>AperturePhotometryProduct()</code>

Methods

<code>setTargetCenter(double centerX, double centerY)</code>
the given pixel coordinates.
Arguments
double centerX [INPUT, MANDATORY]
double centerY [INPUT, MANDATORY]

<code>setTargetCenter(double centerX, double centerY, String centerRA, String centerDec)</code>
to the given pixel and sky coordinates.
Arguments
double centerX [INPUT, MANDATORY]
double centerY [INPUT, MANDATORY]
String centerRA [INPUT, MANDATORY]
String centerDec [INPUT, MANDATORY]

<code>setTargetRadius(double pixels)</code>
to the given number of pixels.
Argument
double pixels [INPUT, MANDATORY]

<code>setTargetRadius(double pixels, double arcsec)</code>
to the given number of pixels and arcsec.
Arguments
double pixels [INPUT, MANDATORY]
double arcsec [INPUT, MANDATORY]

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

<code>double getTargetRadiusArcsec()</code>
double Returns the target radius for this AperturePhotometryProduct in arcsec.
<code>String getAlgorithm()</code>
AperturePhotometryProduct. Return String Returns the String representation of the sky estimation algorithm for this AperturePhotometryProduct.
<code>String getPixels()</code>
for this AperturePhotometryProduct. Return String Returns the String representation of the type of pixels (fractional / entire) used for this AperturePhotometryProduct.
<code>String getUnit()</code>
Return String Returns the unit for this AperturePhotometryProduct.
<code>TableDataset getTable()</code>
Return TableDataset Returns the results table for this AperturePhotometryProduct.
<code>Double2d getDouble2dTable()</code>
Return Double2d Returns the results table for this AperturePhotometryProduct as a Double2d.
<code>Double1d getTotal()</code>
which the sky has been subtracted for this AperturePhotometryProduct. Return Double1d Returns the total flux for the target, the sky and the target from which the sky has been subtracted for this AperturePhotometryProduct.

DoubleId getNbOfPixels()

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()

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()

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()

AperturePhotometryProduct.

Return**double**

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

double getNbOfTargetPlusSkyPixels()

AperturePhotometryProduct.

Return**double**

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

double getIntensityPerTargetPlusSkyPixel()

AperturePhotometryProduct.

Return**double**

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

double getTargetPlusSkyError()**Return**

double <code>getTargetPlusSkyError()</code>
double
Returns the error for the target (sky included) for this AperturePhotometryProduct.

double <code>getSkyTotal()</code>
Return
double
Returns the total flux for this AperturePhotometryProduct.

double <code>getNbOfSkyPixels()</code>
Return
double
Returns the number of pixels for the sky for this AperturePhotometryProduct.

double <code>getIntensityPerSkyPixel()</code>
Return
double
Returns the intensity per pixel for the sky for this AperturePhotometryProduct.

double <code>getSkyError()</code>
Return
double
Returns the error for the sky for this AperturePhotometryProduct.


double <code>getTargetTotal()</code>
AperturePhotometryProduct.
Return
double
Returns the total flux for the target (sky subtracted) for this AperturePhotometryProduct.

double <code>getNbOfTargetPixels()</code>
AperturePhotometryProduct.
Return
double
Returns the number of pixels for the target (sky subtracted) for this AperturePhotometryProduct.

double <code>getIntensityPerTargetPixel()</code>
AperturePhotometryProduct.
Return

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

3.17. AperturePhotometryTask

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

Description

image, INPUT, Image, MANDATORY, No default value

The image.

API Summary

Properties
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]
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 parameters concerning the (annular/rectangular) sky aperture are specified in the subclasses/subtasks (AnnularSkyAperturePhotometryTask and RectangularSkyAperturePhotometryTask).


API details

Properties

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.

3.18. 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](#)

3.19. 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](#)

3.20. 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](#)

3.21. 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)
```

Example 4: Read a file without HCSS header (first row contains names), delimiter is tab.

```
tableDataset = atrt(file="file.ascii", parserDelim='\t',
    parserGuess=AsciiParser.GUESS_TRY, parseNames=1)
```

API Summary

Jython Syntax

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

Jython Syntax
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 #3674), serialVersionUID, parameter descriptions, better jython doc

3.22. 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: AsciiTableWriterTask
<pre> <pre> #write a file ... atwt = AsciiTableWriterTask() atwt(file="file.ascii", table=TableDataset()) ... </pre> </pre>

API Summary

Jython Syntax
<pre> AsciiTableWriterTask()(file="file.ascii", table=TableDataset()) </pre>
see examples

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

API details


Properties

String file [INPUT, true, default=null]
Output file.
String table [INPUT, true, default=null]
TableDataset object to be saved.
String configFile [INPUT, false, 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, false, default=null]
If a file is specified, an output configuration file will be created.
AsciiFormatter formatter [INPUT, false, default=default AsciiTableTool formatter.]
AsciiFormatter object
Boolean formatterHeader [INPUT, false, default=default AsciiFormatter header allowance value.]
Specifies allowance of header information <code>AsciiFormatter.setHeader(true/false)</code> is called.
Boolean formatterCommented [INPUT, false, default=default AsciiFormatter comments allowance value.]
Specifies allowance of comments when writing a file. <code>AsciiFormatter.setCommented(true/false)</code> is called.
String formatterCommentPrefix [INPUT, false, default=default AsciiFormatter comments prefix value.]
Specifies the prefix of all comments. <code>AsciiFormatter.setCommentPrefix(expression)</code> is called.
TableTemplate template [INPUT, false, default=extracted from the first file rows.]
Specifies the data structure.

History

- 22-11-2007 JCS

3.23. 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

- [Querying](#)

3.24. AutomaticContourTask

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

Description

A Task for making an ImageContour for a given image for given contour levels.

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]
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.
ImageContour contours [OUTPUT, MANDATORY, default=No default value]	The ImageContour.

3.25. AverageFilteringTask

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

Description

A Task to smooth an image using the average (mean) filter.

The average filter computes the sum of all pixel values in the 3*3 filter window and then divides the sum by the number of pixels in the filter window (9). In order to filter pixels located near the edges of the image, edge pixel values are replicated to given sufficient data.

API Summary


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

API details

Properties

Image image [INPUT, MANDATORY, default=No default value]
The image before filtering.
Image filtered [OUTPUT, MANDATORY, default=No default value]
The image after filtering.

3.26. AverageNoiseTask

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

Description

A Task to make an average noise image of an image, using average filtering.

API Summary


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

API details

Properties

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

3.27. 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 a SpectrumContainer (see in herschel.ia.dataset.spectrum).

The averaging operation is applied to the spectral segments included in the different spectra. 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 on 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")
```

Different selection schemes are available for configuring the kind of average to be calculated. The most simple scheme is to use 'segments' where you specify which point spectra and subsegments therein you want to consider for the averaging. A different scheme consists in specifying a recipe that typically evaluates information contained in attributes. These recipes we refer to as selection models. In case a selection model is specified, individual spectra are pre-selected from the container before the average is taken on the sub-selection. The different options to specify selections can be combined. Here, an AND logic is adopted. 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.

For further details see the (abstract) SpectrumTask in the same package.

Example

Example 1: from Jide:

```
from herschel.ia.toolbox.spectrum import AverageSpectrum
avg = AverageSpectrum()
avg.register("integrations", "ADD")
averagedSpectra = avg(ds=spectra)
avgSelectedSpectra1 = avg(ds=spectra, selection_lookup={"bbtype": [111, 222]})
avgSelectedSpectra2 = avg(ds=spectra, selection_index=[1, 3, 4, 5, 10])
from herschel.ia.toolbox.spectrum.selections.models import RangesSelectionModel
avgSelectedSpectra3 = avg(ds=spectra, selection=RangesSelectionModel("Chopper",
[-4.2, 4.8], 0.1))
avgSelectedSpectraPerGroup = avg(ds=spectra, selection_lookup={"bbtype":
[111, 222]}, per_group=True)
from herschel.ia.toolbox.spectrum.selections.models import RangesGroupingModel
averagedSpectraPerGroup = avg(ds=spectra,
grouping=RangesGroupingModel("Chopper", 0.1))
```

API Summary

Properties

SpectrumContainer **ds** [INPUT, OPTIONAL, default=no default value.]

Properties
SegmentSelection segments [INPUT, OPTIONAL, default=no default value.]
SelectionModel selection [INPUT, OPTIONAL, default=no default value.]
PyDictionary Map> selection_lookup [INPUT, OPTIONAL, default=no default value.]
PyList selection_index [INPUT, OPTIONAL, default=No default value.]
String variant [INPUT, OPTIONAL, default=no default value.]
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

SpectrumContainer ds [INPUT, OPTIONAL, default=no default value.]
Input container to be processed by the task.
SegmentSelection segments [INPUT, OPTIONAL, default=no default value.]
Specify an instance of a SegmentSelection which gives the information on what segments for each point spectrum included in the container and, possibly, what ranges / masks should be considered for the processing.
SelectionModel selection [INPUT, OPTIONAL, default=no default value.]
Specify an instance of any kind of SelectionModel here.
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.
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"
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

- [SpectrumTask](#)

History

- 2007-06-01 - meli: initial.
- 2007-09-21 - meli: Javadoc updated.
- 2008-03-16 - meli: major refactoring of the toolbox.
- 2008-10-03 - meli: segments, javadoc

3.28. 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

Miscellaneous

No miscellaneous

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.

3.29. Bilinear

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

Examples

Example 1: Create and apply a Bilinear on a Double2d

```
#jython example
from herschel.ia.numeric import *
from herschel.ia.numeric.toolbox.interp import Bilinear
images=RESHAPE(Double1d.range(16), [4,4])
x=Double1d.range(3).add(0.5)
y=Double1d.range(3).add(0.5)
img1=Bilinear(x,y,1)(images)
img2=Bilinear(x,y)(images)
```

Example 2: Create and apply a Bilinear on a Double2d

```
#java example
import herschel.ia.numeric.toolbox.interp.*;
from herschel.ia.numeric import *
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);
Bilinear ff=new Bilinear(x, y, Bilinear.GRID);
Bilinear ff1=new Bilinear(x, y);
Double2d newImages1=images.apply(new Bilinear(x, y, Bilinear.GRID));
Double1d newImages2=images.apply(new Bilinear(x, y))
or
Double2d newImages1=new Bilinear(x, y, Bilinear.GRID)(images);
Double1d newImages2=new Bilinear(x, y)(images);
```

API Summary

Jython Syntax

```
<interp>=Bilinear(<x,y,[GRID]>)(<images>)
```

where <images>=input

<x, y> = coordinates where the interpolated values is to be computed

<interp>= output, the interpolated values at (x, y)

Properties

Double1d **x** [INPUT, MANDATORY, default=no default value]

Double1d **y** [INPUT, MANDATORY, default=no default value]

boolean **grid** [INPUT, NOT_MANDATORY, default=false]

API details

Properties

Double1d **x** [INPUT, MANDATORY, default=no default value]

<code>DoubleId y [INPUT, MANDATORY, default=no default value]</code>
--

<code>boolean grid [INPUT, NOT_MANDATORY, default=false]</code>

The result has the same structure and number of elements as x.
--


See also

- [CubicSplineInterpolator](#)
- [NearestNeighborInterpolator](#)
- [???](#)

History

- March 5, 2007 - first version
- March 24, 2008 - revised version
- Include missing argument to set the value where the point is outside the boundary
- The value to return for elements outside the bounds of P. If this keyword is not
- specified, interpolated positions that fall outside the bounds of the array P -
- that is, elements of the IX or JY arguments that are either less than zero or
- greater than the largest subscript in the corresponding dimension of P - are set
- equal to the value of the nearest element of P.

3.30. 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](#)

3.31. 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)

3.32. 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)

3.33. 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)

3.34. 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)

3.35. 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)

3.36. 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](#)

3.37. 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)

3.38. 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)

3.39. 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)

3.40. 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)

3.41. 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)

3.42. CachedPool

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

Description

A ProductPool implementation that caches a remote pool. This pool is

useful in situations where performance is important, or that network bandwidth needs to be optimized. The cached data is stored in your local file system under the directory defined by: `${hcss.ia.pal.pool.cache.dir}/pal_cache`. By default: `${HOME}/.hcss/pal_cache` (UNIX), or `C:\Documents and Settings\\.hcss\pal_cache` (Windows).

The identifier for the CachedPool (which you may need eg if you want to access that pool remotely) is the same ID as the pool which is being cached.

Example

Example 1: Create a CachedPool around a DbPool storage=ProductStorage()

```
storage.register(CachedPool(DbPool.getInstance()))
```

API Summary

Constructor

`CachedPool(String remote)`

Creates an instance of a CachedPool that is wrapped around another

API details

Constructor

`CachedPool(String remote)`

supplied ProductPool.

Argument

`String remote` [INPUT, MANDATORY, default=The name of the ProductPool]


this CachedPool should cache.

Return

`CachedPool`

An instance of the CachedPool.

3.43. 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
<pre>x=Float1d([0.1,0.5,0.9]) print CEIL(x) # [1.0,1.0,1.0] *</pre>

API Summary

Jython Syntax
<code><y>=CEIL(<x>)</code>
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](#)

3.44. 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

Constructors
ChannelMapPlotting (CubeSpectrumAnalysisToolbox toolbox) Constructor for ChannelMapPlotting. This constructor creates a window
ChannelMapPlotting (boolean useAsComponent, CubeSpectrumAnalysisToolbox 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
ArrayList getImageFigures () Returns all ImageFigures used for this ChannelMapPlotting.

API details

Constructors

ChannelMapPlotting(CubeSpectrumAnalysisToolbox toolbox)

in which you must give the input parameters, needed to perform contour plotting on the image, analysed with the given ImageAnalysisToolbox.

Argument

`CubeSpectrumAnalysisToolbox toolbox` [INPUT, MANDATORY]

ChannelMapPlotting(boolean useAsComponent, CubeSpectrumAnalysisToolbox toolbox)

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()

(manual or automatic).

Return

int

Returns the used type of contour values for this ChannelMapPlotting (manual or automatic).

int getUsedContourLevelRange()

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()

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()

Return

ArrayList

Returns a list with all contour values for this ChannelMapPlotting.

ArrayList `getContourColors()`

different contour values.

Return

ArrayList

Returns the colors that should be used for the contours of the different contour values.

int `getNumberOfContourLevels()`

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()`

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()`

Return

double

Returns the minimum length for contours to be drawn on the image.

String `getUsedDistribution()`

used for this ChannelMapPlotting (linear, log or ln).

Return

String

Returns the type of distribution of the contour levels, that is used for this ChannelMapPlotting (linear, log or ln).


ArrayList `getImageFigures()`

Return

ArrayList

Returns all ImageFigures used for this ChannelMapPlotting.

3.45. CircleHistogramExplorer

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

Description

An explorer for CircleHistogramProducts.

API Summary

Constructors
CircleHistogramExplorer() The constructor of a new CircleHistogramExplorer.
CircleHistogramExplorer(Object object) The constructor of a new CircleHistogramExplorer associated
Methods
String getName() Returns the name for this CircleHistogramExplorer.
String getDescription() Returns the description for this CircleHistogramExplorer.
boolean canHandle(Class className) Checks whether this CircleHistogramExplorer can handle objects of the
setObject(Object object) Sets the object for this CircleHistogramExplorer to the given object.
CircleHistogramProduct getObject() Returns the object for this CircleHistogramExplorer.
JTable getParameterTable() Returns the parameter table for this

API details

Constructors


CircleHistogramExplorer()
CircleHistogramExplorer(Object object)
with the given object.
Argument
Object object [INPUT, MANDATORY]

Methods

String getName()
Return

String getName()
String Returns the name for this CircleHistogramExplorer.
String getDescription()
Return String Returns the description for this CircleHistogramExplorer.
boolean canHandle(Class className)
given class. Argument Class className [INPUT, MANDATORY] Return boolean Returns true if this CircleHistogramExplorer can handle objects of the given class; false otherwise.
setObject(Object object)
Argument Object object [INPUT, MANDATORY]
CircleHistogramProduct getObject()
Return CircleHistogramProduct Returns the object for this CircleHistogramExplorer.
JTable getParameterTable()
CircleHistogramExplorer. Return JTable Returns the parameter table for this CircleHistogramExplorer

3.46. CircleHistogramPanel

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

Description

A panel for the CircleHistogramPanel.

API Summary

Constructor
<code>CircleHistogramPanel()</code> The construction of a new CircleHistogramPanel.
Methods
<code>drawFigure()</code> Draws the circle on the image associated with this
<code>updateFigure()</code> Updates the circle associated with this
<code>trigger()</code> Triggers the execution of the task associated
<code>updateHistogram()</code> Updates the histogram associated with this

API details


Constructor

<code>CircleHistogramPanel()</code>

Methods

<code>drawFigure()</code> CircleHistogramPanel.
<code>updateFigure()</code> CircleHistogramPanel.
<code>trigger()</code> with this CircleHistogramPanel.
<code>updateHistogram()</code> CircleHistogramPanel.

3.47. 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()

Methods

setCenter(double centerX, double centerY)
to the given pixel coordinates.
Arguments

setCenter(double centerX, double centerY)

double **centerX** [INPUT, MANDATORY]

double **centerY** [INPUT, MANDATORY]

setCenter(double centerX, double centerY, String centerRA, String centerDec)

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)

to the given number of pixels.

Argument

double **pixels** [INPUT, MANDATORY]

setRadius(double pixels, double arcsec)

to the given number of pixels and arcsec.

Arguments

double **pixels** [INPUT, MANDATORY]

double **arcsec** [INPUT, MANDATORY]

DoubleId getCenterPixelCoordinates()

pixel coordinates.

Return

DoubleId

Returns the center for this CircleHistogramProduct in pixel coordinates.

StringId getCenterSkyCoordinates()

sky coordinates.

Return

StringId

Returns the center for this CircleHistogramProduct in sky coordinates.

double getRadiusPixels()

Return

double

Returns the radius for this CircleHistogramProduct in pixels.

double getRadiusArcsec()


Return

<code>double getRadiusArcsec()</code>

<code>double</code>

Returns the radius for this CircleHistogramProduct in arcsec.

3.48. 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 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 : "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]

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.

3.49. clampTask

Full Name:	herschel.ia.toolbox.image.ClampTask
Alias:	clampTask
Type:	Java Task - 
Import:	from herschel.ia.toolbox.image import ClampTask
Category:	task/image

Description

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

```
clampTask(image = image, low = 11, high = 240)
```

API Summary

Jython Syntax

```
clampTask(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

See also

- [???](#)

3.50. clear

Full Name:	herschel.ia.task.example.InterruptableTask
Alias:	clear
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

Capture Ctrl-s events and provides a recovery action

Examples

Example 1: run the task

```
interruptable()
```

Example 2: run the task with 1000 iteration

```
interruptable(1000)
```

API Summary

Jython Syntax

```
clear("variableName") <br>
clear("variableA,variableB") <br>
clear(all=True)
```

Property

```
Integer iterations [INPUT, NO, default=10000]
```

Limitations

The current implementation allows deleting of packages and class, no pre-filter is performed

Miscellaneous

No miscellaneous

API details

Property

```
Integer iterations [INPUT, NO, default=10000]
```

The number of iterations to be performed


See also

- [references](#)

History

- 2004-07-13 - NdC: first release.

3.51. 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

3.52. 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)

3.53. 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)

3.54. 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)

3.55. 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)

3.56. 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)

3.57. compose

Full Name:	herschel.ia.history.example.CompositionTask
Alias:	compose
Type:	Java Task - 
Import:	from herschel.ia.history.example import CompositionTask
Category:	task

Description

A fake generic composition task for testing history.

This task is used in the history demo script.

Example

Example 1: overview compose

```
compose(product1, product2, "composition")
```

API Summary

Jython Syntax

```
compose(input1, input2, name)<br>
```

Properties

Product **input** [INPUT, MANDATORY, default=No default value]

Product **input2** [INPUT, MANDATORY, default=No default value]

String **control** [INPUT, OPTIONAL, default=""]

Limitations

No computation is actually performed.

Miscellaneous

No miscellaneous

API details

Properties

Product **input** [INPUT, MANDATORY, default=No default value]

The first product input

Product **input2** [INPUT, MANDATORY, default=No default value]

The second product input

String **control** [INPUT, OPTIONAL, default=""]

A name for the composition


See also

- [references](#)

History

- 2007-06-06 - JSS: first release.

3.58. compute

Full Name:	herschel.ia.history.example.ComputationTask
Alias:	compute
Type:	Java Task - 
Import:	from herschel.ia.history.example import ComputationTask
Category:	task

Description

A fake generic computation task for testing history.

This task is used in the history demo script.

Example

Example 1: overview compute

```
compute(product1, product2, 5)
```

API Summary

Jython Syntax

```
compute(input1, input2, controlvalue)<br>
```

Properties

Product **input** [INPUT, YES, default=No default value]

Product **input2** [INPUT, YES, default=No default value]

Integer **control** [INPUT, NO, default=5]

Limitations

No computation is actually performed.

Miscellaneous

No miscellaneous

API details

Properties

Product **input** [INPUT, YES, default=No default value]

The first product input

Product **input2** [INPUT, YES, default=No default value]

The second product input

Integer **control** [INPUT, NO, default=5]

A scalar control


See also

- [references](#)

History

- 2005-02-24 - NdC: first release.

3.59. 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
<pre>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]</pre>

API Summary

Jython Syntax
<code><y>=CONCATENATE(<x>)</code>

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](#)

3.60. Condense

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

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]
```

Example 1: Untitled

```

IA>>print Condense(0,2,SUM,"boxcar")(x)
[
  [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]
```

Properties
integer alongDimension [INPUT, MANDATORY, default=p]
int chunkSize [INPUT, MANDATORY, default=p]
ArrayReducer ArrayToNumber function [INPUT, MANDATORY, default=p]
OPTIONAL mode [INPUT, default = false, default=no default value]
float or double array outArray [OUTPUT, MANDATORY, default=no default value]

API details

Properties

float or double array inArray [INPUT, MANDATORY, default=p]
Input array of n dimensions
integer alongDimension [INPUT, MANDATORY, default=p]
Dimension in which the function will be applied (starting on zero)
int chunkSize [INPUT, MANDATORY, default=p]
Size of chunks to process. See boxcar parameter.
ArrayReducer ArrayToNumber function [INPUT, MANDATORY, default=p]
Function to apply
OPTIONAL mode [INPUT, default = false, default=no default value]
<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 theorhetical center (see boxcar parameter)

3.61. 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.

3.62. 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.

API Summary

Constructor
Contour (<code>Double2d contourPoints</code>) This constructor for Contour creates a new Contour and sets the
Methods
int <code>getNbOfContourPoints()</code> Returns the number of contour points for this Contour.
convert (<code>Wcs wcsOld</code> , <code>Wcs wcsNew</code>) Converts the pixel coordinates of the contour points of this

API details


Constructor

Contour (<code>Double2d contourPoints</code>)
data (contour points) for the new Contour to the given data (contour points).
Argument
<code>Double2d contourPoints</code> [INPUT, MANDATORY]

Methods

int <code>getNbOfContourPoints()</code>
Return
<code>int</code>
Returns the number of contour points for this Contour.
convert (<code>Wcs wcsOld</code> , <code>Wcs wcsNew</code>)
Contour from one Wcs to pixel coordinates to pixel coordinates in another Wcs.
Arguments
<code>Wcs wcsOld</code> [INPUT, MANDATORY]
<code>Wcs wcsNew</code> [INPUT, MANDATORY]

3.63. 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.

API Summary

Constructor
<p><code>ContourLevel(double contourValue)</code></p> <p>This constructor creates a new ContourLevel for the given contour</p>
Method
<p><code>addContour(Contour contour)</code></p> <p>Adds the given Contour to this ContourLevel.</p>

API details


Constructor

<code>ContourLevel(double contourValue)</code>
value.
Argument
double contourValue [INPUT, MANDATORY]

Method

<code>addContour(Contour contour)</code>
Argument
<code>Contour contour</code> [INPUT, MANDATORY]

3.64. ContourPlotting

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

Description

An implementation of contour plotting.

API Summary

Constructors
ContourPlotting (ImageAnalysisToolbox toolbox) The standard constructor for ContourPlotting. This constructor creates a window
ContourPlotting (boolean useAsComponent, ImageAnalysisToolbox 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)

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)

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\(\)](#)

(manual or automatic).

Return

int

Returns the used type of contour values for this ContourPlotting (manual or automatic).

int [getUsedContourLevelRange\(\)](#)

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\(\)](#)

errors occur in the input, null is returned.

Return

String

Returns an appropriate error message for the given input. If no errors occur in the input, null is returned.

ArrayList [getContourValues\(\)](#)

Return

ArrayList

Returns a list with all contour values for this ContourPlotting.

ArrayList `getContourColors()`

different contour values.

Return

ArrayList

Returns the colors that should be used for the contours of the different contour values.

int `getNumberOfContourLevels()`

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()`

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()`

Return

double

Returns the minimum length for contours to be drawn on the image.

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()`

Return

ArrayList

Returns all ImageFigures used for this ContourPlotting.

3.65. ContourTask

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

Description

A Task for making an ImageContour for a given image for a given contour value.

API Summary


Properties
Image image [INPUT, MANDATORY, default=No default value]
Double value [INPUT, MANDATORY, 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.
ImageContour contours [OUTPUT, MANDATORY, default=No default value]
The image contour.

3.66. 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
<pre>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]</pre>

API Summary

Jython Syntax
<pre><f>=Convolution(<kernel> [, <center>=true false] [, <edge>=Convolution.ZEROES CIRCULAR REPEAT]) <x>=<f>(<x>)</pre>
Properties
<pre>Double1d kernel [INPUT, MANDATORY, default=no default value]</pre>
<pre>boolean center [INPUT, NOT_MANDATORY, default=false]</pre>
<pre>Convolution.ZEROES CIRCULAR REPEAT center [INPUT, NOT_MANDATORY, default=Convolution.ZEROES]</pre>

API details

Properties

<pre>Double1d kernel [INPUT, MANDATORY, default=no default value]</pre>
It must be a Double1d array, when =true, the kernel should have an odd number of elements.
<pre>boolean center [INPUT, NOT_MANDATORY, default=false]</pre>
Set center to true or false.
<pre>Convolution.ZEROES CIRCULAR REPEAT center [INPUT, NOT_MANDATORY, default=Convolution.ZEROES]</pre>
Set edge to true or false
<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 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.

3.67. 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](#)

3.68. 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 $M \times N$ matrix.

The result is a $N \times N$ 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 $N \times N$ matrix with the linear Pearson correlation coefficients of the input matrix.

See also

- [Correlate](#)

3.69. 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
<code><y>=COS(<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>

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

- [ARCCOS](#)
- [COSH](#)
- [SIN](#)
- [TAN](#)

3.70. 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](#)

3.71. cropTask

Full Name:	herschel.ia.toolbox.image.CropTask
Alias:	cropTask
Type:	Java Task - 
Import:	from herschel.ia.toolbox.image import CropTask
Category:	task/image

Description

The Crop task for images and cubes.

Crop is a task which crops an image 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

```
cropTask(image = image, x1 = 11, x2 = 55, y1 = 240, y2 = 300)
```

API Summary

Jython Syntax

```
cropTask(image, 11, 240, 55, 300)
```

Properties

Image **image** [INPUT, MANDATORY, default=No default value]

float **row1** [INPUT, MANDATORY, default=No default value]

float **column1** [INPUT, MANDATORY, default=No default value]

float **row2** [INPUT, OPTIONAL, default=No default value]

float **column2** [INPUT, MANDATORY, default=No default value]

Image **croppedImage** [OUTPUT, MANDATORY, default=No default value]

API details

Properties

Image image [INPUT, MANDATORY, default=No default value]

The Image to be cropped

float row1 [INPUT, MANDATORY, default=No default value]

The x coordinate of the left upper pixel of the rectangle to crop

float column1 [INPUT, MANDATORY, default=No default value]

The y coordinate of the left upper pixel of the rectangle to crop

float row2 [INPUT, OPTIONAL, default=No default value]

The x coordinate of the lower right pixel of the rectangle to crop
--

float column2 [INPUT, MANDATORY, default=No default value]

The y coordinate of the lower right pixel of the rectangle to crop
--


Image croppedImage [OUTPUT, MANDATORY, default=No default value]

The cropped image

See also

- [???](#)

3.72. 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

Constructors
<p><code>CubeSpectrumAnalysis(CubeSpectrumAnalysisToolbox toolbox, String title)</code></p> <p>Standard constructor for CubeSpectrumAnalysis. A new window with the</p>
<p><code>CubeSpectrumAnalysis(CubeSpectrumAnalysisToolbox toolbox, String title, Boolean flag)</code></p> <p>Standard constructor for CubeSpectrumAnalysis. A new window with the</p>
Methods
<p><code>JFrame getFrame()</code></p> <p>Method that returns the JFrame of this CubeSpectrumAnalysis.</p>
<p><code>CubeSpectrumAnalysisToolbox getToolbox()</code></p> <p>Returns the CubeSpectrumAnalysisToolbox, associated with this</p>
<p><code>Container getComponent()</code></p> <p>Returns the content pane (Container) of the JFrame of this</p>
<p><code>ArrayList getImageFigures()</code></p> <p>Returns all ImageFigures used for this CubeSpectrumAnalysis.</p>

API details

Constructors

<p><code>CubeSpectrumAnalysis(CubeSpectrumAnalysisToolbox toolbox, String title)</code></p> <p>given title and associated with the given CubeSpectrumAnalysisToolbox is created.</p>
<p>Arguments</p> <p><code>CubeSpectrumAnalysisToolbox toolbox</code> [INPUT, MANDATORY]</p> <p><code>String title</code> [INPUT, MANDATORY]</p>
<p><code>CubeSpectrumAnalysis(CubeSpectrumAnalysisToolbox toolbox, String title, Boolean flag)</code></p> <p>given title and associated with the given CubeSpectrumAnalysisToolbox is created.</p>

```
CubeSpectrumAnalysis(CubeSpectrumAnalysisToolbox toolbox, String title, Boolean flag)
```

Arguments

```
CubeSpectrumAnalysisToolbox toolbox [INPUT, MANDATORY]  
String title [INPUT, MANDATORY]  
Boolean flag [INPUT, MANDATORY]
```

Methods

```
JFrame getFrame()
```

Return

JFrame

The JFrame of this CubeSpectrumAnalysis.

```
CubeSpectrumAnalysisToolbox getToolbox()
```

CubeSpectrumAnalysis.

Return

CubeSpectrumAnalysisToolbox

Returns the CubeSpectrumAnalysisToolbox, associated with this CubeSpectrumAnalysis.

```
Container getComponent()
```

CubeSpectrumAnalysis.

Return

Container

Returns the content pane (Container) of the JFrame of this CubeSpectrumAnalysis.


```
ArrayList getImageFigures()
```

Return

ArrayList

Returns all ImageFigures used for this CubeSpectrumAnalysis.

