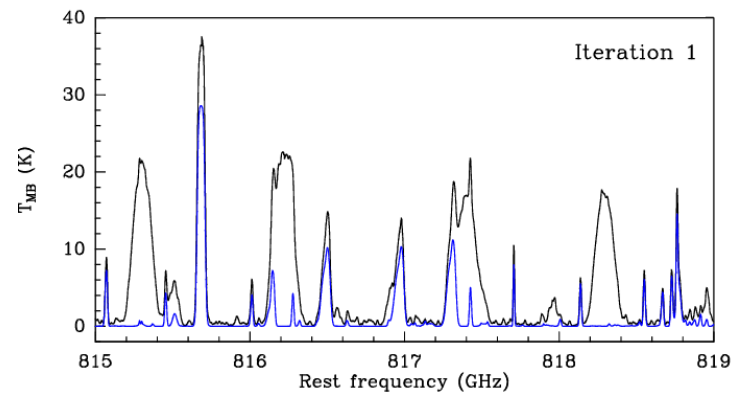




HCSS HIFI

Deconvolution Tool Demo

Steve Lord, NHSC



Steve Lord lord@ipac.caltech.edu



DP Workshop
ESAC (4,5 Dec 2008)



Deconvolution Tool Demo

Intro - The Problem



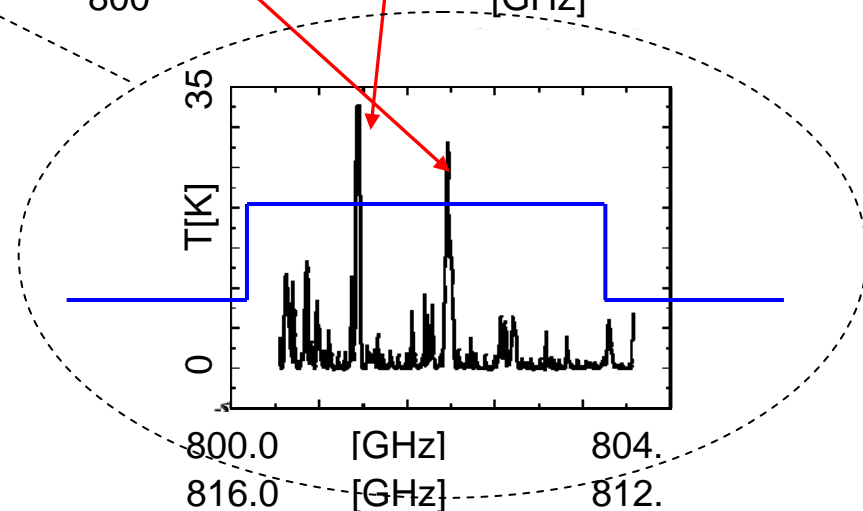
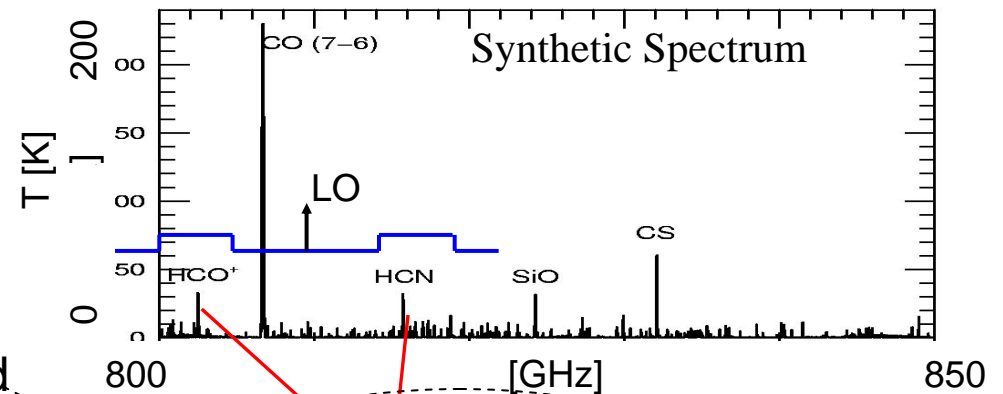
HIFI Deconvolution tool:

deconvolves HIFI's native dual sideband observations; a crucial processing need for spectral surveys.

Every spectrum taken with HIFI contains the upper and lower sideband data folded together.

In regions rich with lines, the user needs a way to sort this out.

Additionally, the upper and lower sidebands may have different gains.