3.73. CubeSpectrumAnalysisToolbox

Full Name:	herschel.ia.gui.cube.CubeSpectrumAnalysisToolbox
Type:	Java Class - 
Import:	from herschel.ia.gui.cube import CubeSpectrumAnalysisToolbox

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)
iat = CubeSpectrumAnalysisToolbox(Double3d)<br>
or<br>
iat = CubeSpectrumAnalysisToolbox(SimpleCube)<br>
or<br>
iat = CubeSpectrumAnalysisToolbox(Cube,Wavearray)<br>
iat = CubeSpectrumAnalysisToolbox()<br>
iat.setCube(Cds)<br>
or <br>
iat = CubeSpectrumAnalysisToolbox() <br>
iat.setCube(Cube,WaveArray)<br>
or <br>
<br>
iat = CubeSpectrumAnalysisToolbox() <br>
iat.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)
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)

```


Example 1: A basic example on how to open a toolbox for spectrum analysis on an

```

simplecube2.setWcs(myWcs)
import herschel.ia.gui.cube.CubeSpectrumAnalysisToolbox
# initialization of the CubeSpectrumAnalysisToolbox
cat = CubeSpectrumAnalysisToolbox()
cat.setCube(simplecube2)

```

API Summary

Constructors	
<code>CubeSpectrumAnalysisToolbox()</code>	The standard constructor for CubeSpectrumAnalysisToolbox. This
<code>CubeSpectrumAnalysisToolbox(Array3dData array3d)</code>	Constructor for CubeSpectrumAnalysisToolbox. This constructor shows an
<code>CubeSpectrumAnalysisToolbox(Array3dData array3d, Double1d wave1d)</code>	Constructor for CubeSpectrumAnalysisToolbox. This constructor shows an
<code>CubeSpectrumAnalysisToolbox(Array3dData array3d, Double1d wave1d, String specDim, String specUnit)</code>	Constructor for CubeSpectrumAnalysisToolbox. This constructor shows an
<code>CubeSpectrumAnalysisToolbox(type simpleCube)</code>	Constructor for CubeSpectrumAnalysisToolbox. This constructor shows a
Methods	
<code>setCube(Array3dData array3d)</code>	Sets the image you wish to analyse with this CubeSpectrumAnalysisToolbox,
<code>setCube(SimpleCube simplecube)</code>	Sets the Cube you wish to analyse with this CubeSpectrumAnalysisToolbox,
<code>setCube(Array3dData array3d, Double1d wavearray)</code>	Sets the image you wish to analyse with this CubeSpectrumAnalysisToolbox,
<code>addCube(Array3dData array3d)</code>	Adds the given image (Array3dData) to the images, analysed with
<code>exit()</code>	Exits this CubeSpectrumAnalysisToolbox (closes (the window of) this
<code>closeActiveTab()</code>	Closes the active tab, if there is one.
<code>closeAllTabs()</code>	Closes all tabs, if there are any.
<code>setEnabled(boolean enabled)</code>	Disables/enables all tabs on the JTabbedPane and the menus for
<code>boolean isEnabled()</code>	Returns whether the JTabbedPane and menus for closing tabs and
<code>Double1d getWaves()</code>	Method that returns the Float1d array of the wavelength of this CubeSpectrumAnalysisToolbox.
<code>SimpleCube getSimpleCube()</code>	

Methods	
	Method that returns the Float1d array of the wavelength of this CubeSpectrumAnalysisToolbox.
JFrame <code>getFrame()</code>	Method that returns the JFrame of this CubeSpectrumAnalysisToolbox.
JPanel <code>getPanel()</code>	Method that returns the JPanel of this CubeSpectrumAnalysisToolbox.
JTabbedPane <code>getTabbedPane()</code>	Method that returns the JTabbedPane of this CubeSpectrumAnalysisToolbox.
SimpleImage <code>getSimpleImage()</code>	Returns the displayed image (SimpleImage).
PlotXY <code>getRtSpectrum()</code>	Returns the displayed image (ImageDataset).
Display <code>getDisplay()</code>	Returns the Display of the ImageDataset you are analysing with
int <code>getWidth()</code>	Gives the width (in pixels) of the image we are analysing with
int <code>getHeight()</code>	Gives the height (in pixels) of the image we are analysing with
int <code>getSelectedLayer()</code>	Returns the index of the shown layer, or the index of the selected
int <code>getSelectedTab()</code>	Returns the index of the tab, selected for the shown layer.
JTabbedPane <code>getTabOnPane()</code>	Returns the tab (JTabbedPane) on the JTabbedPane, that is
VelocityPosMapPlotting <code>getVelocityPosMap()</code>	Returns the tab (JTabbedPane) on the JTabbedPane, that is
SpectrumPlotting <code>getSpecPlot()</code>	Returns the SpectrumPlotting to acces to all the values of
Spectrum1d <code>getsinglepixelspectrum()</code>	Returns the single pixel spectrum as Spectrum1d
SpectrumAvgPlotting <code>getAvgSpecPlot()</code>	Returns the SpectrumAvgPlotting jframe to acces to all the values of
Spectrum1d <code>getAvgspectrum()</code>	that returns the Averaged spectrum from the region spectrum extarction task
ChannelMapPlotting <code>getChannelMapPlot()</code>	Returns the ChannelMapPlotting jframe to acces to all the values of
IntegratedMapDisplay <code>getIntegratedmpaPlot()</code>	Returns the IntegratedMapDisplay jframe to acces to all the values of
RangeExtractionGui <code>getRangeExtraction()</code>	Returns the IntegratedMapDisplay jframe to acces to all the values of
SimpleCube <code>getExtractedCube()</code>	returns the sub-range cube from the range extraction range feature

Methods	
<code>setWaves(Array3dData cube3d)</code>	Method that initialize the Double1d array of the wavelength of
<code>setWaves(Double1d waves1d)</code>	Method that initialize the Double1d array of the wavelength of this CubeSpectrumAnalysisToolbox.
<code>boolean isInImage(MouseEvent e)</code>	Checks whether the given MouseEvent has occurred within the image
<code>boolean isInImage(double pixX, double pixY)</code>	Checks whether the point with given PixelCoordinates (pixX, pixY)
<code>boolean hasWCS()</code>	Returns true if SkyCoordinates are available for the image,
<code>showSpectrum()</code>	Draws a straight line on the image, starting at the point in the
<code>showSpectrumAvg()</code>	Drawn a region on the image, depending of the option selected
<code>showVelocityPosMap()</code>	Draws a straight line on the image, starting at the point in the
<code>showChannelMap()</code>	Draws a straight line on the image, starting at the point in the
<code>showIntegratedMap()</code>	Draws a straight line on the image, starting at the point in the
<code>guiRangeExtract()</code>	Draws a straight line on the image, starting at the point in the
<code>ArrayList getAllFigures()</code>	Returns all ImageFigures used for this CubeSpectrumAnalysisToolbox.
<code>addFigures(ArrayList figures)</code>	Adds the ImageFigures belonging to the selected tab up front to
<code>removeLastFigure(type figures The ImageFigures to add to the list of ImageFigures)</code>	Adds the ImageFigures belonging to the selected tab up front to
<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
<code>createSpaceInTab()</code>	Creates a free space at index 0 in the ArrayList of CanvasFigures,
<code>RangeExtractionGui getRangeextraction()</code>	

API details

Constructors

CubeSpectrumAnalysisToolbox()
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)
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.
Argument
<code>Array3dData array3d</code> [INPUT, MANDATORY]

CubeSpectrumAnalysisToolbox(Array3dData array3d, Double1d wave1d)
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.
Arguments
<code>Array3dData array3d</code> [INPUT, MANDATORY]
<code>Double1d wave1d</code> [INPUT, MANDATORY]

CubeSpectrumAnalysisToolbox(Array3dData array3d, Double1d wave1d, String specDim, String specUnit)
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.
Arguments
<code>Array3dData array3d</code> [INPUT, MANDATORY]
<code>Double1d wave1d</code> [INPUT, MANDATORY]
<code>String specDim</code> [INPUT, MANDATORY]
<code>String specUnit</code> [INPUT, MANDATORY]

CubeSpectrumAnalysisToolbox(type simpleCube)
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 Simplecube you wish to analyse, is displayed.
Argument
<code>type simpleCube</code> [INPUT, MANDATORY, default=no default value]
The SimpleCube containing all the information of an observation you wish to analyse and display with this CubeSpectrumAnalysisToolbox.

Methods

setCube(Array3dData array3d)
to the given ImageDataset.

setCube(Array3dData array3d)
Argument Array3dData array3d [INPUT, MANDATORY]
setCube(SimpleCube simplecube)
to the given SimpleCube. Argument SimpleCube simplecube [INPUT, MANDATORY]
setCube(Array3dData array3d, Double1d wavearray)
to the given ImageDataset. Arguments Array3dData array3d [INPUT, MANDATORY] Double1d wavearray [INPUT, MANDATORY]
addCube(Array3dData array3d)
this CubeSpectrumAnalysisToolbox. Argument Array3dData array3d [INPUT, MANDATORY]
exit()
CubeSpectrumAnalysisToolbox).
closeActiveTab()
closeAllTabs()
setEnabled(boolean enabled)
closing tabs and performing image analysis, of this CubeSpectrumAnalysisToolbox. Argument boolean enabled [INPUT, MANDATORY]
boolean isEnabled()
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()
Return Double1d The Float1d array of the wavelength of this CubeSpectrumAnalysisToolbox.

```
SimpleCube getSimpleCube()
```

Return

SimpleCube

The Float1d array of the wavelength of this CubeSpectrumAnalysisToolbox.

```
JFrame getFrame()
```

Return

JFrame

The JFrame of this CubeSpectrumAnalysisToolbox.

```
JPanel getPanel()
```

Return

JPanel

The JPanel of this CubeSpectrumAnalysisToolbox.

```
JTabbedPane getTabbedPane()
```

Return

JTabbedPane

The JTabbedPane of this CubeSpectrumAnalysisToolbox.

```
SimpleImage getSimpleImage()
```

Return

SimpleImage

Returns the displayed image (SimpleImage).

```
PlotXY getRtSpectrum()
```

Return

PlotXY

Returns the displayed image (ImageDataset).

```
Display getDisplay()
```

this CubeSpectrumAnalysisToolbox.

Return

Display

Returns the display of the ImageDataset you are analysing with this CubeSpectrumAnalysisToolbox.

```
int getWidth()
```

this CubeSpectrumAnalysisToolbox. When we are analysing no image, null is returned.

<code>int getWidth()</code>
<p>Return</p> <p>int</p> <p>Returns the width (in pixels) of the image we are analysing with this CubeSpectrumAnalysisToolbox. When we are analysing no image, null is returned.</p>

<code>int getHeight()</code>
<p>this CubeSpectrumAnalysisToolbox. When we are analysing no image, null is returned.</p> <p>Return</p> <p>int</p> <p>Returns the height (in pixels) of the image we are analysing with this CubeSpectrumAnalysisToolbox. When we are analysing no image, null is returned.</p>

<code>int getSelectedLayer()</code>
<p>tab on the JTabbedPane.</p> <p>Return</p> <p>int</p> <p>Returns the index of the shown layer, or the index of the selected tab on the JTabbedPane.</p>

<code>int getSelectedTab()</code>
<p>Return</p> <p>int</p> <p>The index of the tab, selected for the shown layer.</p>

<code>JTabbedPane getTabOnPane()</code>
<p>selected.</p> <p>Return</p> <p>JTabbedPane</p> <p>Returns the tab (JTabbedPane) on the JTabbedPane, that is selected.</p>

<code>VelocityPosMapPlotting getVelocityPosMap()</code>
<p>selected.</p> <p>Return</p> <p>VelocityPosMapPlotting</p> <p>Returns the tab (JTabbedPane) on the JTabbedPane, that is selected.</p>

<code>SpectrumPlotting getSpecPlot()</code>
<p>single pixel spectrum extraction.</p> <p>Return</p> <p>SpectrumPlotting</p>

SpectrumPlotting `getSpecPlot()`

Returns the SpectrumPlotting to acces to all the values of single pixel spectrum extraction.

Spectrum1d `getsinglepixelspectrum()`

Return

Spectrum1d

Returns the single pixel spectrum as Spectrum1D

SpectrumAvgPlotting `getAvgSpecPlot()`

single pixel spectrum extraction.

Return

SpectrumAvgPlotting

Returns the SpectrumAvgPlotting to acces to all the values of single pixel spectrum extraction.

Spectrum1d `getAvgspectrum()`

Return

Spectrum1d

that returns the Averaged spectrum from the region spectrum extarction task

ChannelMapPlotting `getChannelMapPlot()`

channel map extraction

Return

ChannelMapPlotting

Returns the ChannelMapPlotting jframe to acces to all the values of channel map extraction

IntegratedMapDisplay `getIntegratedmpaPlot()`

channel map extraction

Return

IntegratedMapDisplay

Returns the IntegratedMapDisplay jframe to acces to all the values of channel map extraction

RangeExtractionGui `getRangeExtraction()`

channel map extraction

Return

RangeExtractionGui

Returns the IntegratedMapDisplay jframe to acces to all the values of channel map extraction

SimpleCube `getExtractedCube()`

Return

SimpleCube <code>getExtractedCube()</code>
SimpleCube returns the sub-range cube from the range extraction range feature
setWaves (Array3dData <code>cube3d</code>)
this CubeSpectrumAnalysisToolbox. Argument Array3dData <code>cube3d</code> [INPUT, MANDATORY]
setWaves (Double1d <code>waves1d</code>)
Argument Double1d <code>waves1d</code> [INPUT, MANDATORY]
boolean <code>isInImage</code> (MouseEvent <code>e</code>)
we are analysing with this CubeSpectrumAnalysisToolbox. Argument MouseEvent <code>e</code> [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 <code>isInImage</code> (double <code>pixX</code> , double <code>pixY</code>)
lies in the image we are analysing with this CubeSpectrumAnalysisToolbox. Arguments double <code>pixX</code> [INPUT, MANDATORY] double <code>pixY</code> [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 <code>hasWCS</code> ()
analysed with this CubeSpectrumAnalysisToolbox; false otherwise. Return boolean Returns true if SkyCoordinates are available for the image, analysed with this CubeSpectrumAnalysisToolbox; false otherwise.
showSpectrum ()
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.

showSpectrumAvg()
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()
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 lin. When you click or move outside of the image, no straight line is drawn and no plot is shown.
showChannelMap()
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 lin. When you click or move outside of the image, no straight line is drawn and no plot is shown.
showIntegratedMap()
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 lin. When you click or move outside of the image, no straight line is drawn and no plot is shown.
guiRangeExtract()
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 lin. When you click or move outside of the image, no straight line is drawn and no plot is shown.
ArrayList getAllFigures()
Return ArrayList Returns all ImageFigures used for this CubeSpectrumAnalysisToolbox.
addFigures(ArrayList figures)
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)
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] used for this CubeSpectrumAnalysisToolbox.
updateImage()
removeAllAnnotations()
createSpaceOnTabbedPane()
ArrayLists of ImageFigures, that belong to the image analysis on the shown layer.

<code>createSpaceInTab()</code>

belonging to a new image analysis on the shown layer.

<code>RangeExtractionGui</code> <code>getRangeextraction()</code>

Return


<code>RangeExtractionGui</code>

the RangeExtractionGui

See also

- [Display, PlotXY](#)

3.74. CubicSplineInterpolator

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

Description

Creates a cubic spline interpolation function from a set of knots (x,y),

that can be applied to numeric arrays of rank 1.

The second derivative is assumed to be zero at the end points (i.e. a natural spline).

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) # [1.0,1.21,4.0,6.7599993,9.0,15.210001]
```

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](#)

3.75. 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

The CutLevels task for 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 :

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

```
cutLevelsTask(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)

<code>double percent [INPUT, OPTIONAL, default=99.5]</code>

The percentage of pixels to use in the calculation of the cut levels.


<code>double[] cutLevels [OUTPUT, MANDATORY, default=No default value]</code>

The minimum and maximum cut level

See also

- [???](#)

3.76. database

Full Name:	herschel.ia.history.example.DataBaseTask
Alias:	database
Type:	Java Task - 
Import:	from herschel.ia.history.example import DataBaseTask
Category:	task

Description

A fake fake database access task for testing history.

This task is used in the history demo script.

Example

Example 1: overview compute
<pre>database("select product from t2")</pre>

API Summary

Jython Syntax
<code>database(string)</code>
Properties
<code>String query</code> [INPUT, YES, default=No default value]
<code>Product output</code> [OUTPUT, YES, default=No default value]

Limitations

No computation is actually performed.

Miscellaneous

No miscellaneous

API details

Properties

<code>String query</code> [INPUT, YES, default=No default value]
A database query
<code>Product output</code> [OUTPUT, YES, default=No default value]
The output containing the product.


See also

- [references](#)

History

- 2005-02-24 - NdC: first release.

3.77. 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.

3.78. 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.

3.79. DbPool

Full Name:	herschel.ia.pal.pool.db.DbPool
Type:	Java Class - 
Import:	from herschel.ia.pal.pool.db import DbPool

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).

You will need the logical name of the database you wish the DbPool to work against. By default this will be "hcss.test.database". Please check that the property of the same name points to a valid physical database.

The identifier for the CachedPool (which you may need eg if you want to access that pool remotely) is "db".

For performance reasons, it is recommended to wrap a DbPool around a CachedPool. See the last example below.

Examples

Example 1: Create a DbPool that points to the default logical database of "hcss.test.database"

```
storage=ProductStorage() storage.register(DbPool.getInstance())
```

Example 2: Create a DbPool that points to the logical database of "my.database"

```
storage=ProductStorage() storage.register(DbPool.getInstance("my.database"))
```

Example 3: Untitled

```
Subsequent saving and loading to/from a DbPool ref=storage=save(myproduct)
storage.load(ref)
```

Example 4: Untitled

```
Wrapping a CachedPool around a DbPool
storage.register(CachedPool(DbPool.getInstance()))
```

API Summary

Methods
DbPool <code>getInstance(String database)</code> Creates an instance of a DbPool connected to a logical database
DbPool <code>createInstance(String database)</code> Creates an instance of a DbPool connected to a logical database
DbPool <code>getInstance()</code> Creates an instance of a DbPool connected to the logical database of default name

API details

Methods

DbPool getInstance(**String** database)

Argument

String database [INPUT, MANDATORY, default=The logical name of the database]

"my.test.database"

Return

DbPool

An instance of the DbPool.

DbPool createInstance(**String** database)

Argument

String database [INPUT, MANDATORY, default=The logical name of the database]

"my.test.database"

Return

DbPool

An instance of the DbPool.

DbPool getInstance()


"hcss.test.database"

Return

DbPool

An instance of the DbPool.

3.80. 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
<pre>A=Double2d([[1,2],[3,4]]) print DETERMINANT(A) # -2</pre>

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](#)

3.81. 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.

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.

3.82. 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

```
d = Display(mySimpleImage)
```

Example 2: A basic example on how to Display a Double2d

```
im = Double2d(600, 400)
for i in range(600):
    for j in range(400):
        im.set(i, j, i + j)
d = Display(im)
```

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(Image imds, boolean imageVisible) A Display constructor
Display(Cube imds, boolean imageVisible) A Display constructor
Display(Array2dData data) A constructor to display a Numeric2d.
Display(Array2dData data, boolean imageVisible) A Display constructor

Constructors	
<code>Display(Array3dData data)</code>	A constructor to display a numeric3d
<code>Display(Array3dData data, boolean imageVisible)</code>	A display constructor

Methods	
<code>int getLayerShown()</code>	Returns the number of the shown layer
<code>setVisible(boolean imageVisible)</code>	Toggles the display visible or invisible.
<code>JPanel getComposedComponent()</code>	Returns a panel with all components of this Display on it.
<code>ImageColorbar getColorBarComponent()</code>	Returns the color bar component.
<code>DivaMainImageDisplay getComponent()</code>	Returns the component where the image is displayed.
<code>ImagePanner getPannerComponent()</code>	Returns the panner component with the overview of the image.
<code>StatusPanel getStatusPanelComponent()</code>	Returns the component with the status panel.
<code>setStatusPanelComponent(StatusPanel imageDisplayStatusPanel)</code>	Set a new status panel for the display.
<code>ImageZoom getZoomComponent()</code>	Returns the component with the zoomed view.
<code>setImage(Array2dData imageData)</code>	Set a new numeric2d image.
<code>setImage(Array3dData imageData)</code>	Sets a new numeric3d image.
<code>setImage(Image imds)</code>	Sets a new Image to Display.
<code>setImage(Cube cube)</code>	Sets a new cube to display.
<code>addLayer(Layer layer)</code>	Adds a new layer to the display.
<code>addLayer(Array2dData imageData)</code>	Adds a new layer with numeric2d data
<code>addLayer(Array3dData imageData)</code>	Adds a new layer with numeric3d data
<code>addLayer(Image imds)</code>	Adds a new layer with an Image to the display.
<code>addLayer(Cube cube)</code>	Adds a new layer with a Cube to the display.

Methods	
<code>Layer</code> <code>getLayer()</code>	Returns the shown layer.
<code>Layer</code> <code>getLayer(int i)</code>	Returns the Layer object
<code>showLayer(int i)</code>	Shows a chosen layer.
<code>removeLayer()</code>	Removes the current layer.
<code>removeLayer(int i)</code>	Removes a layer.
<code>updateImage(Array2dData imageData)</code>	Updates the layer with numeric2d data.
<code>updateImage(Image imds)</code>	Updates the layer with an Image.
<code>updateImage(Array3dData imageData)</code>	Updates the layer with numeric3d data.
<code>updateImage(Cube imds)</code>	Updates the layer with a cube.
<code>setBackground(Color bgColor)</code>	Sets the background color.
<code>annotationToolbox()</code>	Fires up the annotation toolbox.
<code>CanvasFigure</code> <code>addAnnotation(String text, double row, double column)</code>	Adds a text annotation.
<code>CanvasFigure</code> <code>addAnnotationWorldCoordinates(String text, double ra, double decl)</code>	Adds a text annotation
<code>CanvasFigure</code> <code>addGreekAnnotation(String text, double row, double column)</code>	Adds a Greek annotation
<code>CanvasFigure</code> <code>addGreekAnnotationWorldCoordinates(String text, double ra, double decl)</code>	Adds a Greek annotation
<code>Font</code> <code>getAnnotationFont()</code>	Returns the default font for annotations.
<code>Font</code> <code>getAnnotationFont(double row, double column)</code>	Returns the font for the specified annotation.
<code>Font</code> <code>getAnnotationFontWorldCoordinates(double ra, double dec)</code>	Returns the font for the specified annotation.
<code>Color</code> <code>getAnnotationFontColor()</code>	Returns the default color for annotations.

Methods
Color <code>getAnnotationFontColor</code> (double row, double column) Returns the color of the specified annotation.
Color <code>getAnnotationFontColorWorldCoordinates</code> (double ra, double dec) Returns the color of the specified annotation.
<code>removeAnnotation</code> (double row, double column) Removes the specified annotation
<code>removeAnnotationWorldCoordinates</code> (double ra, double dec) Remove the specified annotation.
<code>removeAnnotation</code> (CanvasFigure fig) Removes the specified annotation.
<code>removeAnnotations</code> () Removes all annotations.
<code>setAnnotationFont</code> (Font f) Changes the font of all Annotations.
<code>setAnnotationFont</code> (double row, double column, Font f) Changes the font of the specified annotations.
<code>setAnnotationFontWorldCoordinates</code> (double ra, double dec, Font f) Changes the font of the specified annotations.
<code>setAnnotationFont</code> (int size) Changes the font size of all annotations.
<code>setAnnotationFont</code> (double row, double column, int size) Changes the font size of the specified annotations.
<code>setAnnotationFontWorldCoordinates</code> (double ra, double dec, int size) Changes the font size of the specified annotations.
<code>setAnnotationFontColor</code> (Color color) Sets the default color for annotations.
<code>setAnnotationFontColor</code> (double row, double column, Color color) Sets the font color for the specified annotation.
<code>setAnnotationFontColorWorldCoordinates</code> (double ra, double dec, Color color) Sets the font color for the specified annotation.
CanvasFigure <code>addEllipse</code> (double row, double column, double w, double h, float lineWidth, Color color) Adds an ellipse.
<code>addFigure</code> (CanvasFigure fig) Adds a CanvasFigure
CanvasFigure <code>addEllipse</code> (double row, double column, double w, double h, float lineWidth, Color color, double positionAngle) Adds an ellipse
CanvasFigure <code>addCircle</code> (double row, double column, double radius, float lineWidth, Color color)

Methods
<p>Adds a circle</p>
<p>CanvasFigure addLine(double row1, double column1, double row2, double column2, float lineWidth, Color color)</p> <p>Adds a line.</p>
<p>CanvasFigure addPolygon(double[] coords, float lineWidth, Color color)</p> <p>Adds a polygon</p>
<p>CanvasFigure addPolyline(double[] coords, float lineWidth, Color color)</p> <p>Adds a polyline.</p>
<p>CanvasFigure addRectangle(double minRow, double minColumn, double w, double h, float lineWidth, Color color)</p> <p>Adds a rectangle.</p>
<p>ArrayList addImageContour(ImageContour imageContour, ArrayList colors, int minLength)</p> <p>Adds an ImageContour.</p>
<p>ArrayList addImageContour(ImageContour imageContour, ArrayList colors)</p> <p>Adds an ImageContour.</p>
<p>ArrayList addWcsImageContour(ImageContour imageContour, ArrayList colors, int minLength)</p> <p>Add an ImageContour.</p>
<p>ArrayList addWcsImageContour(ImageContour imageContour, ArrayList contourColors)</p> <p>Adds an ImageContour.</p>
<p>ArrayList addContourLevel(ContourLevel contourLevel, Color contourColor, int minimumContourLength)</p> <p>Adds a ContourLevel.</p>
<p>ArrayList addContourLevel(ContourLevel level, Color color)</p> <p>Adds a ContourLevel.</p>
<p>ArrayList addContourLevel(ContourLevel level, Wcs wcs, Color color, int minLength)</p> <p>Adds a ContourLevel.</p>
<p>ArrayList addContourLevel(ContourLevel level, Wcs wcs, Color color)</p> <p>addContourLevel(ContourLevel level, Wcs wcs, Color color)</p>
<p>ImageFigure addContour(Contour contour, Color color, int minLength)</p> <p>Adds a Contour.</p>
<p>ImageFigure addContour(Contour contour, Color color)</p> <p>Adds a Contour</p>
<p>ImageFigure addContour(Contour contour, Wcs wcs, Color color, int minLength)</p> <p>Adds a Contour.</p>
<p>ImageFigure addContour(Contour contour, Wcs wcs, Color color)</p>

Methods	
	Adds a Contour.
<code>ArrayList</code>	<code>showAxes(boolean showAxis)</code> Enables or disables the axes for the displayed image.
<code>ImageAxis</code>	<code>getLeftaxis()</code> Returns the left axis.
<code>ImageAxis</code>	<code>getRightaxis()</code> Returns the right axis.
<code>ImageAxis</code>	<code>getBottomaxis()</code> Returns the bottom axis.
<code>ImageAxis</code>	<code>getTopaxis()</code> Returns the top axis.
<code>ArrayList</code>	<code>getAxes()</code> Returns an array with the axes.
	<code>addAxis(String label, Position orientation)</code> Adds new axis.
	<code>deleteAxis(ImageAxis axis)</code> Deletes the given axis.
	<code>close()</code> Closes the display.
	<code>editColors()</code> Edit the colors using a popup.
	<code>editCutLevels()</code> Edit the cut levels using a popup.
<code>String</code>	<code>getColortable()</code> Returns the name of the color table.
<code>double[]</code>	<code>getCutLevels()</code> Returns the cut levels.
<code>Image</code>	<code>getImage()</code> Returns the displayed image.
<code>double[]</code>	<code>getPixelCoordinates(double c1, double c2)</code> Returns the image coordinates
<code>Number</code>	<code>getIntensity(int row, int column)</code> Returns the intensity of the pixel
<code>Number</code>	<code>getIntensityFromWorldCoordinates(double c1, double c2)</code> Returns the intensity of the pixel.
<code>Unit</code>	<code>getUnit()</code> Returns the unit.
<code>float</code>	<code>getZoomFactor()</code> Returns the used zoom factor.
	<code>printDialog()</code> Opens a printer dialog.

Methods
<code>getPrintControl()</code> Returns the printControl.
<code>createPrintJob()</code> Creates a print job.
<code>PrinterJob getPrintJob()</code> Returns the printer job.
<code>setPrintJob(PrinterJob job)</code> Sets a printer job.
<code>print(String path)</code> Print the displayed image to a ps file.
<code>print(String path, String orientation)</code> Prints the displayed image to a ps file.
<code>save()</code> Pops up a dialog to save your image.
<code>saveAsJPG(String filename)</code> Saves the display as a jpg file.
<code>saveScreenshot(String filename)</code> Save the file as a screenshot.
<code>saveCurrentView(String filename)</code> Saves the current view of the image.
<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>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>

Methods	
	Zooms the image.
<code>zoomIn()</code>	Zooms the image in.
<code>zoomOut()</code>	Zoom the image out.
<code>flipYAxis()</code>	Flips the y axis of the displayed image.
<code>setFlipYAxis(boolean flipAxis)</code>	Sets whether the current image and all newly added images 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>	Sets the depth axis for cubes.
<code>getDepthAxis()</code>	Returns the depth axis for cubes.

API details

Constructors

Display()
The standard constructor for Display. This constructor shows an empty display window.
Display(boolean useAsComponent, boolean imageVisible)
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 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 </pre>

Display(boolean useAsComponent)

```

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)

```

Display(Image imds)

A constructor which immediately displays the given Image.

Argument

Image imds [INPUT, MANDATORY]

Display(Cube cube)

A constructor which immediately displays the given Cube.

Argument

Cube cube [INPUT, MANDATORY]

Display(Image imds, boolean imageVisible)

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 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(Array2dData data)

A constructor which immediately displays the given numeric2d

Argument

Array2dData data [INPUT, MANDATORY]

Display(Array2dData data, boolean imageVisible)

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 which immediately displays the given numeric3d

Display([Array3dData](#) data)

Argument

[Array3dData](#) data [INPUT, MANDATORY]

Display([Array3dData](#) data, boolean imageVisible)

A constructor which immediately displays the given numeric3d

Arguments

[Array3dData](#) data [INPUT, MANDATORY]

boolean imageVisible [INPUT, MANDATORY]

Methods

int getLayerShown()

Returns the number of the layer which is shown.

Return

int

The number of the layer which is shown.

setVisible(boolean imageVisible)

Toggles the imageDisplay visible or invisible. True to make the image visible, False to hide the image.

Argument

boolean imageVisible [INPUT, MANDATORY]

[JPanel](#) getComposedComponent()

Return

[JPanel](#)

Returns a panel with all component of this Display on it.

[ImageColorbar](#) getColorBarComponent()

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](#) getComponent()

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

DivaMainImageDisplay `getComponent()`

Example on how to integrate Display in your own GUI.

```

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 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 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)

Attaches a new status panel to the display.

Argument

StatusPanel imageDisplayStatusPanel [INPUT, MANDATORY]

ImageZoom `getZoomComponent()`

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

The component with the zoomed view to use in your own GUI's

setImage(**Array2dData** imageData)

Sets a new image on the display. A Bool2d, Int2d, Double2d, ... can be used.

Argument

setImage(Array2dData imageData)
<code>Array2dData imageData</code> [INPUT, MANDATORY]
setImage(Array3dData imageData)
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.
Argument
<code>Array3dData imageData</code> [INPUT, MANDATORY]
setImage(Image imds)
Shows a new image on the display. The world-coordinate system of the Image is used.
Argument
<code>Image imds</code> [INPUT, MANDATORY]
setImage(Cube cube)
A Cube is shown on the display. The world-coordinate system of the Cube is used.
Argument
<code>Cube cube</code> [INPUT, MANDATORY]
addLayer(Layer layer)
Adds a new layer to the display. A Layer must be constructed before.
Argument
<code>Layer layer</code> [INPUT, MANDATORY]
addLayer(Array2dData imageData)
Adds a new layer to the display. A Bool2d, Int2d, Double2d, ... can be used.
Argument
<code>Array2dData imageData</code> [INPUT, MANDATORY]
addLayer(Array3dData imageData)
Adds new layers to the display. A Bool3d, Int3d, Double3d, ... can be used.
Argument
<code>Array3dData imageData</code> [INPUT, MANDATORY]
addLayer(Image imds)
Adds a new layer to the display. An Image is added as extra layer of the display.
Argument
<code>Image imds</code> [INPUT, MANDATORY]
addLayer(Cube cube)
Adds a new layer to the display. A Cube is added as extra layer of the display.
Argument
<code>Cube cube</code> [INPUT, MANDATORY]
Layer getLayer()
Returns the Layer that is shown.
Return

Layer <code>getLayer()</code>
<p>Layer</p> <p>The layer that is shown at the moment.</p>
Layer <code>getLayer(int i)</code>
<p>The Layer with the given index is returned.</p> <p>Argument</p> <p><code>int i</code> [INPUT, MANDATORY]</p> <p>Return</p> <p>Layer</p> <p>The layer with the given index.</p>
showLayer(int i)
<p>Shows the Layer with the given index. The Layer with the given index is shown.</p> <p>Argument</p> <p><code>int i</code> [INPUT, MANDATORY]</p>
removeLayer()
<p>Removes the Layer that is shown.</p>
removeLayer(int i)
<p>Removes the given Layer.</p> <p>Argument</p> <p><code>int i</code> [INPUT, MANDATORY]</p>
updateImage(Array2dData imageData)
<p>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.</p> <p>Argument</p> <p><code>Array2dData imageData</code> [INPUT, MANDATORY]</p>
updateImage(Image imds)
<p>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.</p> <p>Argument</p> <p><code>Image imds</code> [INPUT, MANDATORY]</p>
updateImage(Array3dData imageData)
<p>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.</p> <p>Argument</p> <p><code>Array3dData imageData</code> [INPUT, MANDATORY]</p>
updateImage(Cube imds)
<p>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.</p>

updateImage(Cube imds)
Argument Cube imds [INPUT, MANDATORY]
setBackground(Color bgColor)
Changes the background of the image. The background can be black or white. Argument Color bgColor [INPUT, MANDATORY]
annotationToolbox()
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 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)
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)
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

CanvasFigure addGreekAnnotation(**String** text, double row, double column)

The annotation as a CanvasFigure

CanvasFigure addGreekAnnotationWorldCoordinates(**String** text, double ra, double decl)

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 font used for the annotations.

Return

Font

The font used for the annotations.

Font getAnnotationFont(double row, double column)

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 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 used for the text annotations.

Color <code>getAnnotationFontColor()</code>
Return Color The default color used for the text annotations.
Color <code>getAnnotationFontColor(double row, double column)</code>
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 <code>getAnnotationFontColorWorldCoordinates(double ra, double dec)</code>
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 annotation at the given pixel coordinates. Arguments double row [INPUT, MANDATORY] double column [INPUT, MANDATORY]
removeAnnotationWorldCoordinates(double ra, double dec)
Removes the annotation at the given world coordinates. Arguments double ra [INPUT, MANDATORY] double dec [INPUT, MANDATORY]
removeAnnotation(CanvasFigure fig)
Removes the given annotation. Argument CanvasFigure fig [INPUT, MANDATORY]
removeAnnotations()
Removes all the annotations from the display.

setAnnotationFont(Font f)

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 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 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.

Argument

int **size** [INPUT, MANDATORY]

setAnnotationFont(double row, double column, int size)

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 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)

Changes the font color of all annotations.

Argument

Color **color** [INPUT, MANDATORY]

setAnnotationFontColor(double row, double column, Color color)

Changes the font color of the annotations which are located at the given pixel coordinates.


```
setAnnotationFontColor(double row, double column, Color color)
```

Arguments

```
double row [INPUT, MANDATORY]
double column [INPUT, MANDATORY]
Color color [INPUT, MANDATORY]
```

```
setAnnotationFontColorWorldCoordinates(double ra, double dec,
Color color)
```

Changes the font color of the annotations which are located at the given world coordinates.

Arguments

```
double ra [INPUT, MANDATORY]
double dec [INPUT, MANDATORY]
Color color [INPUT, MANDATORY]
```

```
CanvasFigure addEllipse(double row, double column, double w,
double h, float lineWidth, Color color)
```

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 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 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]
```

CanvasFigure addEllipse(double row, double column, double w, double h, float lineWidth, **Color** color, double positionAngle)

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 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 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 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 addPolygon(double[] coords, float lineWidth, Color color)
```

CanvasFigure

The annotation as CanvasFigure

```
CanvasFigure addPolyline(double[] coords, float lineWidth, Color color)
```

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 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

```
ArrayList addImageContour(ImageContour imageContour, ArrayList colors, int minLength)
```

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, ArrayList colors)
```

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 colors [INPUT, MANDATORY]

Return

ArrayList

Returns the drawn Contours as an ArrayList of ImageFigures.

```
ArrayList addWcsImageContour(ImageContour imageContour, ArrayList colors, int minLength)
```

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, ArrayList contourColors)
```

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 addContourLevel(ContourLevel contourLevel, Color contourColor, int minimumContourLength)
```

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 addContourLevel(ContourLevel contourLevel, Color
contourColor, int minimumContourLength)
```

ArrayList

Returns the drawn Contours as an ArrayList of ImageFigures.

```
ArrayList addContourLevel(ContourLevel level, 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]

Color color [INPUT, MANDATORY]

Return

ArrayList

The drawn Contours as an ArrayList of ImageFigures.

```
ArrayList addContourLevel(ContourLevel level, Wcs wcs, Color
color, int minLength)
```

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]

int minLength [INPUT, MANDATORY]

Return

ArrayList

Returns the drawn Contours as an ArrayList of ImageFigures.

```
ArrayList 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.

```
ImageFigure addContour(Contour contour, Color color, int
minLength)
```

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.

```
ImageFigure addContour(Contour contour, Color color, int
minLength)
```

Arguments

```
Contour contour [INPUT, MANDATORY]
Color color [INPUT, MANDATORY]
int minLength [INPUT, MANDATORY]
```

Return

ImageFigure

Returns the drawn Contour as an ImageFigure.

```
ImageFigure addContour(Contour contour, Color color)
```

Draws the given Contour in the given Color on the image and returns the drawn Contour as an ImageFigure.

Arguments

```
Contour contour [INPUT, MANDATORY]
Color color [INPUT, MANDATORY]
```

Return

ImageFigure

Returns the drawn Contour as an ImageFigure.

```
ImageFigure addContour(Contour contour, Wcs wcs, Color color, int
minLength)
```

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.

Arguments

```
Contour contour [INPUT, MANDATORY]
Wcs wcs [INPUT, MANDATORY]
Color color [INPUT, MANDATORY]
int minLength [INPUT, MANDATORY]
```

Return

ImageFigure

Returns the drawn Contour as an ImageFigure.

```
ImageFigure addContour(Contour contour, Wcs wcs, Color color)
```

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 showAxes(boolean showAxis)

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 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 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 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 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. Using methods on the Axes, the appearance can be changed.

Return

ArrayList

An ArrayList with the Axes.

addAxis(String label, Position orientation)
Adds a new axis.
Arguments
String label [INPUT, MANDATORY]
Position orientation [INPUT, MANDATORY]
deleteAxis(ImageAxis axis)
Deletes the given axis.
Argument
ImageAxis axis [INPUT, MANDATORY]
close()
Closes the display and frees the memory.
editColors()
A windows opens where you can change the color table, intensity and scale.
editCutLevels()
A windows opens where you can set the cut levels of the image.
String getColortable()
Returns the name of the color table
Return
String
The name of the color table
double[] getCutLevels()
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.
Return
Image
The displayed Image
double[] getPixelCoordinates(double c1, double c2)
Returns the image coordinates of the given world coordinates
Arguments
double c1 [INPUT, MANDATORY]
double c2 [INPUT, MANDATORY]
Return

double[] <code>getPixelCoordinates(double c1, double c2)</code>
<p>double[]</p> <p>The corresponding image coordinates as a 2 dimensional array of doubles, the first one describing the row and the second the column.</p>
Number <code>getIntensity(int row, int column)</code>
<p>Returns the intensity of the pixel with given pixel coordinates</p> <p>Arguments</p> <p>int row [INPUT, MANDATORY]</p> <p>int column [INPUT, MANDATORY]</p> <p>Return</p> <p>Number</p> <p>The intensity of the given pixel</p>
Number <code>getIntensityFromWorldCoordinates(double c1, double c2)</code>
<p>Returns the intensity of the pixel with given world coordinates.</p> <p>Arguments</p> <p>double c1 [INPUT, MANDATORY]</p> <p>double c2 [INPUT, MANDATORY]</p> <p>Return</p> <p>Number</p> <p>double The intensity of the given pixel.</p>
Unit <code>getUnit()</code>
<p>Returns the unit of the image</p> <p>Return</p> <p>Unit</p> <p>The unit of the image.</p>
float <code>getZoomFactor()</code>
<p>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</p> <p>Return</p> <p>float</p> <p>The zoomfactor of the displayed image.</p>
printDialog()
<p>A dialog is opened where you can choice the printer, number of copies, page setup and appearance.</p>
getPrintControl()
<p>Returns the printControl to make it possible to print using the command line.</p> <p>Example</p>

getPrintControl()

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.

PrinterJob getPrintJob()

Returns the printer job.

Return

PrinterJob

The printerJob.

setPrintJob(PrinterJob job)

Set a given printer job.

Argument

PrinterJob job [INPUT, MANDATORY]

print(String path)

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(String path, String orientation)

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 file chooser and saves the image as fits, jpeg, tiff, png or bmp file.

saveAsJPG(String filename)

Saves the display as a jpg file.

Argument

String filename [INPUT, MANDATORY]

saveScreenshot(String filename)

Saves the file as screenshot. Saves the display as jpg, png, tiff, bmp or FITS.

saveScreenshot(String filename)
Argument String filename [INPUT, MANDATORY]
saveCurrentView(String filename)
Saves the view as screenshot. The annotations are also saved!
Argument String filename [INPUT, MANDATORY]
setColortable(String colortableName)
Sets the color table of the displayed image.
Argument String colortableName [INPUT, MANDATORY]
setColortable(String colortableName, String intensityName)
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 colortable Sets the colortable of the displayed image
Arguments String colortableName [INPUT, MANDATORY] String intensityName [INPUT, MANDATORY] String scaleName [INPUT, MANDATORY]
setCutLevels(double percent)
Sets the cut level of the displayed image.
Argument double percent [INPUT, MANDATORY]
setCutLevels(double min, double max)
Sets the cut level of the displayed image.
Arguments double min [INPUT, MANDATORY] double max [INPUT, MANDATORY]
setCutLevels()
Sets the cut level of the displayed image.
setCutLevels(double[] minmax)
Sets the cut level of the displayed image.
Argument double[] minmax [INPUT, MANDATORY]

setUnit(Unit u)
Sets the unit of the displayed image.
Argument
Unit u [INPUT, MANDATORY]
setZoomFactor(float zoomFactor)
Sets the zoomfactor of the displayed image.
Argument
float zoomFactor [INPUT, MANDATORY]
zoom(double row, double column, float zoomFactor)
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)
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 displayed image. The zoom factor changes the following way : ..., 1/4, 1/3, 1/2, 1, 2, 3, 4, ...
zoomOut()
Zooms the displayed image out. The zoom factor changes the following way : ..., 4, 3, 2, 1, 1/2, 1/3, 1/4, ...
flipYAxis()
If this method is called, the current axis is flipped and all newly added images will have a flipped Y axis. If the displayed image has the keyword flipy, this method will have no effect. The standard way of displaying an image is that the lowest row is displayed at the top.
setFlipYAxis(boolean flipAxis)
This method sets if all newly added images should have a flipped Y axis. The current image will also be flipped. The standard way of displaying an image is that the lowest row is displayed at the top.
Argument
boolean flipAxis [INPUT, MANDATORY]
getFlipYAxis()
Returns true if the current and all newly added images will be flipped.

isFlipped()

Returns true if the current layer is flipped.

setDepthAxis(int depthAxis)

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 which axis is the depth axis.

See also

- [SimpleImage](#)
- [SimpleCube](#)

3.83. 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

```
# 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

```
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</code> <code>logger</code> [INPUT, No, default= <code>java.util.logging.Logger.getLogger("")</code>]
--

The log to display the messages into the window

<code>String</code> <code>option</code> [INPUT, No, default= <code>"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</code> <code>size</code> [INPUT, No, default= <code>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

3.84. displayQCLog

Full Name:	herschel.ia.qcp.DisplayQCLogTask
Alias:	displayQCLog
Type:	Java Task - 
Import:	from herschel.ia.qcp import DisplayQCLogTask
Category:	task

Description

Open a window displaying QCILogs messages organized in a product

The show task is used to open a log window handler and assign it to a log

Example

Example 1: display the QCLog
<pre>displayQCLog()</pre>

Limitations

No time buffering, logs could clutter the java window system


Miscellaneous

No miscellaneous

History

- 2007-03-22 - NdC: first release

3.85. displayQCLog

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

Description

Open a window displaying QCILogs messages organized in a product

The show task is used to open a log window handler and assign it to a log

Example

Example 1: display the QCLog
<pre>displayQCLog()</pre>

Limitations

No time buffering, logs could clutter the java window system


Miscellaneous

No miscellaneous

History

- 2007-03-22 - NdC: first release

3.86. 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 included in a spectrum container by a scalar or for dividing 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 pairwise mode, the task iterates synchronously 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.

In either mode, you can specify the segments you would like to process. See the 'segments' (for the scalar mode) or the 'segments1', 'segments2' parameters for the pair-wise mode.

For scalar mode, you have advanced selection functionality in place using the selection models as already discussed in the SpectrumTask.

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).

For further details see the (abstract) SpectrumTask in the same package.

Example

Example 1: from Jide:

```
from herschel.ia.toolbox.spectrum import DivideSpectrum
divide = DivideSpectrum()
dividedByFactor = divide(ds=spectra, param=2.1)
dividedPairWise = divide(ds1=spectral1, ds2=spectra2)
```

API Summary

Properties
SpectrumContainer ds [INPUT, OPTIONAL, default=no default value.]
Double param [INPUT, OPTIONAL, default=no default value.]
SelectionModel selection [INPUT, OPTIONAL, default=no default value.]
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.]

Properties
SpectrumContainer ds1 [INPUT, OPTIONAL, default=no default value.]
SpectrumContainer ds2 [INPUT, OPTIONAL, default=no default value.]
SegmentSelection segments1 [INPUT, OPTIONAL, default=no default value.]
SegmentSelection 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 many-to-one operation (eg avg) or as scalar operation.
Double param [INPUT, OPTIONAL, default=no default value.]
The scalar parameter to be considered when the task is used as scalar operation.
SelectionModel selection [INPUT, OPTIONAL, default=no default value.]
Specify an instance of any kind of SelectionModel here.
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.
SegmentSelection segments1 [INPUT, OPTIONAL, default=no default value.]
SegmentSelection to be associated with 'ds1'.

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

SegmentSelection 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

3.87. 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)

3.88. 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)

3.89. 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)

3.90. 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)

3.91. 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)

3.92. 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.getD() print evd.getV()</pre>

API Summary

Jython Syntax
<pre>A=Double2d() evd=A.apply(EigenvalueDecomposition())</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.

3.93. EllipseHistogramExplorer

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

Description

An explorer for EllipseHistogramProducts.

API Summary

Constructors
<code>EllipseHistogramExplorer()</code> The constructor of a new EllipseHistogramExplorer.
<code>EllipseHistogramExplorer(Object object)</code> The constructor of a new EllipseHistogramExplorer associated
Methods
<code>String getName()</code> Returns the name for this EllipseHistogramExplorer.
<code>String getDescription()</code> Returns the description for this EllipseHistogramExplorer.
<code>boolean canHandle(Class className)</code> Checks whether this EllipseHistogramExplorer can handle objects of the
<code>setObject(Object object)</code> Sets the object for this EllipseHistogramExplorer to the given object.
<code>EllipseHistogramProduct getObject()</code> Returns the object for this EllipseHistogramExplorer.
<code>JTable getParameterTable()</code> Returns the parameter table for this

API details

Constructors

<code>EllipseHistogramExplorer()</code>
<code>EllipseHistogramExplorer(Object object)</code> with the given object.
Argument <code>Object object</code> [INPUT, MANDATORY]

Methods

<code>String getName()</code>
Return

String getName()
String Returns the name for this EllipseHistogramExplorer.
String getDescription()
Return String Returns the description for this EllipseHistogramExplorer.
boolean canHandle(Class className)
given class. Argument Class className [INPUT, MANDATORY] Return boolean Returns true if this EllipseHistogramExplorer can handle objects of the given class; false otherwise.
setObject(Object object)
Argument Object object [INPUT, MANDATORY]
EllipseHistogramProduct getObject()
Return EllipseHistogramProduct Returns the object for this EllipseHistogramExplorer.
JTable getParameterTable()
EllipseHistogramExplorer. Return JTable Returns the parameter table for this EllipseHistogramExplorer.

3.94. EllipseHistogramPanel

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

Description

A panel for the EllipseHistogramPanel.

API Summary

Constructor
<code>EllipseHistogramPanel()</code> The construction of a new EllipseHistogramPanel.
Methods
<code>drawFigure()</code> Draws the ellipse on the image associated with this
<code>updateFigure()</code> Updates the ellipse associated with this
<code>trigger()</code> Triggers the execution of the task associated
<code>updateHistogram()</code> Updates the histogram associated with this

API details

Constructor

<code>EllipseHistogramPanel()</code>

Methods

<code>drawFigure()</code>
EllipseHistogramPanel.
<code>updateFigure()</code>
EllipseHistogramPanel.
<code>trigger()</code>
with this EllipseHistogramPanel.
<code>updateHistogram()</code>
EllipseHistogramPanel.

3.95. 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><code>DoubleId</code> getCenterPixelCoordinates()</p> <p>Returns the center for this EllipseHistogramProduct in</p>
<p><code>StringId</code> getCenterSkyCoordinates()</p> <p>Returns the center for this EllipseHistogramProduct in</p>
<p><code>double</code> getWidthPixels()</p> <p>Returns the width for this EllipseHistogramProduct in pixels.</p>
<p><code>double</code> getWidthArcsec()</p> <p>Returns the width for this EllipseHistogramProduct in arcsec.</p>
<p><code>double</code> getHeightPixels()</p> <p>Returns the height for this EllipseHistogramProduct in pixels.</p>
<p><code>double</code> getHeightArcsec()</p> <p>Returns the height for this EllipseHistogramProduct in arcsec.</p>

API details

Constructor

<code>EllipseHistogramProduct()</code>
--

Methods

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)
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)
given dimensions in pixels.
Arguments
double widthPixels [INPUT, MANDATORY]
double heightPixels [INPUT, MANDATORY]

setDimensions(double widthPixels, double heightPixels, double widthArcsec, double heightArcsec)
given dimensions in pixels and arcsec.
Arguments
double widthPixels [INPUT, MANDATORY]
double heightPixels [INPUT, MANDATORY]
double widthArcsec [INPUT, MANDATORY]
double heightArcsec [INPUT, MANDATORY]

Double1d getCenterPixelCoordinates()
pixel coordinates.
Return
Double1d
Returns the center for this EllipseHistogramProduct in pixel coordinates.

String1d getCenterSkyCoordinates()
sky coordinates.
Return
String1d
Returns the center for this EllipseHistogramProduct in sky coordinates.

double getWidthPixels()**Return****double**

Returns the width for this EllipseHistogramProduct in pixels.

double getWidthArcsec()**Return****double**

Returns the width for this EllipseHistogramProduct in arcsec.


double getHeightPixels()**Return****double**

Returns the height for this EllipseHistogramProduct in pixels.

double getHeightArcsec()**Return****double**

Returns the height for this EllipseHistogramProduct in arcsec.

3.96. 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 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 : "centerRA"]
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.

3.97. 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

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 Complementary Error function of the corresponding element of the input array.

See also

- [Erf](#)

3.98. 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 # [-0.999999999984626,-0.8427007900291826,-0.5204998760683841, 0.0,0.5204998760683841,0.8427007900291826,0.999999999984626]</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](#)

3.99. 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
<pre>x=Double1d([1,2,3,4]) print EXP10(x) # [10.0, 100.0, 1000.0, 10000.0]</pre>

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](#)

3.100. 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
<pre>x=Double1d([0,1]) print LOG(EXP(x)) # [0.0,1.0]</pre>

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](#)

3.101. 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](#)

3.102. 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 a container.

Different possibilities are available for specifying the ranges:

- Array of Range which specifies for each sub-segment included in the point spectra the range of channel numbers to be extracted.
- Frequency intervals by specifying a DoubleId of minimum frequencies ('minFreq') and a DoubleId of maximum frequencies ('maxFreq'). The frequency range for the i-th segment in the result is then given by the interval [minFreq.get(i), maxFreq.get(i)]. Extracting by 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 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 sub-segments does not need to coincide. For each of the sub-segments, a non-trivial overlap with one of the frequency intervals will lead to a sub-segment in the result container.

Example

Example 1: from Jide

```
from herschel.ia.toolbox.spectrum import ExtractFreqRanges
extract = ExtractFreqRanges()
min = DoubleId([4100, 4600, 5100, 6100, 7100])
max = DoubleId([4400, 4900, 5900, 6900, 7900])
slices = extract(ds=spectra, min=min, max=max)
```

API Summary

Properties
SpectrumContainer ds [INPUT, OPTIONAL, default=no default value.]
SpectrumContainer result [OUTPUT, OPTIONAL, default=no default value.]
Range[] ranges [INPUT, OPTIONAL, default=no default value.]
DoubleId minFreq [INPUT, OPTIONAL, default=no default value.]
DoubleId maxFreq [INPUT, OPTIONAL, default=no default value.]

API details

Properties

SpectrumContainer ds [INPUT, OPTIONAL, default=no default value.]
--

The input container to be processed.

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

The resulting container created by this task.

Range[] ranges [INPUT, OPTIONAL, default=no default value.]
--

The array of Range (range of channel numbers for each sub-segment) to be extracted.

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.

History

- 2008-01-29 - meli: initial.
- 2008-03-16 - meli: major refactoring of the toolbox.

3.103. ExtractRegionPixelSpectrumTask

Full Name:	herschel.ia.toolbox.cube.ExtractRegionPixelSpectrumTask
Type:	Java Task - 
Import:	from herschel.ia.toolbox.cube import ExtractRegionPixelSpectrumTask

Example

Example 1: How to use the Single pixel spectrum extraction

```
singlePixExtraction = ExtractSinglePixelSpectrumTask();
singlePixExtraction.setValue("simplecube",Cube)
singlePixExtraction.setValue("wholeImg",False)
singlePixExtraction.setValue("posArray",arrayofpixel)
singlePixExtraction.execute()
spectrum =(DoubleId) singlePixExtraction.getValue("spectrum")
spectrum Id spectrumId =(DoubleId) singlePixExtraction.getValue("spectrum")
{@link #__abs__() } futur
```

API Summary

Properties
SimpleCube simplecube [INPUT, MANDATORY, default=no default value]
Boolean wholeImg [INPUT, MANDATORY, default=no default value]
Double2d posArray [INPUT, MANDATORY, default=no default value]
Integer nbPixels [INPUT, MANDATORY, default=no default value]
DoubleId spectrum [OUTUT, MANDATORY, default=no default value]
float totalWeight [OUTUT, MANDATORY, default=no default value]

API details

Properties

SimpleCube simplecube [INPUT, MANDATORY, default=no default value]
The Cube containing the spectra
Boolean wholeImg [INPUT, MANDATORY, 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
Integer nbPixels [INPUT, MANDATORY, default=no default value]
The number of pixel to be read
DoubleId spectrum [OUTUT, MANDATORY, default=no default value]
The spectrum extracted at the given position


<code>float totalWeight [OUTUT, MANDATORY, default=no default value]</code>

the 'weight' of the spectrum , ne exact number of pixel from which the spectrum was read
--

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

3.104. ExtractSinglePixelSpectrumTask

Full Name:	herschel.ia.toolbox.cube.ExtractSinglePixelSpectrumTask
Type:	Java Task - 
Import:	from herschel.ia.toolbox.cube import ExtractSinglePixelSpectrumTask

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]
DoubleId spectrum [OUTUT, MANDATORY, default=no default value]
SpectrumId spectrum [OUTUT, 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 bew read.
Integer posY [INPUT, MANDATORY, default=no default value]
The Y coordinate of the pixel to bew read.
DoubleId spectrum [OUTUT, MANDATORY, default=no default value]
The spectrum extracted at the given position
SpectrumId spectrum [OUTUT, MANDATORY, default=no default value]
The spectrum extracted at the given position in a SpectrumId , will become the only output

History

- 2008-06-17 - AG: first complet 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

3.105. 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.

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.

3.106. 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](#)

3.107. FilteringTask

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

Description

An abstract Task to smooth an image using a filter. The subclasses/

subtasks are AverageFilteringTask and MedianFilteringTask, using the average and the median filter.

API Summary


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

API details

Properties

Image image [INPUT, MANDATORY, default=No default value]
The image before filtering.
Image filtered [OUTPUT, MANDATORY, default=No default value]
The image after filtering.

3.108. FilterNoiseTask

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

Description

A Task to make a noise image of an image, using filtering.

API Summary


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

API details

Properties

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

3.109. 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.

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 an externally generated FITS file into a Product:

```
from herschel.ia.io.fits import FitsArchive
# setup a new FitsArchive, but change the reader type
fits=FitsArchive(reader=FitsArchive.STANDARD_READER)
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
<code>FitsReader HCSS_READER</code> HCSS FITS Reader
<code>FitsReader STANDARD_READER</code> Generic FITS Reader
Methods
<code>Product load(String name)</code> Loads a Product from a FITS file.
<code>Product load(String name, boolean handleMissingClasses)</code> Loads a Product from a FITS file.
<code>Product load(InputStream is)</code> Loads a Product from a FITS InputStream.
<code>Product load(InputStream is, boolean handleMissingClasses)</code> Loads a Product from a FITS InputStream.
<code>save(String name, [Optionally derived] Product. product)</code> Saves a Product to a FITS file.

API details

Fields

<code>FitsReader HCSS_READER</code>
A FITS reader that expects a FITS format that was produced by this archive implementation. It can handle nested structures, derived datasets and quantities of the data within FITS. This is the default reader when you create a FitsArchive. Examples creating a new FitsArchive <pre>fits=FitsArchive() # default reader=FitsArchive.HCSS_READER</pre> creating a new FitsArchive <pre>fits=FitsArchive(reader=FitsArchive.HCSS_READER)</pre>
<code>FitsReader STANDARD_READER</code>
A generic FITS reader for FITS files that are not created by the HCSS FitsArchive.

FitsReader STANDARD_READER

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.

Examples

import data from an externally generated FITS file:

```
fits=FitsArchive(reader=FitsArchive.STANDARD_READER)
product=fits.load("external.fits")
```

reusing this FitsArchive but changing the reader:

```
fits.reader=fits.HCSS_READER
product=fits.load("hcss.fits")
```

Methods

Product load(String name)

Loads a Product from a FITS file using the current reader.

Argument

String name [INPUT, MANDATORY, default=no default value]
Name of the FITS file

Return

Product

An object of the herschel.ia.dataset.Product family.

Product load(String name, boolean handleMissingClasses)

Loads a Product from a FITS file using the current reader.

Arguments

String name [INPUT, MANDATORY, default=no default value]
Name of 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.

Product load(InputStream is)

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 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 Hess 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 with sufficient format informat 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.

3.110. 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 = Double1d([ 0,0.5,1,2, 3, 4, 5, 6, 10, 12])
y = Double1d([-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 = Double1d([ 0,0.5,1,2, 3, 4, 5, 6, 10, 12])
y = Double1d([-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

Double1d of abscissas x [INPUT, MANDATORY, default=no default value]
Double1d 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

Double1d of abscissas x [INPUT, MANDATORY, default=no default value]

Abscissas; 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.

3.111. Fitter

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

Description

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.

Example


Example 1: The fit/demo directory contains worked

```
<a href="../../../ia/numeric/toolbox/fit/demo/jython.html">examples</a>  
on almost all aspects of of the package.
```

Limitations

The calculation of the evidence is an Gaussian approximation which is only exact for linear models with a fixed scale.

3.112. 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.

Example

Example 1: FitterTask

```
<pre>
from herschel.hifi.generic.task import FitterTask
# Assume that `tt` and/or `data` are Double1d'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]

Properties
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
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)
Doubleld 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.
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 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.
Doubleld initialpars [INPUT, OPTIONAL, default=from model]
Initial parameters in iterative fitters.
Doubleld highlimits [INPUT, OPTIONAL, default=null]
Upper limits of the parameters (only in AmoebaFitter)
Doubleld lowlimits [INPUT, OPTIONAL, default=null]
Lower limits of the parameters (only in AmoebaFitter)
Intld keepfixed [INPUT, OPTIONAL, default=null]
Keep these parameters fixed. (Parameter numbering starts at 0)
Doubleld 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

3.113. FixedMask

Full Name:	herschel.ia.numeric.toolbox.mask.FixedMask
Type:	Java Class - 
Import:	from herschel.ia.numeric.toolbox.mask import FixedMask

Description

Class for mask handling, where each mask element is fixed at a specific bit offset position.

The implementation is therefore limited to 64 different definitions of a mask element, corresponding to each bit of a long integer. However, all integer types are supported (byte/short/int/long). Note that the mask data itself is not stored in this class.

Example

Example 1: Define some mask bits and use them for mask manipulations.

```
# Create the mask objects
MASTER = FixedMask (0, "Master")
NOISY = FixedMask (2, "Noisy")
GLITCH = FixedMask (4, "Glitch")
# Masks can be printed:
print MASTER
# Master @ 0 = 1
print GLITCH
# Glitch @ 4 = 16
#
# They can be set:
x = 0
x = GLITCH.set (x)
x = MASTER.set (x)
print x
# 17
# and unset:
x = GLITCH.unset (x)
print x
# 1
# and tested
print MASTER.isSet(x)
# 1
print GLITCH.isSet(x)
# 0
#
# You can also use them with the numeric library:
a = Int1d.range(10)
print a.where (MASTER)
# [1,3,5,7,9]
print a.where (MASTER | NOISY)
# [1,3,4,5,6,7,9]
print MASTER.isSet (a)
# [false,true,false,true,false,true,false,true]
print MASTER.set (a)
# [1,1,3,3,5,5,7,7,9,9]
print MASTER.unset (a)
# [0,0,2,2,4,4,6,6,8,8]
```

3.114. 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, SlicedCube 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	
<code>Flag copy()</code>	The copy method returns a new Flag which is a copy of the original flag
<code>addFlagType(String flagType, String description)</code>	Defines a new flag type.
<code>StringId getFlagTypes()</code>	Give a list with the defined flag types.
<code>boolean hasFlag(String name)</code>	Checks if the flag exists.
<code>AbstractArrayData getFlag(String flagType)</code>	Gives the flag for this specific flag type.
<code>AbstractArrayData getFlag()</code>	Gives the flag.
<code>setFlag(String flagType, AbstractArrayData flag)</code>	Sets the flag for this specific flag type.
<code>setFlag(String flagType, int index, boolean flag)</code>	Sets the flag of a certain pixel.
<code>setFlag(String flagType, int row, int column, boolean flag)</code>	Sets the flag of a certain pixel.
<code>setFlag(String flagType, int index1, int index2, int index3, boolean flag)</code>	Sets the flag of a certain pixel.
<code>setFlag(String flagType, int index1, int index2, int index3, int index4, boolean flag)</code>	Sets the flag of a certain pixel.
<code>setFlag(String flagType, int index1, int index2, int index3, int index4, int index5, boolean flag)</code>	Sets the flag of a certain pixel.

API details

Constructors

Flag()
A constructor for a flag. After construction, 1 flagtype is available : UNVALID
Flag(int k)
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
Argument
<code>int k</code> [INPUT, MANDATORY]
Example
Typical example on how to create a one-dimensional Flag.
<pre>flag = Flag(100)</pre>

Flag(int k, int l)

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)

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 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 = Flag(100, 50, 75, 125)
```

Flag(int k, int l, int m, int n, int o)

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]
```

```
Flag(int k, int l, int m, int n, int o)
```

Example

Typical example on how to create a 5-dimensional Flag.

```
flag = Flag(100, 50, 75, 125, 25)
```

```
Flag(Flag flag)
```

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

Return

```
Flag
```

Dataset A new Flag which is a copy of the current Flag.

```
addFlagType(String flagType, String description)
```

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 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 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 as an AbstractArrayData. Depending on the dimensions of the flag, a Boolnd is returned with the flag for a given flag type.

Argument

AbstractArrayData getFlag(String flagType)

`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 as an AbstractArrayData. Depending on the dimensions of the flag, a Boolnd is returned with the flag for a given flag type.

Return

AbstractArrayData

A Boolnd (depending on the dimension of the flag), describing the flag

setFlag(String flagType, AbstractArrayData flag)

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)

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)

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)

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)
```

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)
```

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](#)

3.115. 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.

3.116. 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 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.

3.117. 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)

3.118. 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)

3.119. 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)

3.120. 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)

3.121. 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)

3.122. 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](#)

3.123. 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

- [Querying](#)

3.124. 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](#)

3.125. 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
<pre>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]]</pre>

API Summary

Jython Syntax
<code><y>=GammaP(a)(<x>)</code>
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](#)

3.126. 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
<pre>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]</pre>

API Summary

Jython Syntax
<code><y>=GammaQ(a)(<x>)</code>
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](#)

3.127. 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.

Example

Example 1: GaussFitTask

```
<pre>
from herschel.hifi.generic.task import GaussFitTask
# Assume that `tt` and/or `data` are DoubleI'd'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.
</pre>
```

API Summary

Jython Syntax

see example below

Properties

NumericData **x** [INPUT, false, default=null]

DoubleArray **y** [INPUT, true, default=no]

Boolean **indexview** [INPUT, OPTIONAL, default=true]

DoubleArray **weights** [INPUT, false, default=null]

String **model** [INPUT, true, default=no]

ArrayI'dData **modelarg** [INPUT, false, default=null]

Properties
String fitter [INPUT, true, default=no]
String autoinit [INPUT, false, default=""]
Integer smooth [INPUT, false, default=1]
DoubleId result [OUTPUT, false, default=n/a]
DoubleId parameters [OUTPUT, false, default=n/a]
DoubleId stdev [OUTPUT, false, default=n/a]
Double chisq [OUTPUT, false, default=n/a]
DoubleArray yfit [OUTPUT, false, default=n/a]
DoubleArray yband [OUTPUT, false, default=n/a]
DoubleId prior [INPUT, false, default=null]
Double evidence [OUTPUT, false, default=n/a]
Double scale [OUTPUT, false, default=n/a]
Boolean auto [INPUT, false, default=false]
Double fixed [INPUT, false, default=1.0]
Double mixed [INPUT, false, default=0.0]
Double tolerance [INPUT, false, default=0.01]
Integer iterations [INPUT, false, default=10000]
Double temperature [INPUT, false, default=0.0]
Double cooling [INPUT, false, default=0.95]
Integer tempsteps [INPUT, false, default=100]
DoubleId initialpars [INPUT, false, default=from model]
DoubleId highlimits [INPUT, false, default=null]
DoubleId lowlimits [INPUT, false, default=null]
IntId keepfixed [INPUT, false, default=null]
DoubleId fixedvalues [INPUT, false, default=null]
Boolean gui [INPUT, No mandatory, default=false]

Limitations

Not everything which is possible using the package directly is accessible via this task.

API details

Properties

NumericData x [INPUT, false, 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, true, default=no]
The data to be fitted. It can be more-dimensional.
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, false, 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.
String model [INPUT, true, default=no]
The model to be fitted. Default: GaussModel (for 1d data) or Gauss2DModel (for 2d data)
Array1dData modelarg [INPUT, false, default=null]
Arguments, if any, needed in the Constructor of the model.
String fitter [INPUT, true, default=no]
Fitter to be used. Default: LevenbergMarquardtFitter
String autoinit [INPUT, false, default=""]
Automated search for initial params. options: "high", "low", "center"
Integer smooth [INPUT, false, default=1]
Smooth data with BoxCarFilter before auto-search. smooth > 1.
Double1d result [OUTPUT, false, default=n/a]
The parameters of the fit.
Double1d parameters [OUTPUT, false, default=n/a]
The parameters of the fit. (same as result)
Double1d stdev [OUTPUT, false, default=n/a]
The standard deviations pertaining to the parameters.
Double chisq [OUTPUT, false, default=n/a]
Chi-squared of the fit.
DoubleArray yfit [OUTPUT, false, default=n/a]
An array of the same size as the data, containing the fitted values
DoubleArray yband [OUTPUT, false, default=n/a]
An array of the same size as the data, containing a 1-# MonteCarlo error at each point.
Double1d prior [INPUT, false, default=null]
Prior ranges for the parameters, needed when the evidence is requested.
Double evidence [OUTPUT, false, 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, false, default=n/a]
Noise scale of the data (when auto or mixed is selected)
Boolean auto [INPUT, false, default=false]
Select automatic noise scaling.

Double fixed [INPUT, false, default=1.0]
Fixed noise scale.
Double mixed [INPUT, false, default=0.0]
Automatic noise scaling with a minimum.
Double tolerance [INPUT, false, default=0.01]
Stopping criterion for iterative fitting.
Integer iterations [INPUT, false, default=10000]
Maximum number of iterations.
Double temperature [INPUT, false, default=0.0]
Starting temperature in annealing amoeba fitting.
Double cooling [INPUT, false, default=0.95]
Cooling step in annealing amoeba fitting.
Integer tempsteps [INPUT, false, default=100]
Number of exploratory steps at each temperature.
DoubleId initialpars [INPUT, false, default=from model]
Initial parameters in iterative fitters.
DoubleId highlimits [INPUT, false, default=null]
Upper limits of the parameters (only in AmoebaFitter)
DoubleId lowlimits [INPUT, false, default=null]
Lower limits of the parameters (only in AmoebaFitter)
IntId keepfixed [INPUT, false, default=null]
Keep these parameters fixed. (Parameter numbering starts at 0)
DoubleId fixedvalues [INPUT, false, default=null]
Values at which the parameters should be kept.
Boolean gui [INPUT, No mandatory, default=false]
Allows to handle this task using a gui.

History

- 15-10-2006 DK

3.128. 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
<pre>real sigma [INPUT, MANDATORY, default=no default value]</pre>
<pre>boolean center [INPUT, NOT_MANDATORY, default=false]</pre>
<pre>Convolution.ZEROES CIRCULAR REPEAT center [INPUT, NOT_MANDATORY, default=Convolution.ZEROES]</pre>

API details

Properties

<pre>real sigma [INPUT, MANDATORY, default=no default value]</pre>
It must be a real.
<pre>boolean center [INPUT, NOT_MANDATORY, default=false]</pre>
Set center to true or false.
<pre>Convolution.ZEROES CIRCULAR REPEAT center [INPUT, NOT_MANDATORY, default=Convolution.ZEROES]</pre>
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](#)

3.129. 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.

3.130. 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.

3.131. 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)
```

3.132. 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() <br>
help(item)
```

Property

Object <code>item</code> [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 <code>item</code> [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.

3.133. Histogram

Full Name:	herschel.ia.gui.image.Histogram
Type:	Java Class - 
Import:	from herschel.ia.gui.image 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	
<code>Histogram(ImageAnalysisToolbox toolbox)</code>	The standard constructor for Histogram. This constructor creates a window
<code>Histogram(boolean useAsComponent, ImageAnalysisToolbox toolbox)</code>	Constructor for Histogram. When the new Histogram is
Methods	
<code>PlotXY getPlot()</code>	Returns the PlotXY shown in this Histogram.
<code>JPanel getPanel()</code>	Returns the panel of this Histogram.
<code>int getUsedArea()</code>	Returns the used type of area for this Histogram.
<code>int getUsedRegion()</code>	Returns the used type of region for this Histogram.
<code>int getNumberOfEdges()</code>	Returns the number of edges, if the region of interest is a
<code>int getUsedCutLevels()</code>	Returns the used type of cut levels for this Histogram.
<code>double[] getCutlevels()</code>	Returns the used cut levels (min, max) for this Histogram.
<code>int getNumberOfBins()</code>	Returns the used number of bins for this Histogram.
<code>int getBinWidth()</code>	Returns the width of the bins for this Histogram.
<code>int getNbOfClicks()</code>	Returns the number of valid clicks (i.e. in the image) you made.
<code>boolean checkInput()</code>	Returns true if the given input is correct; false otherwise. The

Methods
boolean <code>checkArea()</code> Checks whether the input area is correct. The error message is
boolean <code>checkParameters()</code> Checks whether the input parameters are correct. The error message
boolean <code>checkCutLevels()</code> Checks whether the input cut levels are correct. The error message
boolean <code>checkNumberOfBins()</code> Checks whether the input number of bins is correct. The error
ArrayList <code>getImageFigures()</code> Returns all ImageFigures used for this AreaHistogram (the eventual

API details

Constructors

Histogram (ImageAnalysisToolbox toolbox)
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)
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 <code>getPlot()</code>
Return
PlotXY
Returns the PlotXY shown in this Histogram.
JPanel <code>getPanel()</code>
Return
JPanel
Returns the panel of this Histogram.
int <code>getUsedArea()</code>
Return
int
Returns the used type of area for this Histogram.

```
int getUsedRegion()
```

Return

```
int
```

Returns the used type of region for this Histogram.

```
int getNumberOfEdges()
```

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()
```

Return

```
int
```

Returns the used type of cut levels for this Histogram.

```
double[] getCutlevels()
```

Return

```
double[]
```

Returns the used cut levels (min, max) for this Histogram.

```
int getNumberOfBins()
```

Return

```
int
```

Returns the used number of bins for this Histogram.

```
int getBinWidth()
```

Return

```
int
```

Returns the width of the bins for this Histogram.

```
int getNbOfClicks()
```

Return

```
int
```

Returns the number of valid clicks (i.e. in the image) you made.

```
boolean checkInput()
```

eventual appropriate error message is also constructed here.

Return

boolean checkInput()**boolean**

Returns true if the given input is correct; false otherwise.

boolean checkArea()

adapted, when errors occur.

Return**boolean**

Returns true if the input area is correct; false otherwise.

boolean checkParameters()

is adapted, when errors occur.

Return**boolean**

Returns true if the input parameters are correct; false otherwise.

boolean checkCutLevels()

is adapted, when errors occur.

Return**boolean**

Returns true if the input cut levels are correct; false otherwise.

boolean checkNumberOfBins()

message is adapted, when errors occur.

Return**boolean**

Returns true if the input number of bins is correct; false otherwise.


ArrayList getImageFigures()

region of interest).

Return**ArrayList**

Returns all ImageFigures used for this AreaHistogram (the eventual region of interest).

3.134. 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](#)

3.135. HistogramPanel

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

Description

A panel for the HistogramTask.

API Summary

Constructor	
<code>HistogramPanel()</code>	The constructor of a new HistogramPanel.
Methods	
<code>JPanel getImagePanel()</code>	Returns a panel that displays the image for this
<code>JPanel getButtonPanel()</code>	Returns the button panel for this HistogramPanel.
<code>setSiteEventHandler(SiteEventHandler handler)</code>	Sets the site event handler for this HistogramPanel to the
<code>setTask(TaskApi task)</code>	Associates the given task with this HistogramPanel.
<code>setVariableSelection(VariableSelection selection)</code>	Sets the variable selection for this HistogramPanel to the given
<code>Display getDisplay()</code>	Returns the display for this HistogramPanel.
<code>Image getImage()</code>	Returns the image associated with this HistogramPanel.
<code>TaskApi getTask()</code>	Returns the task associated with this
<code>Map getMap()</code>	Returns the map associated with this HistogramPanel.
<code>SiteEventHandler getHandler()</code>	Returns the site event handler associated with this
<code>Double getInputLowCut()</code>	Returns the input low cut level for this
<code>Double getInputHighCut()</code>	Returns the input high cut level for this
<code>getInputNbOfBins()</code>	Returns the input number of bins for this
<code>PlotXY getHistogram()</code>	

Methods	
	Returns the histogram associated with this
<code>setPoint(MouseEvent me)</code>	Sets the position of the given mouse event at the
<code>setPoint(Double point)</code>	Sets the given screen coordinates at the appropriate place
<code>updateFigure()</code>	Updates the figure associated with this
<code>ArrayList getClickedPoints()</code>	Returns the clicked points for this HistogramPanel.
<code>int getNbOfClickedPoints()</code>	Returns the number of times there was clicked before
<code>increaseClicks()</code>	Increases the number of clicks made before on the image
<code>drawFigure()</code>	Draws the figure on the image associated with this
<code>trigger()</code>	Triggers the execution of the task associated
<code>updateHistogram()</code>	Updates the histogram associated with this

API details

Constructor

<code>HistogramPanel()</code>

Methods

<code>JPanel getImagePanel()</code>
HistogramPanel.
Return
<code>JPanel</code>
Returns a panel that displays the image for this HistogramPanel.

<code>JPanel getButtonPanel()</code>
Return
<code>JPanel</code>
Returns the button panel for this HistogramPanel.

<code>setSiteEventHandler(SiteEventHandler handler)</code>
given site event handler.
Argument

```
setSiteEventHandler(SiteEventHandler handler)
```

```
  SiteEventHandler handler [INPUT, MANDATORY]
```

```
setTask(TaskApi task)
```

Argument

```
  TaskApi task [INPUT, MANDATORY]
```

```
setVariableSelection(VariableSelection selection)
```

variable selection.

Argument

```
  VariableSelection selection [INPUT, MANDATORY]
```

```
Display getDisplay()
```

Return

[Display](#)

Returns the display for this HistogramPanel.

```
Image getImage()
```

Return

[Image](#)

Returns the image associated with this HistogramPanel.

```
TaskApi getTask()
```

HistogramPanel.

Return

[TaskApi](#)

Returns the task associated with this HistogramPanel.

```
Map getMap()
```

Return

[Map](#)

Returns the map associated with this HistogramPanel.

```
SiteEventHandler getHandler()
```

HistogramPanel.

Return

[SiteEventHandler](#)

Returns the site event handler associated with this HistogramPanel.

```
Double getInputLowCut()
```

HistogramPanel.

Double <code>getInputLowCut()</code>
Return Double Returns the input low cut level for this HistogramPanel.
Double <code>getInputHighCut()</code>
HistogramPanel. Return Double Returns the input high cut level for this HistogramPanel.
<code>getInputNbOfBins()</code>
HistogramPanel.
PlotXY <code>getHistogram()</code>
HistogramPanel. Return PlotXY Returns the histogram associated with this HistogramPanel.
<code>setPoint(MouseEvent me)</code>
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 me [INPUT, MANDATORY]
<code>setPoint(Double point)</code>
(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]
<code>updateFigure()</code>
HistogramPanel.
ArrayList <code>getClickedPoints()</code>
Return ArrayList Returns the clicked points for this HistogramPanel.
int <code>getNbOfClickedPoints()</code>
on the image for this HistogramPanel. Return

int getNbOfClickedPoints()

int

Returns the number of times there was clicked before on the image for this HistogramPanel.
--

increaseClicks()

for this HistogramPanel.

drawFigure()

HistogramPanel.


trigger()

with this HistogramPanel.

updateHistogram()

HistogramPanel.

3.136. HistogramTask

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

Description

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.

3.137. IaProcessImpl

Full Name:	herschel.ia.dataflow.IaProcessImpl
Type:	Java Class - 
Import:	from herschel.ia.dataflow import IaProcessImpl

Description

Implementation of the interface {@link IaProcess}

It has additional helper methods to make a developer's life easier. It is the base class for building event-based processes. A developer may inherit from this class in order to use it within the dataflow framework.

Examples

Example 1: how to implement a event-based process

```
<pre>
import herschel.ia.dataflow.*;
import herschel.ia.numeric.Complex1d;
public class ProcessFFT extends IaProcessImpl implements InputListener {
private final ProcessInput _input;
private final ProcessOutput _output;
public ProcessFFT(String name) {
    super(name);
    _input = createInput("input", Complex1d.class);
    _output = createOutput("output", Complex1d.class);
    // It will always listen to events.
    _input.addInputListener( this );
}
public void inputHandler(ProcessInputEvent ev) {
    Product product = (Complex1d) ev.getData();
    // a doStuff method should be defined and would do the
    processing.
    Product new_product = doStuff(product);
    _output.dataReady( new_product );
}
}
</pre>
```

Example 2: how to implement a event-based process with Launchable

```
<pre>
import herschel.ia.dataflow.*;
import herschel.ia.numeric.Complex1d;
public class ProcessFFT extends IaProcessImpl implements InputListener,
    Launchable {
private final ProcessInput _input;
private final ProcessOutput _output;
public ProcessFFT(String name) {
    super(name);
    _input = createInput("input", Complex1d.class);
    _output = createOutput("output", Complex1d.class);
}
public void inputHandler(ProcessInputEvent ev) {
    Product product = (Complex1d) ev.getData();
    // a doStuff method should be defined and would do the
    processing.
    Product new_product = doStuff(product);
    _output.dataReady( new_product );
}
public void start() {
```

Example 2: how to implement a event-based process with Launchable

```
        _input.addListener( this );
    }
    public void stop() {
        _input.removeListener( this );
    }
}
</pre>
```

Example 3: how to use a process.

```
<pre>
from myProcesses import ProcessFFT
c = Complex1d.range(100);
p = ProcessFFT("fft")
p.getConnector("input").pass(c)
p.getConnector("output").pass(c);
print c
</pre>
```

Limitations

no limitation


See also

- [reference](#)

History

- 26-5-2005 jeg: change javadoc format for help support.

3.138. IFFT

Full Name:	herschel.ia.numeric.toolbox.xform.IFFT.entry.urm.xml
Alias:	IFFT
Type:	Unknown (XML-based documentation) - 
Import:	from herschel.ia.numeric.toolbox.xform import IFFT.entry.urm.xml

Description

Gives the inverse of the Fast Fourier Transform.

Example

Example 1: Apply IFF

```
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))
```

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

- [FFT](#)

3.139. 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.

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.

3.140. ImageAddTask

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

Description

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.

3.141. ImageAnalysis

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

Description

A class to deal with ImageAnalysis (in the current version :

ProfilePlotting, AperturePhotometry, AreaHistograma, and ContourPlotting).

API Summary

Constructor
ImageAnalysis (ImageAnalysisToolbox toolbox, String title) Standard constructor for ImageAnalysis. A new window with the
Methods
JFrame getFrame () Method that returns the JFrame of this ImageAnalysis.
ImageAnalysisToolbox getToolbox () Returns the ImageAnalysisToolbox, associated with this
Container getComponent () Returns the Content Pane of the JFrame of this
ArrayList getImageFigures () Returns all ImageFigures used for this ImageAnalysis.

API details

Constructor

ImageAnalysis (ImageAnalysisToolbox toolbox, String title) given title and associated with the given ImageAnalysisToolbox is created.
Arguments ImageAnalysisToolbox toolbox [INPUT, MANDATORY] String title [INPUT, MANDATORY]

Methods

JFrame getFrame ()
Return JFrame Returns the JFrame of this ImageAnalysis.

ImageAnalysisToolbox `getToolbox()`

ImageAnalysis.

Return

ImageAnalysisToolbox

Returns the ImageAnalysisToolbox, associated with this ImageAnalysis.

Container `getComponent()`

ImageAnalysis.

Return

Container

Returns the Content Pane of the JFrame of this ImageAnalysis.


ArrayList `getImageFigures()`

Return

ArrayList

Returns all ImageFigures used for this ImageAnalysis.

3.142. ImageAnalysisToolbox

Full Name:	herschel.ia.gui.image.ImageAnalysisToolbox
Alias:	ImageAnalysisToolbox
Type:	Java Class - 
Import:	from herschel.ia.gui.image 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
<pre>ImageDataset imds : iat = ImageAnalysisToolbox(imds) or iat = ImageAnalysisToolbox() iat.setImage(imds)</pre>

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	
<code>setImage(Image image)</code>	Sets the image you wish to analyze with this ImageAnalysisToolbox,
<code>setImage(Array2dData array2d)</code>	Sets the image you wish to analyse with this ImageAnalysisToolbox,
<code>setImage(Array3dData array3d)</code>	Sets the image you wish to analyze with this ImageAnalysisToolbox,
<code>addImage(Image image)</code>	Adds the given image to the image, analyzed with
<code>addImage(Array2dData array2d)</code>	Adds the given image to the images, analyzed with
<code>addImage(Array3dData array3d)</code>	Adds the given image to the images, analyzed with
<code>closeActiveTab()</code>	Closes the active tab, if there is one.
<code>closeAllTabs()</code>	Closes all tabs, if there are any.
<code>setEnabled(boolean enabled)</code>	Disables/enables all tabs on the tabbed pane and the menus for
<code>boolean isEnabled()</code>	Returns whether the tabbed pane and menus for closing tabs and
<code>JFrame getFrame()</code>	Method that returns the frame of this ImageAnalysisToolbox.
<code>JPanel getPanel()</code>	Method that returns the panel of this ImageAnalysisToolbox.
<code>JTabbedPane getTabbedPane()</code>	Method that returns the tabbed pane of this ImageAnalysisToolbox.
<code>Image getImage()</code>	Returns the displayed image.
<code>Display getDisplay()</code>	Returns the Display of the image you are analyzing with
<code>int getSelectedLayer()</code>	Returns the index of the shown layer, or the index of the selected
<code>int getSelectedTab()</code>	Returns the index of the tab, selected for the shown layer.
<code>JTabbedPane getTabOnPane()</code>	Returns the tab (tabbed pane) on the tabbed pane, that is
<code>boolean isInImage(MouseEvent me)</code>	Checks whether the given MouseEvent has occurred within the image
<code>boolean isInImage(double pixX, double pixY)</code>	Checks whether the point with given pixel coordinates (pixX, pixY)
<code>boolean hasWCS()</code>	

Methods
Returns true if sky coordinates are available for the image,
<code>plotProfile2D()</code> Draws straight line on the image, starting at the point in the you
<code>aperturePhotometry()</code> Performs aperture photometry on the image you are analyzing with
<code>histogram()</code> Makes an area histogram. You can choose the kind of area you want
<code>ArrayList getAllFigures()</code> Returns all ImageFigures used for this ImageAnalysisToolbox.
<code>addFigures(ArrayList figures)</code> Adds the ImageFigures belonging to the selected tab up front to
<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
<code>createSpaceInTab()</code> Creates a free space at index 0 in the ArrayList of CanvasFigures,

API details

Constructors

ImageAnalysisToolbox()
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)
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
<code>Image image</code> [INPUT, MANDATORY]
ImageAnalysisToolbox(Array2dData array2d)
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 wich to analyse, is displayed.
Argument
<code>Array2dData array2d</code> [INPUT, MANDATORY]
ImageAnalysisToolbox(Array3dData array3d)
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.

<code>ImageAnalysisToolbox(Array3dData array3d)</code>
--

Argument

<code>Array3dData array3d</code> [INPUT, MANDATORY]

Methods

<code>setLayer()</code>

<code>addLayer()</code>

<code>exit()</code>

ImageAnalysisToolbox).

<code>setImage(Image image)</code>

to the given image.

Argument

<code>Image image</code> [INPUT, MANDATORY]

<code>setImage(Array2dData array2d)</code>
--

to the given ImageDataset.

Argument

<code>Array2dData array2d</code> [INPUT, MANDATORY]

<code>setImage(Array3dData array3d)</code>
--

to the given ImageDataset.

Argument

<code>Array3dData array3d</code> [INPUT, MANDATORY]

<code>addImage(Image image)</code>

this ImageAnalysisToolbox.

Argument

<code>Image image</code> [INPUT, MANDATORY]

<code>addImage(Array2dData array2d)</code>
--

this ImageAnalysisToolbox.

Argument

<code>Array2dData array2d</code> [INPUT, MANDATORY]

<code>addImage(Array3dData array3d)</code>
--

this ImageAnalysisToolbox.

Argument

<code>Array3dData array3d</code> [INPUT, MANDATORY]

<code>closeActiveTab()</code>

<code>closeAllTabs()</code>

setEnabled(boolean enabled)

closing tabs and performing image analysis, of this ImageAnalysisToolbox.

Argument

boolean **enabled** [INPUT, MANDATORY]

boolean isEnabled()

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()**Return**

JFrame

The frame of this ImageAnalysisToolbox.

JPanel getPanel()**Return**

JPanel

The panel of this ImageAnalysisToolbox.

JTabbedPane getTabbedPane()**Return**

JTabbedPane

The tabbed pane of this ImageAnalysisToolbox.

Image getImage()**Return**

Image

Returns the displayed image.

Display getDisplay()

this ImageAnalysisToolbox.

Return

Display

Returns the display of the image you are analyzing with this ImageAnalysisToolbox.

int getSelectedLayer()

tab on the tabbed pane.

```
int getSelectedLayer()
```

Return**int**

Returns the index of the shown layer, or the index of the selected tab on the tabbed pane.

```
int getSelectedTab()
```

Return**int**

The index of the tab, selected for the shown layer.

```
JTabbedPane getTabOnPane()
```

selected.

Return**JTabbedPane**

Returns the tab (tabbed pane) on the tabbed pane, that is selected.

```
boolean isInImage(MouseEvent me)
```

we are analyzing with this ImageAnalysisToolbox.

Argument

MouseEvent me [INPUT, MANDATORY]

Return**boolean**

Returns true if the given MouseEvent has occurred within the image that is being analyzed with this ImageAnalysisToolbox; false otherwise.

```
boolean isInImage(double pixX, double pixY)
```

lies in the image we are analyzing with this ImageAnalysisToolbox.

Arguments

double pixX [INPUT, MANDATORY]

double pixY [INPUT, MANDATORY]

Return**boolean**

Returns true if the point with given pixel coordinates (pixX, pixY) lies in the image we are analyzing with this ImageAnalysisToolbox; false otherwise.

```
boolean hasWCS()
```

analyzed with this ImageAnalysisToolbox; false otherwise.

Return**boolean**

Returns true if sky coordinates are available for the image, analyzed with this ImageAnalysisToolbox; false otherwise.

plotProfile2D()

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.

aperturePhotometry()

this ImageAnalysisToolbox, either with an annular sky aperture or a rectangular sky aperture, and with a chosen sky estimation algorithm.

histogram()

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()**Return**

ArrayList

Returns all ImageFigures used for this ImageAnalysisToolbox.

addFigures(ArrayList figures)

the list of ImageFigures, used for this ImageAnalysisToolbox.

Argument

ArrayList figures [INPUT, MANDATORY]

updateImage()**removeAllAnnotations()****createSpaceOnTabbedPane()**

ArrayLists of ImageFigures, that belong to the image analysis on the shown layer.


createSpaceInTab()

belonging to a new image analysis on the shown layer.

See also

- [???](#)
- [???](#)
- [???](#)
- [???](#)
- [???](#)
- [???](#)

3.143. ImageArithmeticsTask

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

Description

An abstract 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.

3.144. 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>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>Font getLabelFont()</code> Returns the font for the label
<code>Color getLabelFontColor()</code> Returns the color of the label
<code>setLabelFontColor(Color labelFontcolor)</code>

Methods
<p>Sets the color for the label</p>
<p><code>showColorTable(boolean showColorTable)</code></p> <p>Shows or hides the color table for this axis.</p>
<p><code>setWorldCoordinates(boolean wcs)</code></p> <p>Show or hide the world coordinates</p>
<p><code>setTickLabelFont(int newSize)</code></p> <p>Sets the size of the font for the ticks.</p>
<p><code>setTickLabelFont(Font tickLabelFont)</code></p> <p>Sets the font of the ticks</p>
<p><code>Font getTickLabelFont()</code></p> <p>Returns the font of the ticks.</p>
<p><code>Color getTickLabelFontColor()</code></p> <p>Returns the color of the font of the ticks</p>
<p><code>setTickLabelFontColor(Color color)</code></p> <p>Sets the color of the ticks font</p>
<p><code>setInnerTickLength(int length)</code></p> <p>Sets the length of the inner ticks</p>
<p><code>int getInnerTickLength()</code></p> <p>Returns the inner tick length</p>
<p><code>setOuterTickLength(int length)</code></p> <p>Sets the length of the outer ticks</p>
<p><code>int getOuterTickLength()</code></p> <p>Returns the outer tick length</p>
<p><code>setMainTicks(int ticks)</code></p> <p>Sets the number of main ticks</p>
<p><code>int getMainTicks()</code></p> <p>Returns the number of main ticks.</p>
<p><code>setMinorTicks(int ticks)</code></p> <p>Sets the number of minor ticks</p>
<p><code>int getMinorTicks()</code></p> <p>Returns the number of minor ticks.</p>
<p><code>setTickDistance(int pixels)</code></p> <p>Sets the tick distance</p>
<p><code>int getTickDistance()</code></p> <p>Returns the number of pixels between two ticks on the axis.</p>
<p><code>showGridLines(boolean gl)</code></p> <p>Enables or disables the grid</p>
<p><code>boolean getShowGridLines()</code></p> <p>Returns the status of the grid</p>
<p><code>setGridColor(Color gridColor)</code></p> <p>Sets the color of the grid</p>

Methods
Color <code>getGridColor()</code> Returns the color of the grid

API details

Methods

<code>disable()</code>
Disables the axis

<code>enable()</code>
Enables the axis

<code>setLabel(String newLabel)</code>
The label of the axis is set. If no label is needed, an empty string should be given as label
Argument
<code>String newLabel</code> [INPUT, MANDATORY]

<code>String getLabel()</code>
Returns the label of the axis.
Return
<code>String</code>
The label of the axis.

<code>setBackground(Color newBackground)</code>
Sets the Background.
Argument
<code>Color newBackground</code> [INPUT, MANDATORY]

<code>Position getOrientation()</code>
Returns the orientation of the axis
Return
<code>Position</code>
The orientation (as a Position)

<code>setOrientation(Position orientation)</code>
Changes the orientation of the axis.
Argument
<code>Position orientation</code> [INPUT, MANDATORY]

<code>setAxisStroke(int stroke)</code>
Sets the stroke (line width) of the axis (in pixels)
Argument
<code>int stroke</code> [INPUT, MANDATORY]

```
int getAxisStroke()
```

Returns the stroke (line width) of the axis.

Return

int

The stroke of the axis.

```
setAxisColor(Color color)
```

Sets the color of the axis.

Argument

Color color [INPUT, MANDATORY]

```
Color getAxisColor()
```

Returns the color of the axis

Return

Color

Color The color of the axis.

```
setLabelFont(Font labelFont)
```

Sets the font for the label.

Argument

Font labelFont [INPUT, MANDATORY]

```
Font getLabelFont()
```

Returns the font for the label of the axis.

Return

Font

The font for the label.

```
Color getLabelFontColor()
```

Returns the color of the label.

Return

Color

the color of the label.

```
setLabelFontColor(Color labelFontcolor)
```

Sets the color for the label

Argument

Color labelFontcolor [INPUT, MANDATORY]

```
showColorTable(boolean showColorTable)
```

Shows (true) or hides (false) the color table for this axis.

Argument

showColorTable(boolean showColorTable)
boolean showColorTable [INPUT, MANDATORY]
setWorldCoordinates(boolean wcs)
Shows the world coordinates (True) or the pixel coordinates (False)
Argument
boolean wcs [INPUT, MANDATORY]
setTickLabelFont(int newSize)
Sets the size of the font for the ticks.
Argument
int newSize [INPUT, MANDATORY]
setTickLabelFont(Font tickLabelFont)
Sets the font of the ticks
Argument
Font tickLabelFont [INPUT, MANDATORY]
Font getTickLabelFont()
Returns the font of the ticks.
Return
Font
The font of the ticks,
Color getTickLabelFontColor()
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
Argument
Color color [INPUT, MANDATORY]
setInnerTickLength(int length)
Sets the distance that the ticks enter in the image
Argument
int length [INPUT, MANDATORY]
int getInnerTickLength()
Returns the distance that the ticks enter in the image.
Return
int

```
int getInnerTickLength()
```

The distance that the ticks enter in the image.

```
setOuterTickLength(int length)
```

Sets the distance that the ticks go out of the image

Argument

```
int length [INPUT, MANDATORY]
```

```
int getOuterTickLength()
```

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. 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 (labeled) ticks.

Return

```
int
```

The number of main ticks.

```
setMinorTicks(int 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 (non-labeled) ticks, between two main ticks.

Return

```
int
```

The number of minor ticks between two main ticks.

```
setTickDistance(int pixels)
```

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. 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)

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 true if the grid is shown.

Return

boolean

True if the grid is shown

setColor(Color gridColor)

Sets the color of the grid for this axis.

Argument

Color gridColor [INPUT, MANDATORY]

Color getGridColor()


Returns the color of the grid

Return

Color

The color of the grid.

3.145. ImageCeilTask

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

Description

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.

3.146. 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.

API Summary

Constructors
<code>ImageContour()</code> This constructor for ImageContour creates a new ImageContour.
<code>ImageContour(Image image)</code> This constructor for ImageContour creates a new ImageContour,
<code>ImageContour(Image image, int nbOfContourValues)</code> This constructor for ImageContour creates a new ImageContour,
<code>ImageContour(Image image, int nbOfContourLevels, double[] extremeContourValues, String distribution)</code> This constructor for ImageContour creates a new ImageContour,
Methods
<code>addContourLevel(ContourLevel contourLevel, double contourValue)</code> Adds the given ContourLevel for the given contour value to this
<code>addContourLevel(ContourLevel contourLevel, String key)</code> Adds the given ContourLevel to this ImageContour with the given
<code>setWcs(Wcs wcs)</code> Sets the Wcs for this ImageContour.
<code>Wcs getWcs()</code> Returns the Wcs for this ImageContour.
<code>boolean hasWcs()</code> Checks whether a Wcs is attached to this ImageContour.
<code>boolean hasValidWcs()</code> Checks whether a valid Wcs is attached to this ImageContour.

API details

Constructors

<code>ImageContour()</code>
<code>ImageContour(Image image)</code> associated with the given image.

ImageContour(Image image)
Argument Image image [INPUT, MANDATORY]
ImageContour(Image image, int nbOfContourValues)
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)
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)
ImageContour. Arguments ContourLevel contourLevel [INPUT, MANDATORY] double contourValue [INPUT, MANDATORY]
addContourLevel(ContourLevel contourLevel, String key)
key. Arguments ContourLevel contourLevel [INPUT, MANDATORY] String key [INPUT, MANDATORY]
setWcs(Wcs wcs)
Argument Wcs wcs [INPUT, MANDATORY]
Wcs getWcs()
Return Wcs Returns the Wcs for this ImageContour.
boolean hasWcs()
Return

boolean hasWcs ()

boolean

Returns true if a Wcs is attached to this ImageContour, false otherwise.


boolean hasValidWcs ()

Return

boolean

Returns true if a valid Wcs is attached to this ImageContour; false otherwise.

3.147. ImageDivideTask

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

Description

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.

3.148. 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.

3.149. 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.

3.150. 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 output image, being the exp scaled image.

3.151. ImageFloorTask

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

Description

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.

3.152. ImageHistogramExplorer

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

Description

An explorer for ImageHistogramProducts.

API Summary

Constructors
<code>ImageHistogramExplorer()</code> The constructor of a new ImageHistogramExplorer.
<code>ImageHistogramExplorer(Object object)</code> The constructor of a new ImageHistogramExplorer associated
Methods
<code>String getName()</code> Returns the name for this ImageHistogramExplorer.
<code>String getDescription()</code> Returns the description for this ImageHistogramExplorer.
<code>boolean canHandle(Class className)</code> Checks whether this ImageHistogramExplorer can handle objects of the
<code>setObject(Object object)</code> Sets the object for this ImageHistogramExplorer to the given object.
<code>ImageHistogramProduct getObject()</code> Returns the object for this ImageHistogramExplorer.
<code>JComponent getComponent()</code> Returns the component that is responsible for displaying the
<code>JTable getParameterTable()</code> Returns the parameter table for this
<code>JComponent getHistogram()</code> Returns the histogram for this ImageHistogramExplorer.
<code>addDataObjectListener(DataObjectListener listener)</code> Add the given listener to this ImageHistogramExplorer to
<code>removeDataObjectListener(DataObjectListener listener)</code> Removes the given listener for this ImageHistogramExplorer, so

API details

Constructors

<code>ImageHistogramExplorer()</code>
--

```
ImageHistogramExplorer(Object object)
```

with the given object.

Argument

`Object object` [INPUT, MANDATORY]

Methods

```
String getName()
```

Return

`String`

Returns the name for this ImageHistogramExplorer.

```
String getDescription()
```

Return

`String`

Returns the description for this ImageHistogramExplorer.

```
boolean canHandle(Class className)
```

given class.

Argument

`Class className` [INPUT, MANDATORY]

Return

`boolean`

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

```
setObject(Object object)
```

Argument

`Object object` [INPUT, MANDATORY]

```
ImageHistogramProduct getObject()
```

Return

`ImageHistogramProduct`

Returns the object for this ImageHistogramExplorer.

```
JComponent getComponent()
```

data object for this ImageHistogramExplorer.

Return

`JComponent`

Returns the component that is responsible for displaying the data object for this ImageHistogramExplorer.

JTable `getParameterTable()`

ImageHistogramExplorer.

Return

JTable

Returns the parameter table for this ImageHistogramExplorer.

JComponent `getHistogram()`**Return**

JComponent

Returns the histogram component for this ImageHistogramExplorer.

addDataObjectListener(DataObjectListener listener)

receive data object events from it.

Argument

DataObjectListener listener [INPUT, MANDATORY]


removeDataObjectListener(DataObjectListener listener)

that it no longer receives data object events by this explorer.

Argument

DataObjectListener listener [INPUT, MANDATORY]

3.153. 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	
<code>ImageHistogramProduct()</code>	The constructor of a new ImageHistogramProduct.
Methods	
<code>setCutLevels(double lowCut, double highCut)</code>	Sets the cut levels for this ImageHistogramProduct
<code>setNbOfBins(int bins)</code>	Sets the number of bins for this
<code>setHistogram(DoubleIcd values, DoubleIcd frequencies)</code>	Sets the histogram for this ImageHistogramProduct to a
<code>setUnit(Unit unit)</code>	Sets the unit for this ImageHistogramProduct to the given unit.
<code>double getLowCut()</code>	Returns the minimum cut level for this
<code>double getHighCut()</code>	Returns the maximum cut level for this
<code>int getNbOfBins()</code>	Returns the number of bins for this
<code>TableDataset getHistogram()</code>	Returns the histogram for this ImageHistogramProduct.
<code>DoubleIcd getValues()</code>	Returns the values for the histogram for this
<code>DoubleIcd getFrequencies()</code>	Returns the frequencies for the histogram for this
<code>String getUnit()</code>	Returns the unit for this ImageHistogramProduct.

API details

Constructor

<code>ImageHistogramProduct()</code>

Methods

setCutLevels(double lowCut, double highCut)

to the given cut levels.

Arguments

double **lowCut** [INPUT, MANDATORY]

double **highCut** [INPUT, MANDATORY]

setNbOfBins(int bins)

ImageHistogramProduct to the given number of bins.

Argument

int **bins** [INPUT, MANDATORY]

setHistogram(DoubleId values, DoubleId frequencies)

histogram with the given values and frequencies.

Arguments

DoubleId values [INPUT, MANDATORY]

DoubleId frequencies [INPUT, MANDATORY]

setUnit(Unit unit)

Argument

Unit unit [INPUT, MANDATORY]

double getLowCut()

ImageHistogramProduct.

Return

double

Returns the minimum cut level for this ImageHistogramProduct.

double getHighCut()

ImageHistogramProduct.

Return

double

Returns the maximum cut level for this ImageHistogramProduct.

int getNbOfBins()

ImageHistogramProduct.

Return

int

Returns the number of bins for this ImageHistogramProduct.

TableDataset getHistogram()

Return

TableDataset `getHistogram()`**TableDataset**

Returns the histogram for this ImageHistogramProduct.

Double1d `getValues()`

ImageHistogramProduct.

Return**Double1d**

Returns the values for the histogram for this

Double1d `getFrequencies()`

ImageHistogramProduct.


Return**Double1d**

Returns the frequencies for the histogram for this ImageHistogramProduct.

String `getUnit()`**Return****String**

Returns the unit for this ImageHistogramProduct.

3.154. 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 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.

3.155. 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.

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.

3.156. 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.

API Summary


Properties
<code>Image.class image</code> [INPUT, MANDATORY, default=No default value]
<code>Double.class n</code> [INPUT, MANDATORY, default=No default value]
<code>Image.class logN</code> [OUTPUT, MANDATORY, default=No default value]

API details

Properties

<code>Image.class image</code> [INPUT, MANDATORY, default=No default value]
The input image.
<code>Double.class n</code> [INPUT, MANDATORY, default=No default value]
The input n.
<code>Image.class logN</code> [OUTPUT, MANDATORY, default=No default value]
The output logN scaled image.

3.157. 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.

3.158. ImageModuloTask

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

Description

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.

3.159. ImageMultiplyTask

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

Description

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.

3.160. ImagePowerTask

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

Description

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.

3.161. ImageRoundTask

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

Description

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.

3.162. imageSaverTask

Full Name:	herschel.ia.toolbox.image.ImageSaverTask
Alias:	imageSaverTask
Type:	Java Task - 
Import:	from herschel.ia.toolbox.image import ImageSaverTask
Category:	task/image

Description

The ImageSaver task for Images.

ImageSaverTask is a task which translates 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

```
translateTask(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

See also

- [???](#)

3.163. ImageSqrtTask

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

Description

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.

3.164. ImageSquareTask

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

Description

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.

3.165. ImageSubtractTask

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

Description

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.

3.166. importCubeTask

Full Name:	herschel.ia.toolbox.cube.ImportCubeTask
Alias:	importCubeTask
Type:	Java Task - 
Import:	from herschel.ia.toolbox.cube import ImportCubeTask
Category:	task/image

Description

The ImportCube task for Cubes.

ImportCubeTask is a task which imports a Cube from a fits file

Example

Example 1: Import a fits file

```
importCubeTask(cube = im, filename = "myFile.fits")
```

API Summary

Jython Syntax

```
importCubeTask(cube, filename)
```

Properties

Cube **cube** [IO, MANDATORY, default=No default value]

string **filename** [INPUT, MANDATORY, default=no default value]

API details

Properties

Cube **cube** [IO, MANDATORY, default=No default value]

The input Cube to load


string **filename** [INPUT, MANDATORY, default=no default value]

The file to import

See also

- ???

3.167. importImageTask

Full Name:	herschel.ia.toolbox.image.ImportImageTask
Alias:	importImageTask
Type:	Java Task - 
Import:	from herschel.ia.toolbox.image import ImportImageTask
Category:	task/image

Description

The ImportImage task for Images.

ImportImageTask is a task which imports an Image from a file (bmp, fpx, gif, jpg, png, pnm, tiff or fits format). It is also possible to import a FITS file.

Example

Example 1: Import a jpg file

```
importFileTask(image = im, filename = "myFile.jpg")
```

API Summary

Jython Syntax

```
importFileTask(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

3.168. 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.

3.169. 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)

3.170. 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)

3.171. 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)

3.172. 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)

3.173. 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)

3.174. 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

```
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
```

API Summary

Jython Syntax

```
<r>=SomeOpenIntervalIntegrator(<a>,<b>).integrate(<f>)
<r>=SomeShutIntervalIntegrator().integrate(<f>)
```

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.

3.175. 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.

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.

3.176. 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.

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.

3.177. 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
<pre>x=Float2d([[1,2],[3,4]]) print INVERSE(x) # [[-2.0,1.0], [1.5,-0.5]]</pre>

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

3.178. 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](#)

3.179. 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](#)

3.180. 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](#)

3.181. 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
<pre>x=Float1d([1,3,2,3,4]) print KURTOSIS(x) # -1.74840236686</pre>

API Summary

Jython Syntax
<code><y>=KURTOSIS(<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](#)
- [SKEWNESS](#)
- [STDDEV](#)

- VARIANCE

3.182. LevenbergMarquardtFitter

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

Description

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.

Example


Example 1: The fit/demo directory contains worked

```
<a href="../../../ia/numeric/toolbox/fit/demo/jython.html">examples</a>  
on almost all aspects of of the package.
```

Limitations

1. LevenbergMarquardtFitter 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.

3.183. LinearInterpolator

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

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 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](#)

3.184. 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()`

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 =

```
storage.save(product) # the ref is a ProductRef object
```

List `getRefs()`

```
listcontext.refs.add(ref)
```


Getting the first product from the ListContext `ref_first =`

```
lcontext.refs.get(0)
```

See also

- [Product Access Layer](#)

3.185. 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 the user.

Example

Example 1: Saving a product into a local store

```
p = Product()
store = PoolManager.getPool("mib")
simpleFitsWriter(product = p, store = store)
```

API Summary

Jython Syntax

```
localStoreWriter(product, store)
```

Properties

Product [product](#) [INPUT, MANDATORY, default=null]

LocalStore [store](#) [INPUT, MANDATORY, default=null]

API details

Properties

Product [product](#) [INPUT, MANDATORY, default=null]

Product to be saved.


LocalStore [store](#) [INPUT, MANDATORY, default=null]

LocalStore where to save the product.

History

- 2008-11-11 - JSS: first release

3.186. 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
<pre>x=Double1d([1,10,100]) print LOG10(x) # [0.0,0.9999999999999999,1.999999999999998] print ROUND(LOG10(x)) # [0.0,1.0,2.0]</pre>

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](#)

3.187. 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
<pre>x=Double1d([0,1]) print LOG(EXP(x)) # [0.0,1.0]</pre>

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](#)

3.188. 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](#)

3.189. 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)

3.190. 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)

3.191. 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)

3.192. 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)

3.193. 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)

3.194. ManualContourTask

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

Description

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]
ArrayList values [INPUT, MANDATORY, 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.
ArrayList values [INPUT, MANDATORY, default=No default value]
The contour values.
ImageContour contours [OUTPUT, MANDATORY, default=No default value]
The ImageContour.

3.195. 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
<p><code>Map</code> getRefs()</p> <p>get the 'map' of ProductRefs stored. From this 'map', you can put</p>

API details

Method

<p><code>Map</code> getRefs()</p> <p>products into the MapContext, or retrieve products by key.</p> <p>Return</p> <p><code>Map</code></p> <p>A map of product refs.</p> <p>Examples</p> <p>Putting a product into the the MapContext</p>
--

Map getRefs()

```
ref = 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

- [Product Access Layer](#)

3.196. MapMakingTask

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

Description

An abstract Task for mapmaking.

API Summary


Properties
<code>ArrayList scans</code> [INPUT, MANDATORY, default=No default value]
<code>SimpleImage map</code> [OUTPUT, MANDATORY, default=No default value]

API details

Properties

<code>ArrayList scans</code> [INPUT, MANDATORY, default=No default value]
The input scans.
<code>SimpleImage map</code> [OUTPUT, MANDATORY, default=No default value]
The reconstructed sky map.

3.197. MATMUL

Full Name:	herschel.ia.numeric.toolbox.matrix.doc.MATMUL.entry.urm.xml
Alias:	MATMUL
Type:	Unknown (XML-based documentation) - 
Import:	from herschel.ia.numeric.toolbox.matrix.doc import MATMUL.entry.urm.xml

Description

Returns matrix multiplication.

This wrapper is deprecated. Please, use the `MatrixMultiply` Java class.


Limitations

This is a Jython wrapper function for the numeric `MatrixMultiply(b)(A)`.

See also

- [MatrixMultiply](#)

3.198. 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

```
[1 2 3]      [1 2]
A=[2,3,4] and B=[2 3]
              [3 4]
X=[1 2]      and Y=[1 2 3]

A.apply(MatrixMultiply(B)) = [14 20]
x.apply(MatrixMultiply(A)) = [20 29]
A.matrixMultiply(y)       = [5 8 11]
```

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


[Array1dData](#) or [Array2dData](#) **a** [INPUT, MANDATORY, default=no default value]

Input must be a rank one or rank two array of any type, but String or Complex.

**Array1dData or Array2dData 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.

3.199. 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
<pre># 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))</pre>

API Summary

Jython Syntax
<code><x>=MatrixSolve()(<A>)</code>
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.

3.200. 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

```
x=Int2d( [ [1,2], [-1,3] ] )
print MAX(x) # 3
print MAX(x, 0) # [1,3]
print MAX(x, 1) # [2,3]
```

API Summary

Jython Syntax

```
<y>=MAX(<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

- [MIN](#)

3.201. 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](#)

3.202. MEDIANDEV

Full Name:	herschel.ia.numeric.toolbox.basic.MedianAbsoluteDeviation
Alias:	MEDIANDEV
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 MedianStandardDeviation to an Int1d array

```
data = Int1d.range(9)
median = MEDIAN(data)
dev = data.apply( MedianStandardDeviation(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]
```

3.203. MedianFilteringTask

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

Description

A Task to smooth an image using the median filter. The median

filter computes the median of all pixel values in the 3*3 filter window. In order to filter pixel located near the edges of the image, edge pixel values are replicated to give sufficient data.

API Summary


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

API details

Properties

Image image [INPUT, MANDATORY, default=No default value]
The image before filtering.
Image filtered [OUTPUT, MANDATORY, default=No default value]
The image after filtering.

3.204. 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.

Example

Example 1: Apply MEDIAN on a Float1d
<pre>x=Float1d([1,3,2,3,4]) print MEDIAN(x) # 3.0</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 a double

See also

- [MEAN](#)
- [STDDEV](#)
- [VARIANCE](#)

3.205. MedianNoiseTask

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

Description

A Task to make a median noise image of an image, using median filtering.

API Summary


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

API details

Properties

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

3.206. 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

- [Querying](#)

3.207. 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](#)

3.208. 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 in a naive way (i.e. by simple co-adding).

API Summary

Properties
<code>ArrayList images</code> [INPUT, MANDATORY, default=No default value]
<code>SimpleImage mosaic</code> [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

<code>ArrayList images</code> [INPUT, MANDATORY, default=No default value]
The input images.
<code>SimpleImage mosaic</code> [OUTPUT, MANDATORY, default=No default value]
The mosaic.

3.209. MultiplySpectrum

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

Description

Task for multiply the flux included in a spectrum container with a scalar or for multiplying two spectrum containers on a spectrum-by-spectrum basis.

For the scalar mode, 'ds' and 'param' need to specified - for the pair-wise mode, 'ds1' and 'ds2' need to be set. In the pairwise mode, the task iterates synchronously 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.

In either mode, you can specify the segments you would like to process. See the 'segments' (for the scalar mode) or the 'segments1', 'segments2' parameters for the pair-wise mode.

For scalar mode, you have advanced selection functionality in place using the selection models as already discussed in the SpectrumTask.

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).

For further details see the (abstract) SpectrumTask in the same package.

Example

Example 1: from Jide:

```
from herschel.ia.toolbox.spectrum import MultiplySpectrum
multiply = MultiplySpectrum()
multipliedByFactor = multiply(ds=spectra, param=2.1)
multipliedPairWise = multiply(ds1=spectra1, ds2=spectra2)
```

API Summary

Properties
SpectrumContainer ds [INPUT, OPTIONAL, default=no default value.]
Double param [INPUT, OPTIONAL, default=no default value.]
SelectionModel selection [INPUT, OPTIONAL, default=no default value.]
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.]

Properties
SpectrumContainer ds1 [INPUT, OPTIONAL, default=no default value.]
SpectrumContainer ds2 [INPUT, OPTIONAL, default=no default value.]
SegmentSelection segments1 [INPUT, OPTIONAL, default=no default value.]
SegmentSelection 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 many-to-one operation (eg avg) or as scalar operation.
Double param [INPUT, OPTIONAL, default=no default value.]
The scalar parameter to be considered when the task is used as scalar operation.
SelectionModel selection [INPUT, OPTIONAL, default=no default value.]
Specify an instance of any kind of SelectionModel here.
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.
SegmentSelection segments1 [INPUT, OPTIONAL, default=no default value.]
SegmentSelection to be associated with 'ds1'.

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

SegmentSelection 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

3.210. NaiveMapMakingTask

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

Description

A Task for naive mapmaking.

API Summary

Properties
<code>ArrayList scans</code> [INPUT, MANDATORY, default=No default value]
<code>SimpleImage map</code> [OUTPUT, MANDATORY, default=No default value]

Miscellaneous


The naive map can be used as a priori model for mapmaking as an inverse problem.

API details

Properties

<code>ArrayList scans</code> [INPUT, MANDATORY, default=No default value]
The input scans.
<code>SimpleImage map</code> [OUTPUT, MANDATORY, default=No default value]
The reconstructed sky map.

3.211. NearestNeighborInterpolator

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

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 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](#)

3.212. NoiseTask

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

API Summary


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

API details

Properties

Image image [INPUT, MANDATORY, default=No default value]
The input image.
Double2d noise [OUTPUT, MANDATORY, default=No default value]
The output noise.

3.213. 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>,[,<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), ...]

3.214. 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](#)

3.215. 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.

3.216. 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=[]}
productContext = MapContext()
obs.level['level0']=productContext #Error
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](#)
- [???](#)


3.217. 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

3.218. PackedMask

Full Name:	herschel.ia.numeric.toolbox.mask.PackedMask
Type:	Java Class - 
Import:	from herschel.ia.numeric.toolbox.mask import PackedMask

Description

This mask handling class compresses the mask to minimise memory usage. This


implementation has no limit on the number of masks that can be created. Note that the mask data is stored internally by this class.

Example

Example 1: Create a packed mask array and set and unset at a specific location.

```
GLITCH = PackedMask ("Glitch", [1024,1024,3])
print GLITCH.isSet ([5,5,2]) # 0
GLITCH.set ([5,5,2])
print GLITCH.isSet ([5,5,2]) # 1
GLITCH.unset ([5,5,2])
print GLITCH.isSet ([5,5,2]) # 0
```

3.219. PacketSequence


Full Name:	herschel.binstruct.PacketSequence
Alias:	PacketSequence
Type:	Java Class - 
Import:	from herschel.binstruct import PacketSequence
Category:	binstruct

Description

a container for telemetry and telecommand source packets

In principle the `PacketSequence` is a general purpose container for telemetry and telecommand packets. However most of the time it will be used to group all the information of one observation or test because those packets have a natural connection.

3.220. 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 beetwen the assignemnt of x and

```
its printing, assignemnt 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.

3.221. 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.

3.222. Polygon

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

Description

A polygon shape.

API Summary

Constructors
Polygon() The construction of a new Polygon without coordinates.
Polygon(double[] coords) The construction of a new Polygon with the given vertices
Polygon(double x, double y) The construction of a new Polygon with a single starting point.
Polygon(int size) The construction of a new Polygon with space for the given
Method
boolean contains(double x, double y) Checks whether the point with the given pixel coordinates is inside this Polygon.

Miscellaneous

This class extends the existing `diva.util.java2d.Polygon2D.Double`, because the `contains()` method was implemented incorrectly there.

API details

Constructors

Polygon()
Polygon(double[] coords) in the format [x0, y0, x1, y1,...].
Argument double[] coords [INPUT, MANDATORY]
Polygon(double x, double y)
Arguments double x [INPUT, MANDATORY] double y [INPUT, MANDATORY]

Polygon(int size)

number of vertices.


Argument`int size [INPUT, MANDATORY]`

Method

boolean contains(double x, double y)**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.

3.223. PolygonHistogramExplorer

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

Description

An explorer for PolygonHistogramProducts.

API Summary

Constructors
<code>PolygonHistogramExplorer()</code> The constructor of a new PolygonHistogramExplorer.
<code>PolygonHistogramExplorer(Object object)</code> The constructor of a new PolygonHistogramExplorer associated
Methods
<code>String getName()</code> Returns the name for this PolygonHistogramExplorer.
<code>String getDescription()</code> Returns the description for this PolygonHistogramExplorer.
<code>boolean canHandle(Class className)</code> Checks whether this PolygonHistogramExplorer can handle objects of the
<code>setObject(Object object)</code> Sets the object for this PolygonHistogramExplorer to the given object.
<code>PolygonHistogramProduct getObject()</code> Returns the object for this PolygonHistogramExplorer.
<code>JTable getParameterTable()</code> Returns the parameter table for this

API details

Constructors


<code>PolygonHistogramExplorer()</code>
<code>PolygonHistogramExplorer(Object object)</code> with the given object.
Argument <code>Object object</code> [INPUT, MANDATORY]

Methods

<code>String getName()</code>
Return

String getName()
String Returns the name for this PolygonHistogramExplorer.
String getDescription()
Return String Returns the description for this PolygonHistogramExplorer.
boolean canHandle(Class className)
given class. Argument Class className [INPUT, MANDATORY] Return boolean Returns true if this PolygonHistogramExplorer can handle objects of the given class; false otherwise.
setObject(Object object)
Argument Object object [INPUT, MANDATORY]
PolygonHistogramProduct getObject()
Return PolygonHistogramProduct Returns the object for this PolygonHistogramExplorer.
JTable getParameterTable()
EllipseHistogramExplorer. Return JTable Returns the parameter table for this EllipseHistogramExplorer.

3.224. PolygonHistogramPanel

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

Description

A panel for the PolygonHistogramTask.

API Summary

Constructor
PolygonHistogramPanel() The construction of a new PolygonHistogramPanel.
Methods
drawFigure() Draws the rectangle on the image associated with this
updateFigure() Updates the polygon associated with this
trigger() Triggers the execution of the task associated
updateHistogram() Updates the histogram associated with this

API details


Constructor

PolygonHistogramPanel()

Methods

drawFigure()	RectangleHistogramPanel.
updateFigure()	PolygonHistogramPanel.
trigger()	with this PolygonHistogramPanel.
updateHistogram()	PolygonHistogramPanel.

3.225. 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(DoubleId edgesPixels) Sets the edges for this PolygonHistogramProduct to the given
setEdges(DoubleId edgesPixels, StringId 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()

Methods

setEdges(DoubleId edgesPixels)
pixel coordinates.
Argument
DoubleId edgesPixels [INPUT, MANDATORY]

```
setEdges(DoubleId edgesPixels, StringId edgesSky)
```

of pixel and sky coordinates.

Arguments

`DoubleId edgesPixels` [INPUT, MANDATORY]

`StringId edgesSky` [INPUT, MANDATORY]

```
CompositeDataset getEdges()
```

Return

`CompositeDataset`

Returns the edges for this PolygonHistogramProduct.

```
int getNbOfEdges()
```

PolygonHistogramProduct.

Return

`int`

Returns the number of edges for this PolygonHistogramTask.

```
TableDataset getEdgesPixelCoordinates()
```

pixel coordinates.

Return

`TableDataset`

Returns the edges for this PolygonHistogramProduct in pixel coordinates.

```
Double2d getEdgesPixelCoordinatesDouble2d()
```

Return

`Double2d`

Returns the pixel coordinates of the edges as a Double2d.

```
TableDataset getEdgesSkyCoordinates()
```


sky coordinates.

Return

`TableDataset`

Returns the edges for this Polygon in sky coordinates.

3.226. 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 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]
DoubleId frequencies [OUTPUT, MANDATORY, default=No default value]
DoubleId edgesPixel [INPUT, OPTIONAL, default=No default value]
StringId 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.
DoubleId 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.
--

3.227. 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.

3.228. 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).

Example

Example 1: PoolDataReaderTask
<pre> <pre> #read by urn ... pdr = PoolDataReaderTask() product = pdr(id='urn:poolid:producClass:N') print product product = pdr(id='myTag') print product ... #read by tag ... pdr = PoolDataReaderTask() product = pdr(id='myTag',pool=ProductPool) print product ... #read two products using the same storage ... ps = ProductStorage() pool = PoolManager.getDefaultPool() ps.register(pool) pdr = PoolDataReaderTask() product1 = pdr(id='myTag1',storage=ps) print product1 product2 = pdr(id='myTag2',storage=ps) print product2 ... #read two products using the same default storage ... pdr = PoolDataReaderTask() product1 = pdr(id='myTag1') print product1 ps = pdr.storage product2 = pdr(id='myTag2',storage=ps) print product2 ... </pre> </pre>

API Summary

Jython Syntax
product=PoolDataReaderTask()(urn='urn:poolid')

Jython Syntax
see examples

Properties
<code>String id [INPUT, true, default=null]</code>
<code>storage [INPUT, ProductStorage, default=false]</code>
<code>ProductPool pool [INPUT, false, default=PAL default pool]</code>
<code>String poolid [INPUT, false, default=null]</code>
<code>Boolean useStorage [INPUT, false, default=true]</code>
<code>Product product [OUTPUT, true, default=null]</code>

API details

Properties

<code>String id [INPUT, true, default=null]</code>
Product identifier, either an urn or a tag. A Tag can be used when reading through storage only.

<code>storage [INPUT, ProductStorage, default=false]</code>
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.

<code>ProductPool pool [INPUT, false, default=PAL default pool]</code>
If storage is specified, the pool parameter is not allowed. If a pool is specified, the poolid parameter is not allowed. 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

<code>String poolid [INPUT, false, default=null]</code>
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. 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.


<code>Boolean useStorage [INPUT, false, default=true]</code>
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.

<code>Product product [OUTPUT, true, default=null]</code>
The loaded product.

History

- 22-11-2007 JCS

3.229. 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 (unless the parameter 'useStorage' is false).

Example

Example 1: PoolDataReaderTask

```

<pre>
#write product
...
p = Product()
pdwt = PoolDataWriterTask()
urn = pdwt(product=p)
print urn
...
#write product with a tag
...
p = Product()
pdwt = PoolDataWriterTask()
urn = pdwt(product=p,tag='myTag')
print urn
...
pdrt = PoolDataReaderTask()
product = pdrt(id='myTag',pool=ProductPool)
print product
...
#write two products using the same storage
...
ps = ProductStorage()
pool = PoolManager.getDefaultPool()
ps.register(pool)
pdwt = PoolDataWriterTask()
urn1 = pdwt(product=Product(),tag='myTag1',storage=ps)
print urn1
urn2 = pdrt(product=Product(),tag='myTag2',storage=ps)
print urn2
...
#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
...
</pre>

```

API Summary

Jython Syntax

```
urn=PoolDataWriterTask()(product=Product())
```

See examples.

Properties

Product product [INPUT, true, default=null]

String tag [INPUT, false, default=null]

storage [INPUT, ProductStorage, default=false]

ProductPool pool [INPUT, false, default=PAL default pool]

String poolid [INPUT, false, default=null]

Boolean useStorage [INPUT, false, default=true]

API details

Properties

Product product [INPUT, true, default=null]

The product to be saved.

String tag [INPUT, false, default=null]

Product tag. A Tag can be used when saving through storage only.

storage [INPUT, ProductStorage, default=false]

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, false, 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. 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, false, 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. 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, false, 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

3.230. 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(flimit, 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(flimit, sigma, deglitch, table)
```


See also

- [???](#)

History

- Feb 22, 2008 - first version

3.231. 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](#)

3.232. 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.

3.233. 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

```
x=Int2d( [ [1,2], [-1,3] ] )
print PRODUCT(x)      # -6
print PRODUCT(x, 0)  # [-1, 6]
print PRODUCT(x, 1)  # [ 2,-3]
```

API Summary

Jython Syntax

```
<y>=PRODUCT(<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

- [SUM](#)

3.234. 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
<code>getProduct()</code> Returns the Product class to which this Product reference is pointing to.
<code>getType()</code> Returns the Product class to which this Product reference is pointing to.

API details

Methods


```
getProduct()
```

```
getType()
```

See also

- [Product Access Layer](#)

3.235. 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()
<pre>myStore.register(SimplePool.getInstance()) # the simple.default pool</pre>

Example 2: loading a Product from this store
<pre>ref=myStore.load("urn:simple.default:herschel.ia.dataset.Product:123")</pre>

Example 3: Untitled
<pre>saving a Product product=Product(...) ref=myStorage.save(product)</pre>

Example 4: select Products based on a query query=AttribQuery(...)
<pre>results=myStorage.select(query)</pre>

API Summary

Constructors
<code>ProductStorage()</code> Creates an empty ProductStorage with no pools registered.
<code>ProductStorage(ProductPool pool)</code> Creates a ProductStorage with the given pool registered, which
<code>ProductStorage(ProductPool[] pools)</code> Creates a ProductStorage with the given pools registered.
<code>ProductStorage(String poolName)</code> Creates a ProductStorage with the the pools corresponding to the
<code>ProductStorage(String[] poolNames)</code> Creates a ProductStorage with the the pools corresponding to the
Methods
<code>setSharedMode(boolean sharedMode)</code> Sets whether the last version info is to be updated before each

Methods	
<code>register(ProductPool pool)</code>	Registers a ProductPool to the ProductStorage. The first
<code>register(ProductPool array pools)</code>	Registers ProductPools to the ProductStorage. The first
<code>getPools()</code>	Provides the set of ProductPools registered, in order in which
<code>getWritablePool()</code>	Returns the writable pool, which is the first registered pool.
Set <code>getProductClasses()</code>	Provides all Product class definitions found in this pool.
<code>remove(String urn)</code>	Removes a product of given URN from the storage.
ProductRef <code>load(String id)</code>	Loads a Product from this store.
ProductRef <code>save(Product product)</code>	Saves a Product to this store. If the product is a context that
ProductRef <code>saveHeadOnly(Product product)</code>	Saves a Product to this store. If the product is a context that
Set <code>select(StorageQuery query)</code>	Returns a set of references to products that match the specified
Set <code>select(StorageQuery query, Set previous)</code>	Returns a set of references to products that match the specified

API details

Constructors

<code>ProductStorage()</code>
<code>ProductStorage(ProductPool pool)</code>
happens to be the writable one.
Argument
<code>ProductPool pool</code> [INPUT, MANDATORY]
Example
Creating a storage initialized with a pool.
<pre>storage = ProductStorage(pool) # pool is the writable one</pre>
<code>ProductStorage(ProductPool[] pools)</code>
The first provided pool is the writable one.
Argument
<code>ProductPool[] pools</code> [INPUT, MANDATORY]
Example
Creating a storage initialized with pools.

ProductStorage(ProductPool[] 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)

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)

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)

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)

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.

register(ProductPool pool)**Example**

How to register a product pool

```
storage.register(pool)
```

register(ProductPool array pools)

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()

they were initially registered.

getWritablePool()**Set getProductClasses()****Return**

Set

a collection of Product classes

remove(String urn)**Argument**

String urn [INPUT, MANDATORY, default=The URN to the Product that]

should be removed.

ProductRef load(String id)**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)

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

ProductRef save(Product product)

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)

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)

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)

query.

Arguments

StorageQuery **query** [INPUT, MANDATORY, default=The query applied to]
 this store.
Set previous [INPUT, MANDATORY]


Return**Set**

A set of references to Products selected by the query

See also

- [Product Access Layer](#)

3.236. ProfileExplorer

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

Description

An explorer for Profiles.

API Summary

Constructors
<code>ProfileExplorer()</code> The constructor of a new ProfileExplorer.
<code>ProfileExplorer(Object object)</code> The constructor of a new ProfileExplorer associated
Methods
<code>String getName()</code> Returns the name for this ProfileExplorer.
<code>String getDescription()</code> Returns the description for this ProfileExplorer.
<code>boolean canHandle(Class className)</code> Checks whether this ProfileExplorer can handle objects of the
<code>setObject(Object object)</code> Sets the object for this ProfileExplorer to the given object.
<code>Profile getObject()</code> Returns the object for this ProfileExplorer.
<code>JComponent getComponent()</code> Returns the component that is responsible for displaying the
<code>JPanel getCoordinatePanel()</code> Constructs and returns the coordinate table for this ProfileExplorer.
<code>JPanel getProfilePlotPanel()</code> Returns the profile plot component for this
<code>addDataObjectListener(DataObjectListener listener)</code> Add the given listener to this ProfileExplorer to
<code>removeDataObjectListener(DataObjectListener listener)</code> Removes the given listener for this ProfileExplorer, so

API details

Constructors

<code>ProfileExplorer()</code>

```
ProfileExplorer(Object object)
```

with the given object.

Argument

`Object object` [INPUT, MANDATORY]

Methods

```
String getName()
```

Return

`String`

Returns the name for this ProfileExplorer.

```
String getDescription()
```

Return

`String`

Returns the description for this ProfileExplorer.

```
boolean canHandle(Class className)
```

given class.

Argument

`Class className` [INPUT, MANDATORY]

Return

`boolean`

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

```
setObject(Object object)
```

Argument

`Object object` [INPUT, MANDATORY]

```
Profile getObject()
```

Return

`Profile`

Returns the object for this ProfileExplorer.

```
JComponent getComponent()
```

data object for this ProfileExplorer.

Return

`JComponent`

Returns the component that is responsible for displaying the data object for this ProfileExplorer.

```
JPanel getCoordinatePanel()
```

Return

```
JPanel
```

Returns the coordinate table for this ProfileExplorer.

```
JPanel getProfilePlotPanel()
```

ProfileExplorer.

Return

```
JPanel
```

Returns the profile plot component for this ProfileExplorer.

```
addDataObjectListener(DataObjectListener listener)
```

receive data object events from it.

Argument

```
DataObjectListener listener [INPUT, MANDATORY]
```

```
removeDataObjectListener(DataObjectListener listener)
```

that it no longer receives data object events by this explorer.

Argument

```
DataObjectListener listener [INPUT, MANDATORY]
```

3.237. 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
Profile() The constructor of a new Profile.
Methods
setBegin(double beginX, double beginY) Sets the begin to the given pixel coordinates.
setBegin(double beginX, double beginY, String beginRA, String beginDec) Sets the begin to the given pixel and sky
setEnd(double endX, double endY) Sets the end to the given pixel coordinates.
setEnd(double endX, double endY, String endRA, String endDec) Sets the end to the given pixel and sky coordinates.
setProfile(DoubleId profile) Sets the profile for this Profile to the given
setUnit(Unit unit) Sets the unit for this Profile.
DoubleId getBeginPixelCoordinates() Returns the begin for this Profile in pixel coordinates.
StringId getBeginSkyCoordinates() Returns the begin for this Profile in sky coordinates.
DoubleId getEndPixelCoordinates() Returns the end for this Profile in pixel coordinates.
StringId getEndSkyCoordinates() Returns the end for this Profile in sky coordinates.
DoubleId getProfile() Returns the profile for this Profile.
String getUnit() Returns the unit for this Profile.

API details

Constructor

```
Profile()
```

Methods

```
setBegin(double beginX, double beginY)
```

Arguments

```
double beginX [INPUT, MANDATORY]
```

```
double beginY [INPUT, MANDATORY]
```

```
setBegin(double beginX, double beginY, String beginRA, String beginDec)
```

coordinates.

Arguments

```
double beginX [INPUT, MANDATORY]
```

```
double beginY [INPUT, MANDATORY]
```

```
String beginRA [INPUT, MANDATORY]
```

```
String beginDec [INPUT, MANDATORY]
```

```
setEnd(double endX, double endY)
```

Arguments

```
double endX [INPUT, MANDATORY]
```

```
double endY [INPUT, MANDATORY]
```

```
setEnd(double endX, double endY, String endRA, String endDec)
```

Arguments

```
double endX [INPUT, MANDATORY]
```

```
double endY [INPUT, MANDATORY]
```

```
String endRA [INPUT, MANDATORY]
```

```
String endDec [INPUT, MANDATORY]
```

```
setProfile(DoubleId profile)
```

profile.

Argument

```
DoubleId profile [INPUT, MANDATORY]
```

```
setUnit(Unit unit)
```

Argument

```
Unit unit [INPUT, MANDATORY]
```

```
DoubleId getBeginPixelCoordinates()
```

Return

Double1d `getBeginPixelCoordinates()`

Double1d

Returns the begin for this Profile in pixel coordinates.

String1d `getBeginSkyCoordinates()`

Return

String1d

Returns the begin for this Profile in sky coordinates.

Double1d `getEndPixelCoordinates()`

Return

Double1d

Returns the end for this Profile in pixel coordinates.

String1d `getEndSkyCoordinates()`

Return

String1d

Returns the end for this Profile in sky coordinates.

Double1d `getProfile()`

Return

Double1d

Returns the profile for this Profile.


String `getUnit()`

Return

String

Returns the unit for this Profile.

3.238. ProfilePanel

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

Description

A panel for the ProfileTask.

API Summary

Constructor	
<code>ProfilePanel()</code>	The constructor of a new ProfilePanel.
Methods	
<code>JPanel getImagePanel()</code>	Returns a panel that displays the image for this
<code>JPanel getButtonPanel()</code>	Returns the button panel for this ProfilePanel.
<code>setSiteEventHandler(SiteEventHandler handler)</code>	Sets the site event handler for this ProfilePanel to the
<code>setTask(TaskApi task)</code>	Associates the given task with this ProfilePanel.
<code>setVariableSelection(VariableSelection selection)</code>	Sets the variable selection for this ProfilePanel to the given
<code>Display getDisplay()</code>	Returns the display for this ProfilePanel.
<code>Image getImage()</code>	Returns the image associated with this ProfilePanel.
<code>ImageFigure getLine()</code>	The straight line for this ProfilePanel.
<code>Double getBegin()</code>	Returns the begin of the straight line associated
<code>Double getEnd()</code>	Returns the end of the straight line associated
<code>TaskApi getTask()</code>	Returns the task associated with this
<code>Map getMap()</code>	Returns the map associated with this ProfilePanel.
<code>SiteEventHandler getHandler()</code>	Returns the site event handler associated with this
<code>PlotXY getPlot()</code>	

Methods
Returns the plot associated with this ProfilePanel.
<code>int getNbOfClicks()</code> Returns the number of clicks made on the image
<code>boolean isInImage(MouseEvent me)</code> Checks whether the given mouse event occurred inside the
<code>boolean isInImage(double pixX, double pixY)</code> Checks whether the given pixel coordinates lies

API details

Constructor

<code>ProfilePanel()</code>

Methods

<code>JPanel getImagePanel()</code>
ProfilePanel.
Return <code>JPanel</code> Returns a panel that displays the image for this ProfilePanel.

<code>JPanel getButtonPanel()</code>
Return <code>JPanel</code> Returns the button panel for this ProfilePanel.

<code>setSiteEventHandler(SiteEventHandler handler)</code>
given site event handler.
Argument <code>SiteEventHandler handler</code> [INPUT, MANDATORY]

<code>setTask(TaskApi task)</code>
Argument <code>TaskApi task</code> [INPUT, MANDATORY]

<code>setVariableSelection(VariableSelection selection)</code>
variable selection.
Argument <code>VariableSelection selection</code> [INPUT, MANDATORY]

<code>Display getDisplay()</code>
Return

Display `getDisplay()`

Display

Returns the display for this ProfilePanel.

Image `getImage()`

Return

Image

Returns the image associated with this ProfilePanel.

ImageFigure `getLine()`

Return

ImageFigure

Returns the straight line for this ProfilePanel.

Double `getBegin()`

with this ProfilePanel in pixel coordinates.

Return

Double

Returns the begin of the straight line associated with this ProfilePanel in pixel coordinates.

Double `getEnd()`

with this ProfilePanel in pixel coordinates.

Return

Double

Returns the end of the straight line associated with this ProfilePanel in pixel coordinates.

TaskApi `getTask()`

ProfilePanel.

Return

TaskApi

Returns the task associated with this ProfilePanel.

Map `getMap()`

Return

Map

Returns the map associated with this ProfilePanel.


SiteEventHandler `getHandler()`

ProfilePanel.

Return

SiteEventHandler <code>getHandler()</code>
SiteEventHandler Returns the site event handler associated with this ProfilePanel.
PlotXY <code>getPlot()</code>
Return PlotXY Returns the plot associated with this ProfilePanel.
int <code>getNbOfClicks()</code>
for this ProfilePanel. Return int Returns the number of clicks made on the image for this ProfilePanel.
boolean <code>isInImage(MouseEvent me)</code>
image for this ProfilePanel. Argument MouseEvent me [INPUT, MANDATORY] Return boolean Returns true if the given mouse event occurred inside the image for this ProfilePanel.
boolean <code>isInImage(double pixX, double pixY)</code>
inside the image for this ProfilePanel. 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.

3.239. ProfilePlotting

Full Name:	herschel.ia.gui.image.ProfilePlotting
Type:	Java Class - 
Import:	from herschel.ia.gui.image 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	
<code>ProfilePlotting(ImageAnalysisToolbox toolbox)</code>	The standard constructor for ProfilePlotting. A new window, associated
<code>ProfilePlotting(boolean useAsComponent, ImageAnalysisToolbox toolbox)</code>	Constructor for ProfilePlotting. When the new ProfilePlotting is
Methods	
<code>setBegin(Double begin)</code>	Sets the begin of the straight line associated with this
<code>setEnd(Double end)</code>	Sets the end of the straight line, associated with this
<code>PlotXY getPlot()</code>	Returns the PlotXY shown in this ProfilePlotting.
<code>ImageFigure getLine()</code>	Returns the straight line along which we wish to plot
<code>Double getBegin()</code>	Returns the begin of the drawn line in pixel coordinates.
<code>Double getEnd()</code>	Returns the end of the drawn line in pixel coordinates.
<code>int getClicks()</code>	Returns the number of valid clicks (i.e. in the image) you made.
<code>String getSkyCoordinates()</code>	Returns the String version of the sky coordinates
<code>String getPixelCoordinates()</code>	Returns the String version of the pixel coordinates of the begin.
<code>boolean isInImage(double pixX, double pixY)</code>	Checks whether the given pixel coordinates lies
<code>DoubleId getPlotPoints()</code>	

Methods
Returns the intensity of the pixels along the straight line, <code>ArrayList</code> <code>getImageFigures()</code>
Returns all ImageFigures used for this ProfilePlotting (the

API details

Constructors

<code>ProfilePlotting(ImageAnalysisToolbox toolbox)</code>
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 <code>ImageAnalysisToolbox toolbox</code> [INPUT, MANDATORY]

<code>ProfilePlotting(boolean useAsComponent, ImageAnalysisToolbox toolbox)</code>
component-based, a window for plotting the intensity is opened; otherwise nothing will be shown.
Arguments <code>boolean useAsComponent</code> [INPUT, MANDATORY] <code>ImageAnalysisToolbox toolbox</code> [INPUT, MANDATORY]

Methods

<code>setBegin(Double begin)</code>
ProfilePlotting to the given point (in mouse coordinates).
Argument <code>Double begin</code> [INPUT, MANDATORY]

<code>setEnd(Double end)</code>
ProfilePlotting to the given point (in mouse coordinates).
Argument <code>Double end</code> [INPUT, MANDATORY]

<code>PlotXY getPlot()</code>
Return <code>PlotXY</code>
Returns the PlotXY shown in this ProfilePlotting.

<code>ImageFigure getLine()</code>
the intensity in this ProfilePlotting.
Return <code>ImageFigure</code>
The straight line along which we wish to plot the intensity in this ProfilePlotting.

```
Double getBegin()
```

Return**Double**

Returns the begin of the drawn line in pixel coordinates.

```
Double getEnd()
```

Return**Double**

Returns the end of the drawn line in pixel coordinates.

```
int getClicks()
```

Return**int**

Returns the number of valid clicks (i.e. in the image) you made.

```
String getSkyCoordinates()
```

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()
```

Return**String**

Returns the String version of the pixel coordinates of the begin and end of the drawn line.

```
boolean isInImage(double pixX, double pixY)
```

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()
```

associated with this ProfilePlotting.

Return

Double1d `getPlotPoints()`**Double1d**

Returns the intensity of the pixels along the drawn straight line associated with this ProfilePlotting.


ArrayList `getImageFigures()`

straight line on the image).

Return**ArrayList**

Returns all ImageFigures used for this ProfilePlotting (the straight line on the image).

3.240. ProfileTask

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

Description

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 : "beginRA"]
String beginDec [INPUT, OPTIONAL, default=Default value : "beginDec"]
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 : "beginRA"]

The right ascension of the begin of the straight line.

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

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 : "endDec"]

The declination of the end of the straight line.

Profile profile [OUTPUT, MANDATORY, default=Default value : Profile()]

The profile.


3.241. 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 creator=='Scott')
<pre>creator=='Scott') q=Query("foo.Product", "type=='AbcProduct' and creator=='Scott'")</pre>

3.242. RadialVelocity

Full Name:	herschel.ia.toolbox.astro.RadialVelocity
Type:	Java Class - 
Import:	from herschel.ia.toolbox.astro import RadialVelocity
Category:	toolbox

Description

The RadialVelocity utility.

Calculates S/C radial velocity projection on the direction of the pointing to help in Doppler correction of observed frequencies by HIFI.

Examples

Example 1: #Get projection with explicit vectors at t= 3, see SCR-2949

```
from herschel.share.fltdyn.math import Vector3
from herschel.ia.toolbox.astro import RadialVelocity
v = Vector3(1,2,3) #the S/C velocity
p = Vector3(1,0,0) #the S/C pointing vector
res = RadialVelocity.getProjection(v, p, 3)
```

Example 2: #A few samples of conversion from some commonly used types to Vector3

```
#Direction to Vector3
from herschel.share.fltdyn.math import Direction
d = Direction(0.2, 0.5)
p = Vector3(d)
# Double1d (sized 3) to Vector3
d = Double1d([1,2,3])
v = Vector3(d.array)
```

Example 3: #Providing only the Observation (context) and returning multiple data items, see SCR-4967

```
from herschel.ia.toolbox.astro import RadialVelocity
c = ObservationContext()
(ra,dec,v)= RadialVelocity.getPointingAndProjection(c, 17)
```


See also

- [???](#)

History

- 2008-10-10 - JDS: first interface candidate release

3.243. 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]

3.244. 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]

3.245. 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=Double1d(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 = Double1d(100).apply(f)
# To generate 100 Integer values in [0,100)
fx = RandomUniform(100.0)
yint = Int1d(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]

3.246. 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

Methods
<code>setBegin(Double begin)</code> Sets the begin of the straight line, associated with this
<code>setEnd(Double end)</code> Sets the end of the straight line, associated with this
<code>setPlot()</code> initiate the PlotXY, shown in this SpectrumPlotting.
<code>ImageFigure getLine()</code> Returns the straight line (ImageFigure), of which we wish to plot
<code>Double getBegin()</code> Returns the begin of the drawn line in PixelCoordinates.
<code>Double getEnd()</code> Returns the end of the drawn line in PixelCoordinates.

API details

Methods

<code>setBegin(Double begin)</code> Velocity position map to the given point (in UserCoordinates). Argument <code>Double begin</code> [INPUT, MANDATORY]
<code>setEnd(Double end)</code> Velocity position map to the given point (in UserCoordinates). Argument <code>Double end</code> [INPUT, MANDATORY]
<code>setPlot()</code>
<code>ImageFigure getLine()</code> the intensity in this IntegratedMapDisplay. Return

ImageFigure `getLine()`

ImageFigure

The straight line (ImageFigure), of which we wish to plot the intensity in this IntegratedMapDisplay.

Double `getBegin()`

Return

Double

Returns the begin of the drawn line in PixelCoordinates.


Double `getEnd()`

Return

Double

Returns the end of the drawn line in PixelCoordinates.

3.247. 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 raw averaged spectrum from a set of pixels of a cube as Double3d.

3.249. REBIN

Full Name:	herschel.ia.numeric.toolbox.interp.Rebin
Alias:	REBIN
Type:	Java Class - 
Import:	from herschel.ia.numeric.toolbox.interp import Rebin

Description

Resize a vector or array to dimensions given by the parameters di. The

maximum dimension supported is 5. The input array is a numeric type of array with rank from 1 to 5. The output array has the same rank and same type with new dimensions.

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)
```

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]

Properties
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 magnifying 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 requires more time.
--


See also

- ???

History

- August 30, 2006 - first version

3.250. RectangleHistogramExplorer

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

Description

An explorer for RectangleHistogramProducts.

API Summary

Constructors
<code>RectangleHistogramExplorer()</code> The constructor of a new RectangleHistogramExplorer.
<code>RectangleHistogramExplorer(Object object)</code> The constructor of a new RectangleHistogramExplorer associated
Methods
<code>String getName()</code> Returns the name for this RectangleHistogramExplorer.
<code>String getDescription()</code> Returns the description for this RectangleHistogramExplorer.
<code>boolean canHandle(Class className)</code> Checks whether this RectangleHistogramExplorer can handle objects of the
<code>setObject(Object object)</code> Sets the object for this RectangleHistogramExplorer to the given object.
<code>RectangleHistogramProduct getObject()</code> Returns the object for this RectangleHistogramExplorer.
<code>JTable getParameterTable()</code> Returns the parameter table for this

API details

Constructors


<code>RectangleHistogramExplorer()</code>
<code>RectangleHistogramExplorer(Object object)</code> with the given object.
Argument <code>Object object</code> [INPUT, MANDATORY]

Methods

<code>String getName()</code>
Return

String getName()
String Returns the name for this RectangleHistogramExplorer.
String getDescription()
Return String Returns the description for this RectangleHistogramExplorer.
boolean canHandle(Class className)
given class. Argument Class className [INPUT, MANDATORY] Return boolean Returns true of this RectangleHistogramExplorer can handle objects of the given class; false otherwise.
setObject(Object object)
Argument Object object [INPUT, MANDATORY]
RectangleHistogramProduct getObject()
Return RectangleHistogramProduct Returns the object for this RectangleHistogramExplorer.
JTable getParameterTable()
RectangleHistogramExplorer. Return JTable Returns the parameter table for this RectangleHistogramExplorer.

3.251. RectangleHistogramPanel

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

Description

A panel for the RectangleHistogramPanel.

API Summary

Constructor
<code>RectangleHistogramPanel()</code> The construction of a new RectangleHistogramPanel.
Methods
<code>drawFigure()</code> Draws the rectangle on the image associated with this
<code>updateFigure()</code> Updates the rectangle associated with this
<code>trigger()</code> Triggers the execution of the task associated
<code>updateHistogram()</code> Updates the histogram associated with this

API details

Constructor

<code>RectangleHistogramPanel()</code>

Methods

<code>drawFigure()</code>	RectangleHistogramPanel.
<code>updateFigure()</code>	RectangleHistogramPanel.
<code>trigger()</code>	with this RectangleHistogramPanel.
<code>updateHistogram()</code>	RectangleHistogramPanel.

3.252. 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><code>RectangleHistogramProduct()</code></p> <p>The constructor of a new RectangleHistogramProduct.</p>
Methods
<p><code>setUpperLeftCorner(double upperLeftX, double upperLeftY)</code></p> <p>Sets the upper left corner for this RectangleHistogramProduct</p>
<p><code>setUpperLeftCorner(double upperLeftX, double upperLeftY, String upperLeftRA, String upperLeftDec)</code></p> <p>Sets the upper left corner for this RectangleHistogramProduct</p>
<p><code>setDimensions(double widthPixels, double heightPixels)</code></p> <p>Sets the dimensions for this RectangleHistogramProduct to the</p>
<p><code>setDimensions(double widthPixels, double heightPixels, double widthArcsec, double heightArcsec)</code></p> <p>Sets the dimensions for this RectangleHistogramProduct to the</p>
<p><code>DoubleId getUpperLeftCornerPixelCoordinates()</code></p> <p>Returns the upper left corner for this RectangleHistogramProduct in</p>
<p><code>StringId getUpperLeftCornersSkyCoordinates()</code></p> <p>Returns the upper left corner for this RectangleHistogramProduct in</p>
<p><code>double getWidthPixels()</code></p> <p>Returns the width for this RectangleHistogramProduct in pixels.</p>
<p><code>double getWidthArcsec()</code></p> <p>Returns the width for this RectangleHistogramProduct in arcsec.</p>
<p><code>double getHeightPixels()</code></p> <p>Returns the height for this RectangleHistogramProduct in pixels.</p>
<p><code>double getHeightArcsec()</code></p> <p>Returns the height for this RectangleHistogramProduct in arcsec.</p>

API details

Constructor

<code>RectangleHistogramProduct()</code>
--

Methods

setUpperLeftCorner(double upperLeftX, double upperLeftY)

to the given pixel coordinates.

Arguments

double **upperLeftX** [INPUT, MANDATORY]

double **upperLeftY** [INPUT, MANDATORY]

setUpperLeftCorner(double upperLeftX, double upperLeftY, String upperLeftRA, String upperLeftDec)

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)

given dimensions in pixels.

Arguments

double **widthPixels** [INPUT, MANDATORY]

double **heightPixels** [INPUT, MANDATORY]

setDimensions(double widthPixels, double heightPixels, double widthArcsec, double heightArcsec)

given dimensions in pixels and arcsec.

Arguments

double **widthPixels** [INPUT, MANDATORY]

double **heightPixels** [INPUT, MANDATORY]

double **widthArcsec** [INPUT, MANDATORY]

double **heightArcsec** [INPUT, MANDATORY]

Double1d **getUpperLeftCornerPixelCoordinates()**

pixel coordinates.

Return

Double1d

Returns the upper left corner for this RectangleHistogramProduct in pixel coordinates.

String1d **getUpperLeftCornerSkyCoordinates()**

sky coordinates.

Return

String1d

Returns the upper left corner for this RectangleHistogramProduct in sky coordinates.

double getWidthPixels()**Return****double**

Returns the width for this RectangleHistogramProduct in pixels.

double getWidthArcsec()**Return****double**

Returns the width for this RectangleHistogramProduct in arcsec.

double getHeightPixels()**Return****double**

Returns the height for this RectangleHistogramProduct in pixels.

double getHeightArcsec()**Return****double**

Returns the height for this RectangleHistogramProduct in arcsec.

3.253. 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 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.

3.254. RectangularSkyAperturePhotometryExplorer

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

Description

An explorer for RectangularSkyAperturePhotometryProducts.

API Summary

Constructors
<code>RectangularSkyAperturePhotometryExplorer()</code> The constructor of a new RectangularSkyAperturePhotometryExplorer.
<code>RectangularSkyAperturePhotometryExplorer(Object object)</code> The constructor of a new RectangularSkyAperturePhotometryExplorer
Methods
<code>boolean canHandle(Class className)</code> Checks whether this RectangularSkyAperturePhotometryExplorer can
<code>RectangularSkyAperturePhotometryProduct getObject()</code> Returns the object for this RectangularSkyAperturePhotometryExplorer.
<code>JTable getParameterTable()</code> Constructs and returns the parameter table for this
<code>JPanel getPlotPanel()</code> Constructs and returns the plot panel for this

API details

Constructors

<code>RectangularSkyAperturePhotometryExplorer()</code>
<code>RectangularSkyAperturePhotometryExplorer(Object object)</code> associated with the given object.
Argument <code>Object object</code> [INPUT, MANDATORY]

Methods

<code>boolean canHandle(Class className)</code> handle objects of the given class.
Argument <code>Class className</code> [INPUT, MANDATORY]
Return

```
boolean canHandle(Class className)
```

boolean

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

```
RectangularSkyAperturePhotometryProduct getObject()
```

Return

RectangularSkyAperturePhotometryProduct

Returns the object for this RectangularSkyAperturePhotometryExplorer.

```
JTable getParameterTable()
```

RectangularSkyAperturePhotometryExplorer.

Return

JTable

Returns the parameter table for this RectangularSkyAperturePhotometryExplorer.

```
JPanel getPlotPanel()
```


RectangularSkyAperturePhotometryExplorer.

Return

JPanel

Returns the plot panel for this RectangularSkyAperturePhotometryExplorer.

3.255. RectangularSkyAperturePhotometryPanel

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

Description

A panel for the RectangularSkyAperturePhotometryTask.

API Summary

Constructor
<p><code>RectangularSkyAperturePhotometryPanel()</code></p> <p>The constructor of a new RectangularSkyAperturePhotometryPanel.</p>
Methods
<p><code>JPanel getAperturesPanel()</code></p> <p>Returns the panel where to specify the target radius and</p>
<p><code>JComponent getSkyApertureComponent()</code></p> <p>Returns the panel where it is explained how to</p>
<p><code>transferFromModifierToSelectionMap()</code></p> <p>Transfers information from the modifier map to the</p>
<p><code>drawRectangle()</code></p> <p>Allows to draw a rectangle on the image associated</p>
<p><code>int getNbOfRectanglePoints()</code></p> <p>Returns the number of points that were confirmed</p>
<p><code>increaseNbOfRectanglePoints()</code></p> <p>Increases the number of rectangle points that were</p>
<p><code>drawFigures()</code></p> <p>Draws the circles on the image for this</p>
<p><code>moveConfirmedFigures()</code></p> <p>Allows the user to drag the figures bounding the</p>

API details

Constructor

<code>RectangularSkyAperturePhotometryPanel()</code>
--

Methods

<code>JPanel getAperturesPanel()</code>
where it is explained how to specify the rectangular sky aperture for this RectangularSkyAperturePhotometryPane.

JPanel `getAperturesPanel()`**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()`

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`.

transferFromModifierToSelectionMap()

variable selection map associated with this `RectangularSkyAperturePhotometryPanel`.

drawRectangle()

with this `RectangularSkyAperturePhotometryPanel`.

int `getNbOfRectanglePoints()`

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()

confirmed before on the image for this `RectangularSkyAperturePhotometryPanel`.


drawFigures()

`RectangularSkyAperturePhotometryPanel`.

moveConfirmedFigures()

apertures for this `RectangularSkyAperturePhotometryPanel`.

3.256. 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	
<code>RectangularSkyAperturePhotometryProduct()</code>	The constructor of a new RectangularSkyAperturePhotometryProduct.
Methods	
<code>setUpperLeftCorner(double upperLeftX, double upperLeftY)</code>	Sets the upper left corner for this RectangularSkyAperturePhotometryProduct
<code>setUpperLeftCorner(double upperLeftX, double upperLeftY, String upperLeftRA, String upperLeftDec)</code>	Sets the upper left corner for this RectangularSkyAperturePhotometryProduct.
<code>setDimensions(double widthPixels, double heightPixels)</code>	Sets the dimensions for this RectangularSkyAperturePhotometryProduct to the
<code>setDimensions(double widthPixels, double heightPixels, double widthArcsec, double heightArcsec)</code>	Sets the dimensions for this RectangularSkyAperturePhotometryProduct to the
<code>DoubleId getUpperLeftCornerPixelCoordinates()</code>	Returns the upper left corner for this RectangularSkyAperturePhotometryProduct in
<code>StringId getUpperLeftCornersSkyCoordinates()</code>	Returns the upper left corner for this RectangularSkyAperturePhotometryProduct
<code>double getWidthPixels()</code>	Returns the width for this RectangularSkyAperturePhotometryProduct
<code>double getWidthArcsec()</code>	Returns the width for this RectangularSkyAperturePhotometryProduct
<code>double getHeightPixels()</code>	Returns the height for this RectangularSkyAperturePhotometryProduct
<code>getHeightArcsec()</code>	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()
```

Methods

```
setUpperLeftCorner(double upperLeftX, double upperLeftY)
```

to the given pixel coordinates.

Arguments

double **upperLeftX** [INPUT, MANDATORY]

double **upperLeftY** [INPUT, MANDATORY]

```
setUpperLeftCorner(double upperLeftX, double upperLeftY, String
upperLeftRA, String upperLeftDec)
```

Arguments

double **upperLeftX** [INPUT, MANDATORY]

double **upperLeftY** [INPUT, MANDATORY]

String **upperLeftRA** [INPUT, MANDATORY]

String **upperLeftDec** [INPUT, MANDATORY]

```
setDimensions(double widthPixels, double heightPixels)
```

given dimensions in pixels.

Arguments

double **widthPixels** [INPUT, MANDATORY]

double **heightPixels** [INPUT, MANDATORY]

```
setDimensions(double widthPixels, double heightPixels, double
widthArcsec, double heightArcsec)
```

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()
```

pixel coordinates.

Return

DoubleId

Returns the upper left corner for this RectangularSkyAperturePhotometryProduct in pixel coordinates.

```
StringId getUpperLeftCornerSkyCoordinates()
```

in sky coordinates.

StringId getUpperLeftCornerSkyCoordinates()

Return

StringId

Returns the upper left corner for this RectangularSkyAperturePhotometryProduct in sky coordinates.

double getWidthPixels()

in pixels.

Return

double

Returns the width for this RectangularSkyAperturePhotometryProduct in pixels.

double getWidthArcsec()

in arcsec.

Return

double

Returns the width for this RectangularSkyAperturePhotometryProduct in arcsec.

double getHeightPixels()

in pixels.

Return


double

Returns the height for this RectangularSkyAperturePhotometryProduct in pixels.

getHeightArcsec()

in arcsec.

3.257. RectangularSkyAperturePhotometryTask

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

Description

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 : "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]
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 : "upperLeftRA"]
String minDec [INPUT, OPTIONAL, default=Default value : "upperLeftDec"]
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.

<code>Double heightArcsec [INPUT, OPTIONAL, default=Default value : NaN]</code>

The input height of the rectangular sky aperture in arcsec.

3.258. 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

```
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")
```

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.

3.259. 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 standard (default) scheme uses a trapezoidal integration scheme in combination with linear interpolation. An alternative scheme consists of using an euler integration scheme in combination with nearest neighbor interpolation.

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 a 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.

Example

Example 1: from Jide:

```
resampled1 = resample(ds=spectra, density=True, resolution=1.0)
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)
```

API Summary

Properties
SpectrumContainer ds [INPUT, MANDATORY, default=no default value.]
SpectrumContainer result [OUTPUT, MANDATORY, default=no default value.]
String scheme [INPUT, OPTIONAL, default=no default value.]
Double1d[] grid [INPUT, OPTIONAL, default=no default value.]
double resolution [INPUT, OPTIONAL, default=no default value.]
Boolean density [INPUT, OPTIONAL, default=no default value.]

API details

Properties

SpectrumContainer ds [INPUT, MANDATORY, default=no default value.]

The input container to be resampled.

SpectrumContainer result [OUTPUT, MANDATORY, default=no default value.]

The output container with the re-sampled spectra.

String scheme [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").

- 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.

Double1d[] grid [INPUT, OPTIONAL, default=no default value.]

The new frequency grid the spectra should be resampled to. It is specified as a Double1d 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.

double resolution [INPUT, OPTIONAL, default=no default value.]

In case no output frequency grid is specified, the task creates an output grid with this given resolution (width) by determining, for each sub-segment, the smallest minimum frequency and the largest maximum frequency. The width is specified in the same units as the original frequency grid.


Boolean 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, ...

3.260. 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](#)

3.261. 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

```
name space
def my_function():
    restore("test.sav", space=locals())
    print locals()
```

API Summary

Jython Syntax

```
restore(<filename>[, <space>])
```

Properties

`String filename` [INPUT, YES, default=No default value]

`PyStringMap space` [INPUT, NO, default=""]

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, YES, default=No default value]
The name of the file where the variables along with their values are saved

PyStringMap space [INPUT, NO, default=""]
A jython dictionary (PyStringMap) containing the namespace to look for, The choices available are: <ol style="list-style-type: none">1. "local()" for reading variables from the local namespace (i.e. from where the function is used)2. "global()" for reading variables from the module where the function is used When no space is specified the top level namespace is updated.


See also

- ???

History

- 2004-07-13 - NdC: first release.

3.262. 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()
```

Miscellaneous

No miscellaneous


See also

- [resume](#)

History

- 2007-10-11 - NdC: first release.

3.263. 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
<pre>print REVERSE(Double1d.range(3)) # [2.0,1.0,0.0]</pre>

API Summary


Jython Syntax
<code><y>=REVERSE(<x>)</code>
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.

3.264. 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.appendColumn(Int1d.range(255))
```

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 )<br>
Counterclockwise nearest neighbour rotation
result = Rotate(-54.0)( array )<br>
Clockwise bicubic rotation
result = Rotate(54.0, Rotate.BICUBIC) ( array ) or
result = Rotate(54.0, 3) ( array )<br>
Counterclockwise bilinear rotation
result = Rotate(54.0, Rotate.BILINEAR, false) ( array )<br><br>
array = Int2d();
for i in range(5):
    array.appendColumn(Int1d.range(5))<br>
print array
[
[0,0,0,0,0],
[1,1,1,1,1],
[2,2,2,2,2],
```

Example 1: apply Rotate to a Int2d array

```
[3,3,3,3],
[4,4,4,4]
]
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) )<br>
or<br>
<y> = Rotate(angle, interpolation method, clockwise)( <x> )<br>
```

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

the interpolation method interpolation method [INPUT, NOT_MANDATORY, default=Rotate.NEAREST_NEIGHBOR]

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 counterclockwise 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.

3.265. rotateTask

Full Name:	herschel.ia.toolbox.image.RotateTask
Alias:	rotateTask
Type:	Java Task - 
Import:	from herschel.ia.toolbox.image import RotateTask
Category:	task/image

Description

The Rotate task for Image.

RotateTask is 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:

$$r(x) = (a + 2)|x|^3 - (a + 3)|x|^2 + 1, \quad 0 \leq |x| < 1$$

$$r(x) = a|x|^3 - 5a|x|^2 + 8a|x| - 4a, \quad 1 \leq |x| < 2$$

$$r(x) = 0, \quad \text{otherwise}$$

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:

$$r(x) = (a + 2)|x|^3 - (a + 3)|x|^2 + 1, \quad 0 \leq |x| < 1$$

$$r(x) = a|x|^3 - 5a|x|^2 + 8a|x| - 4a, \quad 1 \leq |x| < 2$$

$$r(x) = 0, \quad \text{otherwise}$$

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)

```
rotateTask(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

```
rotateTask(image = im, angle = 12.2, interpolation = Rotate.INTERP_BICUBIC,
  subsampleBits = 32)
```

API Summary

Jython Syntax

```
rotateTask(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)

<code>Image rotatedImage [OUTPUT, MANDATORY, default=No default value]</code>

The rotated Image

3.266. 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](#)

3.267. 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.

3.268. 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 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 variable names and values

```
save("foo.sav")
```

Example 2: save specified 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())
```

API Summary

Jython Syntax

```
save(<filename>[, <variable>][, <space>])
```

Properties

String filename [INPUT, YES, default=No default value]

String variable [INPUT, NO, default=""]

PyStringMap space [INPUT, NO, default=""]

Limitations

Only Serializable variable can be saved

Miscellaneous

No miscellaneous

API details

Properties

String filename [INPUT, YES, default=No default value]

The name of the file where to store the values of variables

String variable [INPUT, NO, default=""]
--

A string containing a comma separated list of variable names
--

PyStringMap space [INPUT, NO, default=""]
--

A jython dictionary (PyStringMap) containing the namespace to look for, The choices available are:
--

- | |
|---|
| <ol style="list-style-type: none">1. "local()" for reading variables from the local namespace (i.e. from where the function is used)2. "global()" for reading variables from the module where the function is used |
|---|

When no space is specified the top level namespace is updated.
--


See also

- ???

History

- 2004-07-13 - NdC: first release.

3.269. scaleTask

Full Name:	herschel.ia.toolbox.image.ScaleTask
Alias:	scaleTask
Type:	Java Task - 
Import:	from herschel.ia.toolbox.image import ScaleTask
Category:	task/image

Description

The Scale task for Images.

ScaleTask is a task which scales 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:

$$r(x) = (a + 2)|x|^3 - (a + 3)|x|^2 + 1, \quad 0 \leq |x| < 1$$

$$r(x) = a|x|^3 - 5a|x|^2 + 8a|x| - 4a, \quad 1 \leq |x| < 2$$

$$r(x) = 0, \quad \text{otherwise}$$

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:

$$r(x) = (a + 2)|x|^3 - (a + 3)|x|^2 + 1, \quad 0 \leq |x| < 1$$

$$r(x) = a|x|^3 - 5a|x|^2 + 8a|x| - 4a, \quad 1 \leq |x| < 2$$

$$r(x) = 0, \quad \text{otherwise}$$

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)

```
scaleTask(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

```
scaleTask(image = im, x = 1.4, y = 0.4, interpolation = Scale.INTERP_BICUBIC,
  subsampleBits = 32)
```

API Summary

Jython Syntax

```
scaleTask(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

See also

- [???](#)

3.270. 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 ([@link SpectrumContainer](#)) data structure and pack them in a new [@link SpectrumContainer](#)-object of the same runtime type.

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.

The class is a wrapper of the [@link Select](#)-class.

Example

Example 1: from jide:

```
from herschel.ia.toolbox.spectrum import SelectSpectrum
from herschel.ia.toolbox.spectrum.selections.models import RangesSelectionModel
select = SelectSpectrum()
selectedBBType = select(ds=data, selection_lookup={"bbtype": [3413]})
selectedBBTypeAndBuffer = select(ds=data, selection_lookup={"bbtype":
[3515], "buffer": [2]})
selectChopper = RangesSelectionModel("Chopper", [5.9], 0.1)
selectedChopperPositions = select(ds=data, selection=selectChopper)
```

API Summary

Properties
SpectrumContainer ds [INPUT, OPTIONAL, default=no default value.]
SegmentSelection segments [INPUT, OPTIONAL, default=no default value.]
SelectionModel selection [INPUT, OPTIONAL, default=no default value.]
PyDictionary Map> selection_lookup [INPUT, OPTIONAL, default=no default value.]
PyList selection_index [INPUT, OPTIONAL, default=No default value.]

Properties
<code>SpectrumContainer result [OUTPUT, OPTIONAL, default=no default value]</code>

API details

Properties

<code>SpectrumContainer ds [INPUT, OPTIONAL, default=no default value.]</code>
Input container to be processed by the task: in case the task is used as a many-to-one operation (eg avg) or as scalar operation.

<code>SegmentSelection segments [INPUT, OPTIONAL, default=no default value.]</code>
Specify an instance of a SegmentSelection which gives the information on what segments for each point spectrum included in the container and, possibly, what ranges / masks should be considered for the processing.

<code>SelectionModel selection [INPUT, OPTIONAL, default=no default value.]</code>
Specify an instance of any kind of SelectionModel here.

<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>SpectrumContainer result [OUTPUT, OPTIONAL, default=no default value]</code>
Result object containing the results of the operation applied.


See also

- [SpectrumTask](#)

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

3.271. 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
<code>Product load(InputStream stream)</code> Loads a Product from an input stream.
<code>Product load(String fileName)</code> Loads a Product from a serialized file.
<code>save(String fileName, [Optionally derived] Product product)</code> Saves a Product to a serialized file.
<code>createOutputStream(String fileName)</code> Creates an object output stream for storing in a file.
<code>createOutputStream(OutputStream file)</code> Creates an object output stream for storing in a file.
<code>createOutputStream(OutputStream stream)</code> Creates an object output stream for storing in a file.
<code>createInputStream(String fileName)</code> Creates an object input stream for reading from a file.
<code>createInputStream(String file)</code> Creates an object input stream for reading from a file.
<code>createInputStream(InputStream stream)</code> Creates an object input stream for reading from a file.

API details

Methods

Product load(InputStream 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 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 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 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
String fileName [INPUT, MANDATORY]

createOutputStream(OutputStream 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

createOutputStream(OutputStream file)

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 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 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 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 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](#)

3.272. 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 pool.

API details

Constructor

```
SerialClientPool(String hostName, String portNumber, String id)
```

Arguments

String hostName [INPUT, MANDATORY, default=The URL at which the 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

SerialClientPool(**String** hostName, **String** portNumber, **String** id)

String id [INPUT, MANDATORY, default=The identifier for the remote pool]


"simple.mypool"

Return

SerialClientPool

An instance of the SerialClientPool.

3.273. 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.

3.274. 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)

3.275. 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)

3.276. 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)

3.277. 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)

3.278. 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)

3.279. 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 $n \times$ (standard deviation) larger than the next smaller surrounding value in a box of user defined size. Sigclip replaces these higher values either with the median or the mean of the surrounding values. Sigclip does not include the investigated Pixel for the computation of the median, mean or the sigma. The user has the choice of 5 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 boxsize), 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)

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]
```

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]

Properties

determines `mode` [INPUT, if median or mean is used to replace sigclipped values, 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 sigclipped 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.

3.280. 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
<code>SimpleCube()</code> The standard constructor
<code>SimpleCube(SimpleCube copy)</code> The copy constructor
<code>SimpleCube(String description)</code> Constructor with a description
Methods
<code>Wcs getWcs()</code> Returns the World Coordinates System.
<code>setImage(AbstractOrdered3dData image, Unit unit, String description)</code> Sets the cube.
<code>setImage(AbstractOrdered3dData image, Unit unit)</code> Sets the cube.
<code>setWcs(Wcs wcs)</code> Sets the world coordinates system.
<code>getIntensity(int depth, double row, double column)</code> Returns the intensity
<code>double getIntensityWorldCoordinates(int depth, double x, double y)</code> Returns the intensity of the cube
<code>setIntensity(int depth, int row, int column, Number value)</code> Sets the intensity of the image.
<code>Number getPixel(int depth, int row, int column)</code>

Methods	
	Returns 1 pixel value.
<code>int getDepth()</code>	Returns the number of layers
<code>Cube getPreview(float res)</code>	Returns a preview of the cube
<code>AbstractOrdered3dData getError()</code>	Returns the error of the cube as a Numeric3d.
<code>AbstractOrdered3dData getExposure()</code>	Returns the exposure of the cube as a Numeric3d.
<code>Flag getFlag()</code>	Returns the flag of the cube.
<code>int getHeight()</code>	Returns the height.
<code>AbstractOrdered3dData getImage()</code>	returns the cube as a Numeric3d.
<code>Unit getUnit()</code>	Returns the unit.
<code>int getWidth()</code>	Returns the width
<code>boolean hasError()</code>	Checks whether the cube has an error.
<code>boolean hasExposure()</code>	Checks whether the cube has an exposure.
<code>boolean hasFlag()</code>	Checks when the cube has a flag.
<code>setUnit(Unit unit)</code>	Sets the unit.
<code>removeError()</code>	Removes the error.
<code>removeExposure()</code>	Removes the exposure.
<code>removeFlag()</code>	Removes the flag.
<code>setImage(AbstractOrdered3dData cube)</code>	Sets the cube.
<code>setError(AbstractOrdered3dData error)</code>	Sets the error.
<code>setError(AbstractOrdered3dData error, String description)</code>	Sets the error.
<code>setFlag(Flag flag, String description)</code>	Sets the flag.

Methods
<code>setFlag(Flag flag)</code> Sets the flag.
<code>setExposure(AbstractOrdered3dData exposure, String description)</code> Sets the exposure.
<code>setExposure(AbstractOrdered3dData exposure)</code> Sets the exposure.
<code>int[] getDimensions()</code> Returns the dimension.

API details

Constructors

<code>SimpleCube()</code>
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.
<pre>image=SimpleCube(description="ngc 6992", image=im, flag=flag, errors=er, unit=unit, wcs=wcs)</pre>

<code>SimpleCube(SimpleCube copy)</code>
Creates a new SimpleCube with the same contents of the given SimpleCube.
Argument <code>SimpleCube copy</code> [INPUT, MANDATORY]

<code>SimpleCube(String description)</code>
Creates a SimpleCube with a given description.
Argument <code>String description</code> [INPUT, MANDATORY]

Methods

<code>Wcs getWcs()</code>
Returns the World Coordinates System of the image.
Return <code>Wcs</code> The World Coordinates System of the image.

<code>setImage(AbstractOrdered3dData image, Unit unit, String description)</code>
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 are reset to 0.0 and the mask is set to have no bad pixels. If the new

setImage(AbstractOrdered3dData image, Unit unit, String description)

cube has the same dimensions, then the old values for the errors and the mask 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. 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 are reset to 0.0 and the mask is set to have no bad pixels. If the new cube has the same dimensions, then the old values for the errors and the mask are kept (but the unit of the errors is not).

Arguments

`AbstractOrdered3dData image` [INPUT, MANDATORY]
`Unit unit` [INPUT, MANDATORY]

setWcs(Wcs wcs)

Sets the world coordinates system of the cube.

Argument

`Wcs wcs` [INPUT, MANDATORY]

getIntensity(int depth, double row, double column)

Returns the intensity of the SimpleCube at a given point. If the pixel is masked 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 vube at a given point in world coordinates. If the pixel is masked 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 at a given point.

Arguments

setIntensity(int depth, int row, int column, [Number](#) value)

int **depth** [INPUT, MANDATORY]
 int **row** [INPUT, MANDATORY]
 int **column** [INPUT, MANDATORY]
[Number](#) **value** [INPUT, MANDATORY]

[Number](#) getPixel(int depth, int row, int column)

Returns 1 pixel value of the image.

Arguments

int **depth** [INPUT, MANDATORY]
 int **row** [INPUT, MANDATORY]
 int **column** [INPUT, MANDATORY]

Return

[Number](#)

1 pixel value of the image.

int getDepth()

Returns the depth of this SimpleCube. This is the first axis (slowly varying axis).

Return

int

Returns the depth of this SimpleCube.

[Cube](#) getPreview(float res)

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.

Argument

float **res** [INPUT, MANDATORY]

Return

[Cube](#)

The preview of the cube

[AbstractOrdered3dData](#) getError()

Returns the error of the cube as a Numeric3d containing the error of every pixel.

Return

[AbstractOrdered3dData](#)

The error

[AbstractOrdered3dData](#) getExposure()

Returns the exposure of the cube as a Numeric3d containing the exposure of every pixel.

Return

[AbstractOrdered3dData](#)

AbstractOrdered3dData `getExposure()`

The exposure

Flag `getFlag()`

Returns the flag of the cube.

Return**Flag**

The flag

int `getHeight()`

Returns the height of this SimpleCube.

Return**int**

Returns the height of this SimpleCube.

AbstractOrdered3dData `getImage()`

Returns the cube as a Numeric3d containing the data of the cube.

Return**AbstractOrdered3dData**

The cube as a Numeric3d

Unit `getUnit()`

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

int `getWidth()`

Returns the width of this SimpleImage.

Return**int**

Returns the width of this SimpleImage.

boolean `hasError()`

Returns true if the cube has an error.

Return**boolean**

True if the error is set.

boolean `hasExposure()`

Returns true if the cube has an exposure.

boolean hasExposure()
Return boolean True if the exposure is set.
boolean hasFlag()
Returns true if the cube has a flag. Return boolean True if the cube has a flag.
setUnit(Unit 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]
removeError()
Removes the error, if an error exists.
removeExposure()
Removes the exposure, if an exposure exists.
removeFlag()
Removes the flag, if a flag exists.
setImage(AbstractOrdered3dData cube)
Sets the cube. If the new cube has other dimensions than the original cube, the errors are reset to 0.0 and the flag is set to have no bad pixels. If the new cube has the same dimensions, then the old values for the errors and the mask are kept. Argument AbstractOrdered3dData cube [INPUT, MANDATORY]
setError(AbstractOrdered3dData 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 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 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 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)

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 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]

int[] getDimensions()


Returns the dimension (depth, width, height) of this SimpleCube.

Return

int[]

Returns the dimension (width, height, depth) of this SimpleCube.

3.281. 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 param

```
file = "path_to_file"
product=simpleFitsReader(file)
```

Example 2: SimpleFitsReaderTask with an optional reader type

```
file = "path_to_file"
reader = "STANDARD"
product=simpleFitsReader(file, reader)
```

API Summary

Jython Syntax

```
product=simpleFitsReader(file)
product=simpleFitsReader(file,reader) //with optional reader type
```

Properties

String file [INPUT, MANDATORY, default=null]

String reader [INPUT, OPTIONAL, default='HCSS_THEN_STANDARD']

Product product [OUTPUT, MANDATORY, default=null]

API details

Properties

String file [INPUT, MANDATORY, default=null]

FITS file.

String reader [INPUT, OPTIONAL, default='HCSS_THEN_STANDARD']

Reader type. Possible values: 'HCSS_THEN_STANDARD' (default), 'HCSS', 'STANDARD' 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.

<code>String reader [INPUT, OPTIONAL, default='HCSS_THEN_STANDARD']</code>
--

- | |
|--|
| <ul style="list-style-type: none">• HCSS_THEN_STANDARD : try to read the FITS Archive as an HCSS FITS file, if it fails then try to read it as a standard FITS file.• HCSS : read it as an HCSS FITS file.• STANDARD : read it as an STANDARD FITS file. |
|--|


<code>Product product [OUTPUT, MANDATORY, default=null]</code>
--

Product from FITS file.

History

- 2008-07-09 - JCS: first release
- 2008-08-18 - JDS: added optional parameter reader (type) to allow transparent failover reading

3.282. 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, user is warned and allowed to choose
- if no filename is finally chosen, flow is finished with `RuntimeException` "File is required."
- if the file already exists, user is asked to confirm overwriting
- if the user does not confirm overwriting, flow is finished with `RuntimeException` "Execution cancelled."
- Otherwise the product is saved in FITS format in the chosen file

Example

Example 1: Saving a product into a file

```
p = Product()
file = "path_to_file"
simpleFitsWriter(product=p,file=file)
```

API Summary

Jython Syntax

```
simpleFitsWriter(product,file)
```

Properties

```
Product product [INPUT, MANDATORY, default=null]
```

```
String file [INPUT, MANDATORY, default=null]
```

API details

Properties

```
Product product [INPUT, MANDATORY, default=null]
```

Product to be saved.

```
String file [INPUT, MANDATORY, default=null]
```

FITS filename to save the product into.

History

- 2008-07-08 - JCS: first release
- 2008-08-19 - JDS: Using PopUpDialog, SCR 4573 asking before overwriting

3.283. 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
<code>SimpleImage()</code> The standard constructor
<code>SimpleImage(SimpleImage copy)</code> Copy constructor
<code>SimpleImage(String description)</code> Constructor with a description
Methods
<code>setImage(AbstractOrdered2dData image, Unit unit, String description)</code> Sets the image.
<code>setWcs(Wcs wcs)</code> Sets the world coordinates system.
<code>Wcs getWcs()</code> Returns the World Coordinates System.
<code>setImage(AbstractOrdered2dData image, Unit unit)</code> Sets the image.
<code>getIntensity(double row, double column)</code> Returns the intensity
<code>double getIntensityWorldCoordinates(double x, double y)</code> Returns the intensity of the image
<code>setIntensity(int row, int column, Number value)</code> Sets the intensity of the image.
<code>Number getPixel(int row, int column)</code> Returns 1 pixel value.

Methods	
<code>Image</code> <code>getPreview(float res)</code>	Returns a preview of the image
<code>AbstractOrdered2dData</code> <code>getError()</code>	Returns the error of the image as a <code>Numeric2d</code> .
<code>AbstractOrdered2dData</code> <code>getExposure()</code>	Returns the exposure of the image as a <code>Numeric2d</code> .
<code>Flag</code> <code>getFlag()</code>	Returns the flag of the image.
<code>int</code> <code>getHeight()</code>	Returns the height.
<code>AbstractOrdered2dData</code> <code>getImage()</code>	returns the image as a <code>Numeric2d</code> .
<code>Unit</code> <code>getUnit()</code>	Returns the unit.
<code>int</code> <code>getWidth()</code>	Returns the width
<code>double</code> <code>getWavelength()</code>	Returns the reference wavelength
<code>double</code> <code>getWavelength(Length unit)</code>	Returns the reference wavelength
<code>boolean</code> <code>hasError()</code>	Checks whether the image has an error.
<code>boolean</code> <code>hasExposure()</code>	Checks whether the image has an exposure.
<code>boolean</code> <code>hasFlag()</code>	Checks when the image has a flag.
<code>setUnit(Unit unit)</code>	Sets the unit.
<code>setWavelength(double wavelength)</code>	Sets the reference wavelength.
<code>setWavelength(double wavelength, Length unit)</code>	Sets the reference wavelength.
<code>removeError()</code>	Removes the error.
<code>removeExposure()</code>	Removes the exposure.
<code>removeFlag()</code>	Removes the flag.
<code>setImage(AbstractOrdered2dData image)</code>	Sets the image.
<code>setError(AbstractOrdered2dData error)</code>	

Methods
Sets the error.
<code>setError(AbstractOrdered2dData error, String description)</code> Sets the error.
<code>setFlag(Flag flag, String description)</code> Sets the flag.
<code>setFlag(Flag flag)</code> Sets the flag.
<code>setExposure(AbstractOrdered2dData exposure, String description)</code> Sets the exposure.
<code>setExposure(AbstractOrdered2dData exposure)</code> Sets the exposure.
<code>double getFrequency()</code> Returns the reference frequency.
<code>double getFrequency(Frequency freq)</code> Returns the reference frequency.
<code>setFrequency(double frequency)</code> Sets the reference frequency.
<code>setFrequency(double frequency, Frequency unit)</code> Sets the reference frequency.
<code>int[] getDimensions()</code> Returns the dimension.

API details

Constructors

SimpleImage()
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>
SimpleImage(SimpleImage copy)
Constructor which makes a copy from an existent SimpleImage.
Argument
<code>SimpleImage copy</code> [INPUT, MANDATORY]
SimpleImage(String description)
Creates a SimpleImage with a given description.
Argument
<code>String description</code> [INPUT, MANDATORY]

Methods

setImage([AbstractOrdered2dData](#) image, [Unit](#) unit, [String](#) description)

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 are reset to 0.0 and the mask is set to have no bad pixels. If the new image has the same dimensions, then the old values for the errors and the mask are kept (but the unit of the errors is not).

Arguments

[AbstractOrdered2dData](#) image [INPUT, MANDATORY]

[Unit](#) unit [INPUT, MANDATORY]

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

setWcs([Wcs](#) wcs)

Sets the world coordinates system of the image.

Argument

[Wcs](#) wcs [INPUT, MANDATORY]

Wcs [getWcs](#)()

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. 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 are reset to 0.0 and the mask is set to have no bad pixels. If the new image has the same dimensions, then the old values for the errors and the mask 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 of the SimpleImage at a given point. If the pixel is masked 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 at a given point in world coordinates. If the pixel is masked out, NaN is given back.

Arguments

double x [INPUT, MANDATORY]

double y [INPUT, MANDATORY]

<code>double getIntensityWorldCoordinates(double x, double y)</code>
<p>Return</p> <p>double</p> <p>The intensity</p>
<code>setIntensity(int row, int column, Number value)</code>
<p>Sets the intensity of the image at a given point.</p> <p>Arguments</p> <p><code>int row</code> [INPUT, MANDATORY]</p> <p><code>int column</code> [INPUT, MANDATORY]</p> <p><code>Number value</code> [INPUT, MANDATORY]</p>
<code>Number getPixel(int row, int column)</code>
<p>Returns 1 pixel value of the image.</p> <p>Arguments</p> <p><code>int row</code> [INPUT, MANDATORY]</p> <p><code>int column</code> [INPUT, MANDATORY]</p> <p>Return</p> <p>Number</p> <p>1 pixel value of the image.</p>
<code>Image getPreview(float res)</code>
<p>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.</p> <p>Argument</p> <p><code>float res</code> [INPUT, MANDATORY]</p> <p>Return</p> <p>Image</p> <p>The preview of the image</p>
<code>AbstractOrdered2dData getError()</code>
<p>Returns the error of the image as a Numeric2d containing the error of every pixel.</p> <p>Return</p> <p>AbstractOrdered2dData</p> <p>The error</p>
<code>AbstractOrdered2dData getExposure()</code>
<p>Returns the exposure of the image as a Numeric2d containing the exposure of every pixel.</p> <p>Return</p> <p>AbstractOrdered2dData</p> <p>The exposure</p>

Flag <code>getFlag()</code>
Returns the flag of the image.
Return
Flag
The flag
int <code>getHeight()</code>
Returns the height of this SimpleImage.
Return
int
The height of this SimpleImage.
AbstractOrdered2dData <code>getImage()</code>
Returns the image as a Numeric2d containing the data of the image.
Return
AbstractOrdered2dData
The image as a Numeric2d
Unit <code>getUnit()</code>
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 <code>getWidth()</code>
Returns the width of this SimpleImage.
Return
int
The width of this SimpleImage.
double <code>getWavelength()</code>
Returns the reference wavelength of the image.
Return
double
The reference wavelength
double <code>getWavelength(Length unit)</code>
Returns the reference wavelength of the image.
Argument
Length unit [INPUT, MANDATORY]
Return

double getWavelength(Length unit)
double The reference wavelength
boolean hasError()
Returns true if the image has an error. Return boolean True if the error is set.
boolean hasExposure()
Returns true if the image has an exposure. Return boolean True if the exposure is set.
boolean hasFlag()
Returns true if the image has a flag. Return boolean True if the image has a flag.
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]
setWavelength(double wavelength)
Set the reference wavelength of the image in microns. Argument double wavelength [INPUT, MANDATORY]
setWavelength(double wavelength, Length unit)
Set the reference wavelength of the image. Arguments double wavelength [INPUT, MANDATORY] Length unit [INPUT, MANDATORY]
removeError()
Removes the error, if an error exists.
removeExposure()
Removes the exposure, if an exposure exists.

removeFlag()
Removes the flag, if a flag exists.
setImage(AbstractOrdered2dData image)
Sets the image. If the new image has other dimensions than the original image, the errors are reset to 0.0 and the flag is set to have no bad pixels. If the new image has the same dimensions, then the old values for the errors and the mask are kept.
Argument
AbstractOrdered2dData image [INPUT, MANDATORY]
setError(AbstractOrdered2dData 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 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 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]
String description [INPUT, MANDATORY]
setFlag(Flag 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)
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 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

```
setExposure(AbstractOrdered2dData exposure)
```

```
AbstractOrdered2dData exposure [INPUT, MANDATORY]
```

```
double getFrequency()
```

Returns the reference frequency of the image.

Return

double

The reference frequency of the image.

```
double getFrequency(Frequency freq)
```

Returns the reference frequency of the image.

Argument

```
Frequency freq [INPUT, MANDATORY]
```

Return

double

The reference frequency of the image.

```
setFrequency(double frequency)
```

Set the reference frequency of the image in gigahertz.

Argument

```
double frequency [INPUT, MANDATORY]
```

```
setFrequency(double frequency, Frequency unit)
```

Set the reference frequency of the image.

Arguments

```
double frequency [INPUT, MANDATORY]
```

```
Frequency unit [INPUT, MANDATORY]
```

```
int[] getDimensions()
```


Returns the dimension (width, height) of this SimpleImage.

Return

int[]

The dimension (width, height) of this SimpleImage.

3.284. SimplePool

Full Name:	herschel.ia.pal.pool.simple.SimplePool
Type:	Java Class - 
Import:	from herschel.ia.pal.pool.simple import SimplePool

Description

A simple implementation of a ProductPool which saves data in serialised (binary) form on your

local file system under the `${hcss.ia.pal.pool.simple.dir}` directory. By default, this system variable is defined as: `${HOME}/.hcss` (UNIX), or `C:\Documents and Settings \<user>\.hcss` (Windows). The pool itself is located under the subdirectory in accordance to the pool identifier (see examples). By default, the subdirectory will have a name "simple.default".

Examples

Example 1: Create a SimplePool with the default id of "simple.default" storage=ProductStorage()

```
storage.register(SimplePool.getInstance())
```

Example 2: Create a SimplePool with the id of "simple.mypool" storage=ProductStorage()

```
storage.register(SimplePool.getInstance("mypool"))
```

Example 3: Subsequent saving and loading to/from a SimplePool ref=storage=save(myproduct)

```
storage.load(ref)
```

API Summary

Methods
<p><code>SimplePool</code> getInstance(String id) Creates an instance of a SimplePool given a specified identifier</p>
<p><code>SimplePool</code> getInstance() Creates an instance of a SimplePool with the default ID ("default").</p>

API details

Methods

<code>SimplePool</code> getInstance (String id)
<p>Argument</p> <p><code>String</code> id [INPUT, MANDATORY, default=The identifier for the pool]</p>
<p>Return</p> <p><code>SimplePool</code></p>

<code>SimplePool getInstance(String id)</code>
--

An instance of the SimplePool.


<code>SimplePool getInstance()</code>

Return

<code>SimplePool</code>

An instance of the SimplePool.

3.285. 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](#)

3.286. 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
any type x [INPUT, MANDATORY, default=no default value]
any type y [OUTPUT, NOT_MANDATORY, default=no default value]

Miscellaneous

Does not work for complex values.

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

- [SIN](#)
- [ARCSIN](#)

3.287. 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

3.288. 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, OPTIONAL, default=no default value.]
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, OPTIONAL, default=no default value.]
Smoothing width parameter to configure the filter/convolution kernel. It is specified as a number of channels.
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".

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.

3.289. SOLVE

Full Name:	herschel.ia.numeric.toolbox.matrix.doc.SOLVE.entry.urm.xml
Alias:	SOLVE
Type:	Unknown (XML-based documentation) - 
Import:	from herschel.ia.numeric.toolbox.matrix.doc import SOLVE.entry.urm.xml

Description

Solves systems of linear equations.

This wrapper is deprecated. Please, use the `MatrixSolve` Java class.


Limitations

This is a Jython wrapper function for the numeric `MatrixSolve(b)(A)`.

See also

- [MatrixSolve](#)

3.290. 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.

3.291. SourceExtractorTask

Full Name:	herschel.ia.toolbox.srcext.SourceExtractorTask
Type:	Java Task - 
Import:	from herschel.ia.toolbox.srcext import SourceExtractorTask
Category:	task

Description

unknown

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]
String prf [INPUT, OPTIONAL, default=no default value]
Double pixelRegion [INPUT, MANDATORY, default=no default value]
Double corner1Ra [INPUT, OPTIONAL, default=default value: Ra of image pixel (0)]
Double corner1Dec [INPUT, OPTIONAL, default=default value: Dec of image pixel (0)]
Double corner2Ra [INPUT, OPTIONAL, default=default value: Ra of image pixel (width-1)]
Double corner2Dec [INPUT, OPTIONAL, default=default value: Dec of image pixel (width-1)]
String algorithm [INPUT, OPTIONAL, default=default value: daophot]
SourceListProduct result [OUTPUT, MANDATORY, default=no default value]

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 maxima is detected
Double fwhm [INPUT, MANDATORY, default=no default value]
Width in arcsecs of a gaussian default beam profile. For SPIRE, the values should be: PSW : 18.6 PMW : 24.0 PLW : 35.2
String prf [INPUT, OPTIONAL, default=no default value]
Filename of a FITS file that contains a custom point response function. If defined, this will be used instead of the default gaussian.

Double pixelRegion [INPUT, MANDATORY, default=no default value]

Radius around each pixel to consider when searching for local maxima.

Double corner1Ra [INPUT, OPTIONAL, default=default value: Ra of image pixel (0)]

Double corner1Dec [INPUT, OPTIONAL, default=default value: Dec of image pixel (0)]

Double corner2Ra [INPUT, OPTIONAL, default=default value: Ra of image pixel (width-1)]

Double corner2Dec [INPUT, OPTIONAL, default=default value: Dec of image pixel (width-1)]

These parameters contain two points on the image in world coordinates. These define opposite corners of a box. Only sources within this box 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.


String algorithm [INPUT, OPTIONAL, default=default value: daophot]

algorithm to use to extract sources. Currently only DAOPHOT is implemented; in future releases other algorithms (notably SussExtractor) will be available.

SourceListProduct result [OUTPUT, MANDATORY, default=no default value]

The resultant list of sources found on the input image.

3.292. 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.
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](#)}

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...]
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.

3.293. 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 = $1 / \text{stdev}^2$, 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.appendRow( Double1d( range(1000 ) ) + i )
```

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.

3.294. 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 creactomputation 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 rectance, associated with this
PlotXY getPlot () Returns the PlotXY, shown in this SpectrumPlotting.
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
setPlot () initiate the PlotXY, shown in this SpectrumPlotting.
boolean checkPointclicked () Returns a Flag
ArrayList getImageFigures () Returns all ImageFigures used for this SpectrumPlotting (the
DoubleId getSpectrum () Returns a DoubleId at this date containing the spectrum values
void saveData () Returns all ImageFigures used for this SpectrumPlotting (the

Methods

```
void exportCassis()
```

Returns all ImageFigures used for this SpectrumPlotting (the

API details

Constructor

```
SpectrumPlotting(boolean useAsComponent,  
CubeSpectrumAnalysisToolbox toolbox)
```

component-based, a window for plotting the intensity is opened; otherwise nothing will be shown.

Arguments

boolean **useAsComponent** [INPUT, MANDATORY]

CubeSpectrumAnalysisToolbox toolbox [INPUT, MANDATORY]

Methods

```
setClicked(Double center)
```

SpectrumPlotting to the given point (in UserCoordinates).

Argument

Double center [INPUT, MANDATORY]

```
PlotXY getPlot()
```

Return

PlotXY

The PlotXY, shown in this SpectrumPlotting.

```
ImageFigure getRect()
```

the intensity in this SpectrumPlotting.

Return

ImageFigure

The straight line (ImageFigure), of which we wish to plot the intensity in this SpectrumPlotting.

```
Double getClicked()
```

Return

Double

Returns the clicked position in PixelCoordinates.

```
int getClicks()
```

Return

int

Returns the number of valid clicks (i.e. in the image) you made.

String `getSkyCoordinates()`
(WorldCoordinates) of the point Clicked.
Return
String
Returns the String version of the sky coordinates (WorldCoordinates) of the point Clicked.

String `getPixelCoordinates()`
the rectangle which is also the point analysed.
Return
String
Returns the String version of the PixelCoordinates.

setPlot()

boolean `checkPointClicked()`
Return
boolean
returns a Boolean Flag which allow the extraction of the spectrum at the given position.

ArrayList `getImageFigures()`
rectangle on the image).
Return
ArrayList
Returns all ImageFigures used for this SpectrumPlotting (the rectangle on the image).

Double1d `getSpectrum()`
Return
Double1d
Returns a Double1d at this date containing the spectrum values


void `saveData()`
rectangle on the image).
Return
void
Returns all ImageFigures used for this SpectrumPlotting (the rectangle on the image).

void `exportCassis()`
rectangle on the image).
Return
void

void exportCassis()

Returns all ImageFigures used for this SpectrumPlotting (the rectangle on the image).

3.295. 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))
```

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.]
double[] percentiles [INPUT, OPTIONAL, default=no default value.]

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 Spectrum1d. 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. You have two alternatives to do that: Specify a 'Range'-object: Then this range object is considered for all the sub-segments. Specify an array of 'Range'-objects: Here, each Range object in the array is mapped to a sub-segments. This means that the length of the array should correspond to the number of sub-segments included in point spectra.


double[] percentiles [INPUT, OPTIONAL, default=no default value.]
--

Specify what percentiles should be given in the output (a list of probabilities).

History

- 2008-06-05 - meli: initial.

3.296. 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 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.

Three operation types can be distinguished:

- Many-to-one operations (e.g. average): Here, the input is a single spectrum container.
- Scalar operations such as adding a scalar to the spectra. The input is a single container **and** a double parameter. Here, the selection functionality is not used.
- Pair operations such as subtract or divide. Here, the input is a pair of spectrum containers and the operation is applied to the pairs of individual spectra included in the two containers.

API Summary

Properties
SpectrumContainer ds [INPUT, OPTIONAL, default=no default value.]
SpectrumContainer ds1 [INPUT, OPTIONAL, default=no default value.]
SpectrumContainer ds2 [INPUT, OPTIONAL, default=no default value.]
SelectionModel selection [INPUT, OPTIONAL, default=no default value.]
PyDictionary Map> selection_lookup [INPUT, OPTIONAL, default=no default value.]
PyList selection_index [INPUT, OPTIONAL, default=No default value.]
SegmentSelection segments [INPUT, OPTIONAL, default=no default value.]
SegmentSelection segments1 [INPUT, OPTIONAL, default=no default value.]
SegmentSelection segments2 [INPUT, OPTIONAL, default=no default value.]
Boolean overwrite [INPUT, OPTIONAL, default=False.]
String variant [INPUT, OPTIONAL, default=no default value.]
Double param [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 many-to-one operation (eg avg) or as scalar operation.
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.
SelectionModel selection [INPUT, OPTIONAL, default=no default value.]
Specify an instance of any kind of SelectionModel here.
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.
SegmentSelection segments [INPUT, OPTIONAL, default=no default value.]
Specify an instance of a SegmentSelection which gives the information on what segments for each point spectrum included in the container and, possibly, what ranges / masks should be considered for the processing.
SegmentSelection segments1 [INPUT, OPTIONAL, default=no default value.]
SegmentSelection to be associated with 'ds1'.
SegmentSelection segments2 [INPUT, OPTIONAL, default=no default value.]
SegmentSelection to be associated with 'ds2'.
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"

<code>Double param [INPUT, OPTIONAL, default=no default value.]</code>
--

The scalar parameter to be considered when the task is used as scalar operation.
--


<code>SpectrumContainer result [OUTPUT, OPTIONAL, default=no default value]</code>
--

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

History

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


3.297. SpireDataFrameFactoryImpl

Full Name:	herschel.vanilla.ccm.spire.SpireDataFrameFactoryImpl
Type:	Java Class - 
Import:	from herschel.vanilla.ccm.spire import SpireDataFrameFactoryImpl

History

- March 2004 : Original
- January 2005 : Use Number[] to set the data
- March 2005 : Implements dataframes with sequenceCount and packetTime
- December 2005: DataFrameDetails now adds package name to "classname"
- 8 May 2006 : Remove usage of deprecated DataFrame methods. Use generics
- 4 Oct 2006 : Add constructors using TmSource
- 20 Mar 2007 : Use getSourceName() instead of getModelName().
- 21 Jul 2008 : Remove useless methods. Fix method to copy spire data frames


3.298. SpireDataFrameFactoryImpl

Full Name:	herschel.versant.ccm.spire.SpireDataFrameFactoryImpl
Type:	Java Class - 
Import:	from herschel.versant.ccm.spire import SpireDataFrameFactoryImpl

History

- March 2004 : Original
- January 2005 : Use Number[] to set the data
- March 2005 : Implements dataframes with sequenceCount and packetTime
- December 2005: DataFrameDetails now adds package name to "classname"
- 8 May 2006 : Remove usage of deprecated DataFrame methods. Use generics
- 4 Oct 2006 : Add constructors using TmSource
- 20 Mar 2007 : Use getSourceName() instead of getModelName().
- 21 Jul 2008 : Remove useless methods. Fix method to copy spire data frames

3.299. 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
<pre>x=Int1d([0,4,9]) print SQRT(x) # [0,2.,3.]</pre>

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

3.300. 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

3.301. 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](#)

3.302. 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)

3.303. SubtractSpectrum

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

Description

Task for subtracting from the flux included in a spectrum container a scalar or for subtracting 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 pairwise mode, the task iterates synchronously 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.

In either mode, you can specify the segments you would like to process. See the 'segments' (for the scalar mode) or the 'segments1', 'segments2' parameters for the pair-wise mode.

For scalar mode, you have advanced selection functionality in place using the selection models as already discussed in the SpectrumTask.

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).

For further details see the (abstract) SpectrumTask in the same package.

Example

Example 1: from Jide:

```
from herschel.ia.toolbox.spectrum import SubtractSpectrum
subtract = SubtractSpectrum()
subtractConstant = subtract(ds=spectra, param=2.1)
subtractedPairWise = subtract(ds1=spectra1, ds2=spectra2)
```

API Summary

Properties
SpectrumContainer ds [INPUT, OPTIONAL, default=no default value.]
Double param [INPUT, OPTIONAL, default=no default value.]
SelectionModel selection [INPUT, OPTIONAL, default=no default value.]
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.]

Properties
SpectrumContainer ds1 [INPUT, OPTIONAL, default=no default value.]
SpectrumContainer ds2 [INPUT, OPTIONAL, default=no default value.]
SegmentSelection segments1 [INPUT, OPTIONAL, default=no default value.]
SegmentSelection 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 many-to-one operation (eg avg) or as scalar operation.
Double param [INPUT, OPTIONAL, default=no default value.]
The scalar parameter to be considered when the task is used as scalar operation.
SelectionModel selection [INPUT, OPTIONAL, default=no default value.]
Specify an instance of any kind of SelectionModel here.
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.
SegmentSelection segments1 [INPUT, OPTIONAL, default=no default value.]
SegmentSelection to be associated with 'ds1'.

<code>SegmentSelection segments2 [INPUT, OPTIONAL, default=no default value.]</code>
--

SegmentSelection 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

3.304. 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
<pre>x=Int2d([[1,2], [-1,3]]) print SUM(x) # 5 print SUM(x, 0) # [0,5] print SUM(x, 1) # [3,2]</pre>

API Summary

Jython Syntax
<code><y>=SUM(<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

- [PRODUCT](#)

3.305. 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
<pre>x=Float1d([0,0.5]) print TAN(x) # [0.0,0.5463025]</pre>

API Summary

Jython Syntax
<code><y>=TAN(<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>

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

- [ARCTAN](#)
- [COS](#)
- [SINH](#)
- [TANH](#)

3.306. 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](#)

3.307. 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 an 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.

3.308. ThreadDynamicProcessImpl

Full Name:	herschel.ia.dataflow.ThreadDynamicProcessImpl
Type:	Java Class - 
Import:	from herschel.ia.dataflow import ThreadDynamicProcessImpl

Description

Provides the thread-based model for implementing processes.

Class to be extended. It provides thread-based model. The only method that in general should be overridden is run method. The most interested methods are start, stop and run. The run method has always to be overridden, since it is where the execution of the process is done. As this class is handling a Thread, run method has to follow some rules. The typical run method should be like:

```
public void run() {
while( isRunning() ) {
try {
input = _input.getData();
output =doJob(input);
_output.dataReady(output);
} catch( InterruptedException ex ) {
break;
} catch( Exception ex2 ) {
handlingException(ex2);
}
}
}
```

The way a thread is stopped is simply letting it to return from the run method. isRunning() will stop the thread, but sometimes in the run method is stucked, waiting in a blocked method (as _input.getData() in the example above). So, the stop method not only makes that isRunning() will return false, but it will try to interrupt the thread just in case. That means that it is important to let run() method to be finished in case an InterruptedException is caught. If not, probably the thread will be eating all the CPU in an infinite loop. For a simpler thread-based super class, see herschel.ia.dataflow.template.ThreadDynamicProcessImplTemplate

IMPORTANT NOTE: This class provides two methods start/stop. These two methods are managing an internal thread. That means that when start is called, it will launch a thread, that may be stopped with stop. So, if you are using your own thread, redefine start/stop methods but do not call super.start()/super.stop() because two threads will be launched (unless this is really what you want to do).

Example

Example 1: how to implement a ThreadDynamicProcessImpl

```
<pre>
import herschel.ia.dataflow.*;
public class ProcessFFT extends ThreadDynamicProcessImpl {
    private final ProcessInput _input;
    private final ProcessOutput _output;
    public ProcessFFT(String name) {
        super(name);
        _input = createInput("input", Product.class);
        _output = createOutput("output", Product.class);
    }
    public void run() {
        while( isRunning() ) {
            try {
                Product product = (Product)_input.nextData();
            }
        }
    }
}
```

Example 1: how to implement a ThreadDynamicProcessImpl

```
        // a doStuff method should be defined and would do the
processing.
        Product new_product = doStuff(product);
        _output.dataReady( new_product );
    } catch ( InterruptedException ex ) {
        System.out.println("Interrupted exception thrown:" + ex + ").
Exiting thread");
        break;
    }
} // while
}
}
</pre>
```

Limitations

no limitation


See also

- [reference](#)

History

- 26-5-2005 jcg: change javadoc format for help support.

3.309. 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

The TiledImage task for Image.

TiledImageTask is a task which returns the TiledImage of an Image.

- flag : The tiledImage will take into account the flag of the image.

Examples

Example 1: Return the TiledImage, taking into account the flag.

```
tiledImageTask(image = im, flag = True)
```

Example 2: Return the TiledImage, not taking into account the flag.

```
tiledImageTask(image = im, method=CutLevels.MEDIAN_FILTER)
```

API Summary

Jython Syntax

```
TiledImageTask(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

3.310. 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.

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.

3.311. translateTask

Full Name:	herschel.ia.toolbox.image.TranslateTask
Alias:	translateTask
Type:	Java Task - 
Import:	from herschel.ia.toolbox.image import TranslateTask
Category:	task/image

Description

The Translate task for Images.

Translate is 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

```
translateTask(image = im, x = 5, y = 7)
```

Example 2: translating an image using sky coordinates

```
translateTask(image = im, ra = 0.03, dec = 0.03)
```

API Summary

Jython Syntax

```
translateTask()
translateTask(image, ra, dec)
translateTask(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

See also

- [???](#)

3.312. 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
<pre>x=Int2d([[1,2],[3,4],[5,6]]) print TRANSPOSE(x) # [[1,3,5],[2,4,6]]</pre>

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.

3.313. transposeTask

Full Name:	herschel.ia.toolbox.image.TransposeTask
Alias:	transposeTask
Type:	Java Task - 
Import:	from herschel.ia.toolbox.image import TransposeTask
Category:	task/image

Description

The Transpose task for Images.

TransposeTask is 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

```
transposeTask(image = im, type = TransposeTask.FLIP_HORIZONTAL)
```

Example 2: Flipping an image antidiagonal

```
transposeTask(im, TransposeTask.FLIP_ANTIDIAGONAL)
```

API Summary

Jython Syntax

```
transposeTask()  
transposeTask(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 Image to transpose
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]
Result of the transposition of the Image

3.314. 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.

3.315. 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 Double1d

```
x=Double1d([-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=Double1d([-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=Double1d([4, 5, 7, 4, 8, 8, 3, 1, 5])
print UNIQ.COUNT_UNIQ_VALUES(x) # 6
```

Example 4: Apply UNIQ.IS_UNIQ

```
x=Double1d([-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]

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

API details

Properties


any sorted/unsorted 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.
--

3.316. 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](#)

3.317. 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
<code>simpleCube simplecube [INPUT, MANDATORY, default=no default value]</code>
<code>axis unknown [INPUT, Boolean, default=MANDATORY]</code>
<code>coordSlitArray unknown [INPUT, Double2d, default=no default value]</code>
<code>widthSlit unknown [INPUT, Integer, default=no default value]</code>
<code>referenceLayer unknown [INPUT, Integer, default=no default value]</code>
<code>totalWeight unknown [INPUT, no default value, default=no default value]</code>
<code>velocityMap unknown [OUTPUT, Double3d, default=no default value]</code>
<code>velocityMapAxis unknown [OUTPUT, Double2d, default=no default value]</code>
<code>velocityMapAxisProd unknown [OUTPUT, SimpleImage, default=no default value]</code>
<code>cubeVelocityMap unknown [OUTPUT, SimpleCube, default=no default value]</code>


API details

Properties

<code>simpleCube simplecube [INPUT, MANDATORY, default=no default value]</code>
The spectralCube as a SimpleCube containing the spectral and velocities information
<code>axis unknown [INPUT, Boolean, default=MANDATORY]</code>
Define the type of map to generate axis coordSlitArray widthSlit referenceLayer totalWeight
<code>coordSlitArray unknown [INPUT, Double2d, default=no default value]</code>
new version array of pixel to be read : manage the width of the slit the order of the information is index, X, Y , weight
<code>widthSlit unknown [INPUT, Integer, default=no default value]</code>
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 ...

3.318. VelocityPosMapPlotting

Full Name:	herschel.ia.gui.cube.VelocityPosMapPlotting
Type:	Java Class - 
Import:	from herschel.ia.gui.cube import VelocityPosMapPlotting

Description

VelocityPosMapPlotting

A class to deal with VelocityPosMapComputeTask.

API Summary

Constructors
<code>VelocityPosMapPlotting(type toolbox)</code>
<code>VelocityPosMapPlotting(boolean useAsComponent, CubeSpectrumAnalysisToolbox toolbox)</code> Constructor for VelocityPosMapPlotting.
Methods
<code>setBegin(Double begin)</code> Sets the begin of the straight line, associated with this
<code>setEnd(Double end)</code> Sets the end of the straight line, associated with this
<code>ImageFigure getLine()</code> Returns the straight line (ImageFigure), of which we wish to plot
<code>Double getBegin()</code> Returns the begin of the drawn line in PixelCoordinates.
<code>Double getEnd()</code> Returns the end of the drawn line in PixelCoordinates.
<code>DoubleId IntensityVelocityPosMapPlotting()</code> Returns the intensity of the pixels along the straight line,
<code>Point2D[] getPlotPoints()</code> Returns the intensity of the pixels along the straight line,
<code>void getCoordPoints()</code> Returns the intensity of the pixels along the straight line,
<code>String getSkyCoordinates()</code> Returns the String version of the sky coordinates

API details

Constructors

<code>VelocityPosMapPlotting(type toolbox)</code>
Argument

VelocityPosMapPlotting (type toolbox)
type toolbox [INPUT, MANDATORY, default=no default value] The CubeSpectrumAnalysisToolbox, for which you want to extract a velocity position map

VelocityPosMapPlotting (boolean useAsComponent, CubeSpectrumAnalysisToolbox toolbox)
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)
Velocity position map to the given point (in UserCoordinates).
Argument
Double begin [INPUT, MANDATORY]

setEnd (Double end)
Velocity position map to the given point (in UserCoordinates).
Argument
Double end [INPUT, MANDATORY]

ImageFigure getLine()
the intensity in this VelocityPosMapPlotting.
Return
ImageFigure
The straight line (ImageFigure), of which we wish to plot the intensity in this VelocityPosMapPlotting.

Double getBegin()
Return
Double
Returns the begin of the drawn line in PixelCoordinates.

Double getEnd()
Return
Double
Returns the end of the drawn line in PixelCoordinates.

DoubleId IntensityVelocityPosMapPlotting()
associated with the given Velocity position map.

DoubleId IntensityVelocityPosMapPlotting()**Return****DoubleId**

Returns the intensity of the pixels along the drawn straight line, associated with the given Velocity position map.

Point2D[] getPlotPoints()

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.

void getCoordPoints()

associated with the given Velocity position map.

Return**void**

Returns the intensity of the pixels along the drawn straight line, associated with the given Velocity position map.


String getSkyCoordinates()

(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.

3.319. 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	
<code>Wcs()</code>	The default constructor.
<code>Wcs(int naxis)</code>	Constructor for 2 or 3 dimensional Wcs.
<code>Wcs(Wcs orig)</code>	The copy constructor.
Methods	
<code>Wcs copy()</code>	Returns a copy of the Wcs object.
<code>setParameter(String paramname, Object param, String description)</code>	Adds a new parameter to the Wcs.
<code>isCompleteWcs()</code>	Checks whether the Wcs is complete.
<code>Object getParameter(String paramname)</code>	Returns a parameter from the Wcs.
<code>Object[] getParameters()</code>	Returns all parameters.
<code>removeParameter(String paramname)</code>	Removes a parameter.
<code>setNAxis(int naxis)</code>	Sets the number of axes.
<code>int getNAxis()</code>	Returns the number of axes.

Methods
<code>setImageIndex(DoubleId index, Unit unit)</code> Sets the image index for a non-equidistant 3rd dimension.
<code>boolean hasImageIndex()</code> Checks the image index
<code>TableDataset getImageIndex()</code> Returns the image index.
<code>boolean isEquidistantInZ()</code> Checks whether the Wcs is equidistant in the 3rd dimension.
<code>setCrval1(double crval1)</code> Sets crval1.
<code>double getCrval1()</code> Returns crval1
<code>setCrval2(double crval2)</code> Sets crval2.
<code>double getCrval2()</code> Returns crval2
<code>setCrval3(double crval3)</code> Sets crval3.
<code>double getCrval3()</code> Returns crval3
<code>setCrpix1(double crpix1)</code> Sets crpix1.
<code>double getCrpix1()</code> Returns crpix1
<code>setCrpix2(double crpix2)</code> Sets crpix2.
<code>double getCrpix2()</code> Returns crpix2
<code>setCrpix3(double crpix3)</code> Sets crpix3.
<code>double getCrpix3()</code> Returns crpix3
<code>setCdelt1(double cdelt1)</code> Sets cdelt1.
<code>double getCdelt1()</code> Returns cdelt1
<code>setCdelt2(double cdelt2)</code> Sets cdelt2.
<code>double getCdelt2()</code> Returns cdelt2
<code>setCdelt3(double cdelt3)</code>

Methods
Sets cdelt3.
double <code>getCdelt3()</code> Returns cdelt3
boolean <code>checkCtypeValidity(String ctype)</code> Checks the validity of the ctype parameter.
<code>setCtype1(String ctype1)</code> Sets ctype1.
String <code>getCtype1()</code> Returns ctype1
<code>setCtype2(String ctype2)</code> Sets ctype2.
String <code>getCtype2()</code> Returns ctype2
<code>setCtype3(String ctype3)</code> Sets ctype3.
String <code>getCtype3()</code> Returns ctype3
<code>setCunit1(String cunit1)</code> Sets cunit1.
String <code>getCunit1()</code> Returns cunit1
<code>setCunit2(String cunit2)</code> Sets cunit2.
String <code>getCunit2()</code> Returns cunit2
boolean <code>hasParameter(String parameter)</code> Checks the availability of a parameter.
<code>setCunit3(String cunit3)</code> Sets cunit3.
String <code>getCunit3()</code> Returns cunit3
<code>setEpoch(double epoch)</code> Sets the epoch.
double <code>getEpoch()</code> Returns the epoch
<code>setEquinox(double equinox)</code> Sets the equinox.
double <code>getEquinox()</code> Returns the equinox
<code>setRadesys(String Radesys)</code> Sets the reference frame.

Methods
String <code>getRadesys()</code> Returns the reference frame
<code>setCd1_1(double cd1_1)</code> Sets the cd1_1 element.
<code>setCd1_2(double cd1_2)</code> Sets the cd1_2 element.
<code>setCd1_3(double cd1_3)</code> Sets the cd1_3 element.
<code>setCd2_1(double cd2_1)</code> Sets the cd2_1 element.
<code>setCd2_2(double cd2_2)</code> Sets the cd2_2 element.
<code>setCd2_3(double cd2_3)</code> Sets the cd2_3 element.
<code>setCd3_1(double cd3_1)</code> Sets the cd3_1 element.
<code>setCd3_2(double cd3_2)</code> Sets the cd3_2 element.
<code>setCd3_3(double cd3_3)</code> Sets the cd3_3 element.
double <code>getCd1_1()</code> Returns cd1_1
double <code>getCd1_2()</code> Returns cd1_2
double <code>getCd2_1()</code> Returns cd2_1
double <code>getCd2_2()</code> Returns cd2_2
double <code>getCd1_3()</code> Returns cd1_3
double <code>getCd2_3()</code> Returns cd2_3
double <code>getCd3_1()</code> Returns cd3_1
double <code>getCd3_2()</code> Returns cd3_2
double <code>getCd3_3()</code> Returns cd3_3
<code>setPc1_1(double pc1_1)</code> Sets the pc1_1 element.
double <code>getPc1_1()</code>

Methods
Returns pc1_1
<code>setPc1_2(double pc1_2)</code> Sets the pc1_2 element.
<code>double getPc1_2()</code> Returns pc1_2
<code>setPc1_3(double pc1_3)</code> Sets the pc1_3 element.
<code>double getPc1_3()</code> Returns pc1_3
<code>setPc2_1(double pc2_1)</code> Sets the pc2_1 element.
<code>double getPc2_1()</code> Returns pc2_1
<code>setPc2_2(double pc2_2)</code> Sets the pc2_2 element.
<code>double getPc2_2()</code> Returns pc2_2
<code>setPc2_3(double pc2_3)</code> Sets the pc2_3 element.
<code>double getPc2_3()</code> Returns pc2_3
<code>setPc3_1(double pc3_1)</code> Sets the pc3_1 element.
<code>double getPc3_1()</code> Returns pc3_1
<code>setPc3_2(double pc3_2)</code> Sets the pc3_2 element.
<code>double getPc3_2()</code> Returns pc3_2
<code>setPc3_3(double pc3_3)</code> Sets the pc3_3 element.
<code>double getPc3_3()</code> Returns pc3_3
<code>setProjection(String projection)</code> Sets the projection.
<code>String getProjection()</code> Returns the projection.
<code>MetaData getMeta()</code> Return the Wcs as metadata.
<code>String toString()</code> Returns a string representation of the Wcs.

Methods
double[] getWorldCoordinates(double row, double column) Returns the world coordinates of the given image coordinates.
double getZCoordinate(int depth) Returns the world coordinates of the given layer.
double[] getPixelCoordinates(double c1, double c2) Returns the pixel coordinates.
setCrota2(double crota2) Sets crota2
double getCrota2() Returns crota2.
boolean isValid() Checks the possibility to convert from pixel to world coordinates.

API details

Constructors

Wcs ()
A constructor which creates a standard Wcs object. The standard Wcs sets naxis = 2 (NAXIS = 2)
Example
Typical example on how to create a Wcs.
<pre>wcs = Wcs(crval1=30.0, crval2=-89.50, crpix1=109, crpix2=109)</pre>

Wcs(int naxis)
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.
<pre>wcs = Wcs(3) wcs.setCrval1(30.0)</pre>

Wcs(Wcs orig)
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.
Return

Wcs copy()
Wcs A copy of the Wcs.
setParameter(String paramname, Object param, String description)
Adds a new parameter to the Wcs. Arguments String paramname [INPUT, MANDATORY] Object param [INPUT, MANDATORY] String description [INPUT, MANDATORY]
isCompleteWcs()
Returns true if the Wcs has enough information to convert to and from World Coordinates.
Object getParameter(String paramname)
Returns the requested parameter of the Wcs. Argument String paramname [INPUT, MANDATORY] Return Object The requested parameter.
Object[] getParameters()
Returns the list of all Wcs parameters. Return Object[] The list of all Wcs parameters.
removeParameter(String paramname)
Removes the requested parameter of the Wcs. Argument String paramname [INPUT, MANDATORY]
setNAxis(int naxis)
Sets the number of axes of the Wcs. Argument int naxis [INPUT, MANDATORY]
int getNAxis()
Returns the number of axes of the Wcs. Return int The number of axes.

```
setImageIndex(DoubleId index, Unit unit)
```

Sets the image index for a non-equidistant 3rd dimension.

Arguments

`DoubleId index` [INPUT, MANDATORY]

`Unit unit` [INPUT, MANDATORY]

```
boolean hasImageIndex()
```

Returns true if there is an image index.

Return

boolean

True if there is an image index.

```
TableDataset getImageIndex()
```

Returns the image index for a non-equidistant 3rd dimension.

Return

TableDataset

The image index for a non-equidistant 3rd dimension.

```
boolean isEquidistantInZ()
```

`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 the first coordinate of the center.

Argument

`double crval1` [INPUT, MANDATORY]

```
double getCrval1()
```

Returns the first coordinate of the reference pixel.

Return

double

The first coordinate of the reference pixel.

```
setCrval2(double crval2)
```

Sets the second coordinate of the center.

Argument

`double crval2` [INPUT, MANDATORY]

```
double getCrval2()
```

Returns the second coordinate of the reference pixel.

```
double getCrval2()
```

Return

```
double
```

The second coordinate of the reference pixel.

```
setCrval3(double crval3)
```

Sets the third coordinate of the center.

Argument

```
double crval3 [INPUT, MANDATORY]
```

```
double getCrval3()
```

Returns the third coordinate of the reference pixel.

Return

```
double
```

The third coordinate of the reference pixel.

```
setCrpix1(double crpix1)
```

Sets the reference pixel position of axis 1.

Argument

```
double crpix1 [INPUT, MANDATORY]
```

```
double getCrpix1()
```

Returns the reference pixel position of axis 1.

Return

```
double
```

The reference pixel position of axis 1.

```
setCrpix2(double crpix2)
```

Sets the reference pixel position of axis 2.

Argument

```
double crpix2 [INPUT, MANDATORY]
```

```
double getCrpix2()
```

Returns the reference pixel position of axis 2.

Return

```
double
```

The reference pixel position of axis 2.

```
setCrpix3(double crpix3)
```

Sets the reference pixel position of axis 3.

Argument

```
double crpix3 [INPUT, MANDATORY]
```

double getCrpix3()
Returns the reference pixel position of axis 3.
Return
double
The reference pixel position of axis 3.
setCdelt1(double cdelt1)
Sets the pixel scale of axis 1.
Argument
double cdelt1 [INPUT, MANDATORY]
double getCdelt1()
Returns the pixel scale of axis 1.
Return
double
The pixel scale of axis 1.
setCdelt2(double cdelt2)
Sets the pixel scale of axis 2.
Argument
double cdelt2 [INPUT, MANDATORY]
double getCdelt2()
Returns the pixel scale of axis 2.
Return
double
The pixel scale of axis 2.
setCdelt3(double cdelt3)
Sets the pixel scale of axis 3.
Argument
double cdelt3 [INPUT, MANDATORY]
double getCdelt3()
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.
Argument

```
boolean checkCtypeValidity(String ctype)
```

```
    String ctype [INPUT, MANDATORY]
```

Return

```
    boolean
```

```
    true if the ctype parameter is valid.
```

```
setCtype1(String ctype1)
```

Sets the projection type of axis 1.

Argument

```
    String ctype1 [INPUT, MANDATORY]
```

```
String getCtype1()
```

Returns the projection type of axis 1.

Return

```
    String
```

```
    The projection type of axis 1.
```

```
setCtype2(String ctype2)
```

Sets the projection type of axis 2.

Argument

```
    String ctype2 [INPUT, MANDATORY]
```

```
String getCtype2()
```

Returns the projection type of axis 2.

Return

```
    String
```

```
    The projection type of axis 2.
```

```
setCtype3(String ctype3)
```

Sets the projection type of axis 3.

Argument

```
    String ctype3 [INPUT, MANDATORY]
```

```
String getCtype3()
```

Returns the projection type of axis 3.

Return

```
    String
```

```
    The projection type of axis 3.
```

```
setCunit1(String cunit1)
```

Sets the unit of axis 1.

Argument

```
    String cunit1 [INPUT, MANDATORY]
```

String getCunit1()

Returns the unit of axis 1.

Return**String**

The unit of axis 1.

setCunit2(String cunit2)

Sets the unit of axis 2.

Argument**String cunit2** [INPUT, MANDATORY]**String** getCunit2()

Returns the unit of axis 2.

Return**String**

The unit of axis 2.

boolean hasParameter(**String** parameter)

Checks whether the parameter is available.

Argument**String parameter** [INPUT, MANDATORY]**Return****boolean**

True if the parameter is available.

setCunit3(String cunit3)

Sets the unit of axis 3.

Argument**String cunit3** [INPUT, MANDATORY]**String** getCunit3()

Returns the unit of axis 3.

Return**String**

The unit of axis 3.

setEpoch(double epoch)

Sets the epoch.

Argument**double epoch** [INPUT, MANDATORY]**double** getEpoch()

Returns the epoch.

double getEpoch()
Return double The epoch.
setEquinox(double equinox)
Sets the equinox. Argument double equinox [INPUT, MANDATORY]
double getEquinox()
Returns the equinox. Return double The equinox.
setRadesys(String Radesys)
Sets the reference frame. Argument String Radesys [INPUT, MANDATORY]
String getRadesys()
Returns the reference frame. Return String The reference frame.
setCd1_1(double cd1_1)
Sets element (1, 1) of the corrected CD matrix CDi_j.. Argument double cd1_1 [INPUT, MANDATORY]
setCd1_2(double cd1_2)
Sets element (1, 2) of the corrected CD matrix CDi_j.. Argument double cd1_2 [INPUT, MANDATORY]
setCd1_3(double cd1_3)
Sets element (1, 3) of the corrected CD matrix CDi_j.. Argument double cd1_3 [INPUT, MANDATORY]
setCd2_1(double cd2_1)
Sets element (2, 1) of the corrected CD matrix CDi_j..

setCd2_1(double cd2_1)
Argument double cd2_1 [INPUT, MANDATORY]
setCd2_2(double cd2_2)
Sets element (2, 2) of the corrected CD matrix $CDi_{j..}$. Argument double cd2_2 [INPUT, MANDATORY]
setCd2_3(double cd2_3)
Sets element (2, 3) of the corrected CD matrix $CDi_{j..}$. Argument double cd2_3 [INPUT, MANDATORY]
setCd3_1(double cd3_1)
Sets element (3, 1) of the corrected CD matrix $CDi_{j..}$. Argument double cd3_1 [INPUT, MANDATORY]
setCd3_2(double cd3_2)
Sets element (3, 2) of the corrected CD matrix $CDi_{j..}$. Argument double cd3_2 [INPUT, MANDATORY]
setCd3_3(double cd3_3)
Sets element (3, 3) of the corrected CD matrix $CDi_{j..}$. Argument double cd3_3 [INPUT, MANDATORY]
double getCd1_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 element (1,2) of the corrected CD matrix $CDi_{j..}$. Return double Element (1,2) of the corrected CD matrix $CDi_{j..}$.
double getCd2_1()
Returns element (2,1) of the corrected CD matrix $CDi_{j..}$. Return double

double getCd2_1()

Element (2,1) of the corrected CD matrix CDi_j.

double getCd2_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 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 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 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 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 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 element (1, 1) of the linear transformation matrix PCi_j..

setPc1_1(double pc1_1)**Argument**double **pc1_1** [INPUT, MANDATORY]**double getPc1_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 element (1, 2) of the linear transformation matrix PCi_j..

Argumentdouble **pc1_2** [INPUT, MANDATORY]**double getPc1_2()**

Returns element (1,2) of the linear transformation matrix PCi_j.

Return**double**

Element (1,2) of the linear transformation matrix PCi_j.

setPc1_3(double pc1_3)

Sets element (1, 3) of the linear transformation matrix PCi_j..

Argumentdouble **pc1_3** [INPUT, MANDATORY]**double getPc1_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 element (2, 1) of the linear transformation matrix PCi_j..

Argumentdouble **pc2_1** [INPUT, MANDATORY]**double getPc2_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 element (2, 2) of the linear transformation matrix PCi_j..
Argument
double pc2_2 [INPUT, MANDATORY]
double getPc2_2()
Returns element (2,2) of the linear transformation matrix PCi_j.
Return
double
Element (2,2) of the linear transformation matrix PCi_j.
setPc2_3(double pc2_3)
Sets element (2, 3) of the linear transformation matrix PCi_j..
Argument
double pc2_3 [INPUT, MANDATORY]
double getPc2_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 element (3, 1) of the linear transformation matrix PCi_j..
Argument
double pc3_1 [INPUT, MANDATORY]
double getPc3_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 element (3, 2) of the linear transformation matrix PCi_j..
Argument
double pc3_2 [INPUT, MANDATORY]
double getPc3_2()
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 element (3, 3) of the linear transformation matrix PCi_j.
Argument
double pc3_3 [INPUT, MANDATORY]
double getPc3_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 type.
Argument
String projection [INPUT, MANDATORY]
String getProjection()
Returns the projection type.
Return
String
The projection type.
MetaData getMeta()
Returns the Wcs as metadata.
Return
MetaData
The Wcs as metadata.
String toString()
Returns a string representation of the values of the Wcs object. The format of this string is undefined and subject to change.
Return
String
a string representation of the values of the Wcs object.
double[] getWorldCoordinates(double row, double column)
Returns the world coordinates when the image coordinates are given.
Arguments
double row [INPUT, MANDATORY]
double column [INPUT, MANDATORY]
Return
double[]

double[] getWorldCoordinates(double row, double column)

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.

double getZCoordinate(int depth)

Returns the world coordinates of the given layer.

Argument

int **depth** [INPUT, MANDATORY]

Return

double

The corresponding world coordinates of the Z axis.

double[] getPixelCoordinates(double c1, double c2)

Returns the pixelCoordinates of the given SkyCoordinates. The pixel- coordinates are the row and the column of the image!

Arguments

double **c1** [INPUT, MANDATORY]
 double **c2** [INPUT, MANDATORY]

Return

double[]

A 2-dimensional array of doubles describing the pixel coordinates of the given world coordinates.

setCrota2(double crota2)

Sets the rotation angle in degrees.

Argument

double **crota2** [INPUT, MANDATORY]

double getCrota2()

Returns the rotation angle in degrees.

Return

double

The rotation angle in degrees.

boolean isValid()


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.

3.320. 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'th 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) )
The uncertainty of the weighted mean is then:
ewmean = sqrt( sum( (x[i]-wmean)^2/(dx[i]^2) ) / 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

```
values = Double1d([1,3,5])
errors = Double1d([0.1,0.2,0.3])
wmean = WeightedMean(values,errors)
print wmean.mean()
print wmean.error()
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.error()
</pre>
```

API Summary

Jython Syntax

```
wm = WeightedMean(<values>,<errors>,<ignoreNaN>)
wm.mean()
wm.error()
```

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

<code>number array values [INPUT, MANDATORY, default=no default value]</code>

The input numbers array can be integer, long, float, double and complex one dimensional array.
--

<code>number array errors [INPUT, MANDATORY, default=no default value]</code>

The input uncertainties array can be integer, long, float, double and complex one dimensional array.
--

<code>boolean ignoreNaN [INPUT, OPTIONAL, default=false]</code>

If 'true' a NaN value will be ignored. By default is 'false'.

See also

- [WeightedMean](#)