Double sideband spectrum





Deconvolution Demo - The Solution

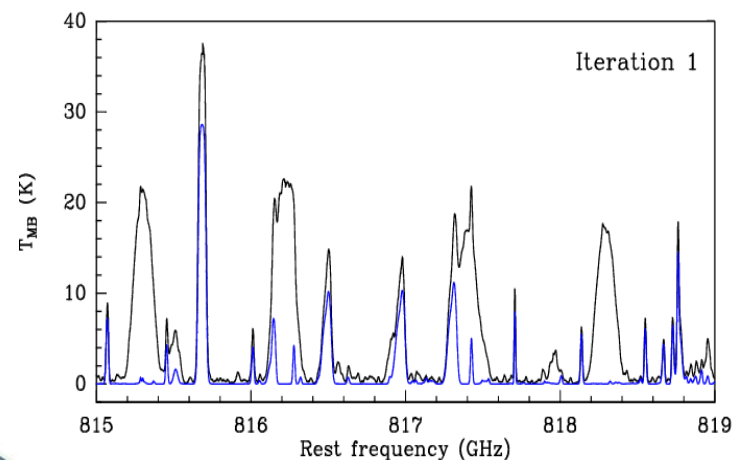
With a series of LO settings, the double sidebands are marched across the band, providing a sufficient number of **overlapping “confused” observed double sideband spectra to constrain the single sideband result** - I.e., the source’s true spectrum

An **iterative chi-square minimization** method (the Conjugate Gradient Method) finds the best single sideband solution.

CGM minimizes chi-square of the DSB observed & modelled residuals **using knowledge of the chi-square gradients.**

Successfully used on CSO surveys, gas cell data, & HIFI Data Simulator products.

Before and After Deconvolution:





Deconvolution Demo Considerations



The algorithm solves the LSB & USB sideband channel values and also the gains at each LO setting in a 2-step process: 1) fit SSB solution, 2) fit gains and SSB solution together.

Implementation in XCLASS was the prototype (Schilke and Comito) & subsequent improvements by Delforge and Comito.

Implementation was done in pure Fortran (Lord) and HCSS JAVA (Xie and Lord) with a Jython pre-processor added by Borys

Pre-processor removes spurs, and resamples the data to a grid.

Currently the HCSS and XCLASS versions are tested successfully on HIFI ILT gas cell data, & HIFI Data Simulator products.





Deconvolution - Status and Improvement Details



Items Complete:

- Conjugate Gradient Method available in HCSS for all users
- HCSS JAVA HIFI Deconvolution Package – up and running
- Interfaced and End-to-End tested with HCSS HIFI Simulator
- Preprocessing HTPs into JAVA code (Resampling, Upper, Lower Freqs)
- Integrated w/HIPE – run with scripts or GUI – viewing with spectrum explorer

Additional features In Development:

- Include Frequency Switching Spectral Survey Deconvolutions
- Channel-by-channel weights (from Cal) included in the JAVA Software
- Improved HIPE GUI with DSB Display Capabilities, DSP products, RMS vs. iteration number, SSB viewing
- Ghost detection and ghost prevention tools





Sideband Deconvolution HCSS Decon & XCLASS Decon



**Mostly in synch - but some
development differences**

Comparison of HCSS and XCLASS Development

| | HCSS/HIFI Deconvolution | Comment | XCLASS Deconvolution | Comment |
|----------------------|----------------------------|-------------------|-------------------------|-----------------|
| Freq. Switch | In progress | Coded, not tested | Implemented | Tested |
| Channel Weights | In progress | Coded, not tested | Unimplemented | Future work |
| Polynomial Gain Fits | On Hold | | Prototype | Not yet Working |





NHSC Review Pasadena, CA

1 Feb 2007

Demo Results



- Line results accurate in sideband placement and frequency
- Gain results `_very_` good...
- Line amplitudes undershoot currently due to coupling factors present in simulator but not pipeline: `eta_mb`





Screen shot of Decon/HIPE



The screenshot displays the HIPE environment with the following components:

- Editor:** Contains Python code for deconvolution. Key lines include:

```
decon_result = DoDeconvolution()(htp)

# extract the data from the result product
ssb = decon_result["ssb"]
gains = decon_result["gain"]

# plot the deconvolved output over the input template
from herschel.ia.toolbox.util import AsciiTableReaderTask
atrt = AsciiTableReaderTask()
template = atrt(file="/Users/lord/hcss/scripts/decon_demo...")

resultLayer = LayerXY(ssb.getWave(), ssb.getFlux(), name="Decon result")
templateLayer = LayerXY(template["freq"].data, template["flux"].data/2, name="Input template")
p = PlotXY(layers=[resultLayer, templateLayer])
p.getLegend().setVisible(1)
p.title.text = "Deconvolution output"
p.xaxis.title.text = "Frequency [MHz]"
p.yaxis.title.text = "Flux"
```
- Console:** Shows the execution of the code:

```
IA> template =
atrt(file="/Users/lord/hcss/scripts/decon_demo/band2b.data")
IA> resultLayer
=LayerXY(ssb.getWave(), ssb.getFlux(), name='Decon result')
IA>
templateLayer=LayerXY(template["freq"].data, template["flux"].data/2, name="Input template")
IA> p=PlotXY(layers=[resultLayer, templateLayer])
IA> p.getLegend().setVisible(1)
IA> p.title.text="Deconvolution output"
IA> p.xaxis.title.text="Frequency [MHz]"
IA> p.yaxis.title.text="Flux"
IA>
```
- Variables:** Lists various objects and constants such as `chsu02`, `chwidth`, `cut`, `database_property`, `DAYS`, `DECIBELS`, `decon_result`, `ELECTRON_CHARGE`, `gains`, `GIGAHERTZ`, `H_PLANCK`, `H_2`, `HERTZ`, `HOURS`, `hoverk`, `hplank`, `hrs_chwidth`, `htp`, `k`, `K_BULLI_ZMANN`, `KILOHERTZ`, `KILOMETERS`, `MEGAHERTZ`, `METERS`, `MICROMETERS`, `MICROSECONDS`, `MILLIMETERS`, `MILLISECONDS`, `MINUTES`, `ONE`, `p`, `PERCENT`, `resultLayer`, `SECONDS`, `shotnoise`, `Sorting`, `SPEED_OF_LIGHT`, `ssb`, `subbands`, `sys`, `template`, `templateLayer`, `TERAHERTZ`, and `zbd`.
- Tasks:** Shows a list of tasks including "All", "Applicable", and "By Category".





Results

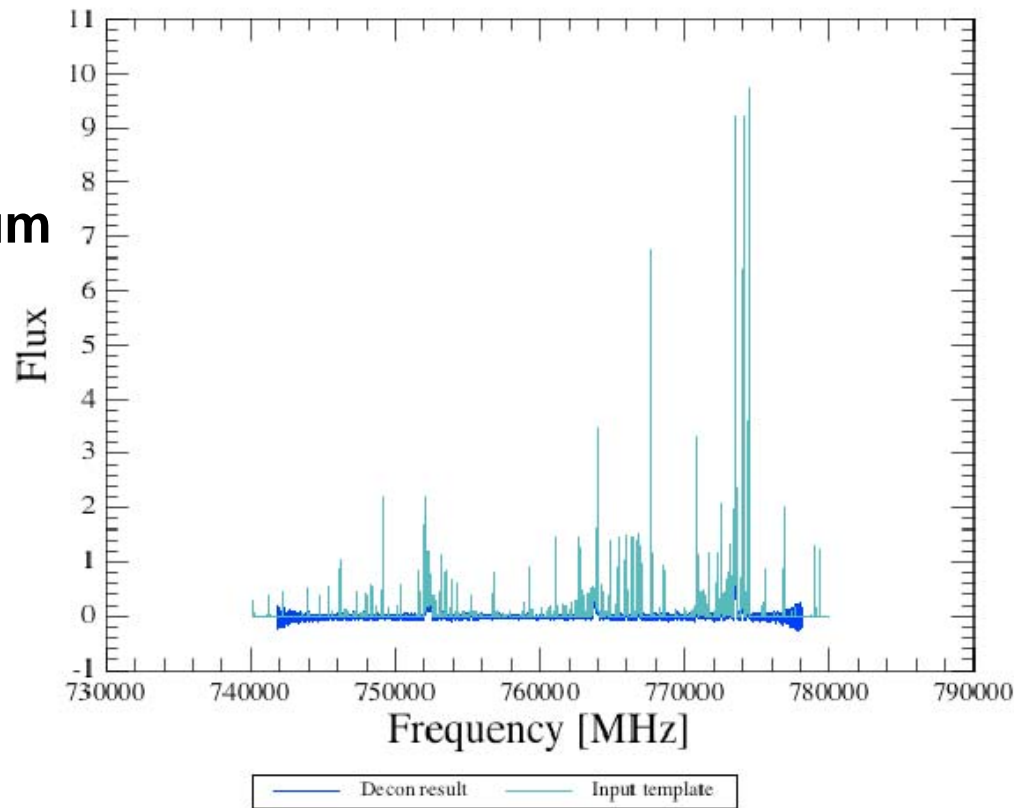


Herschel PlotXY

Deconvolution output

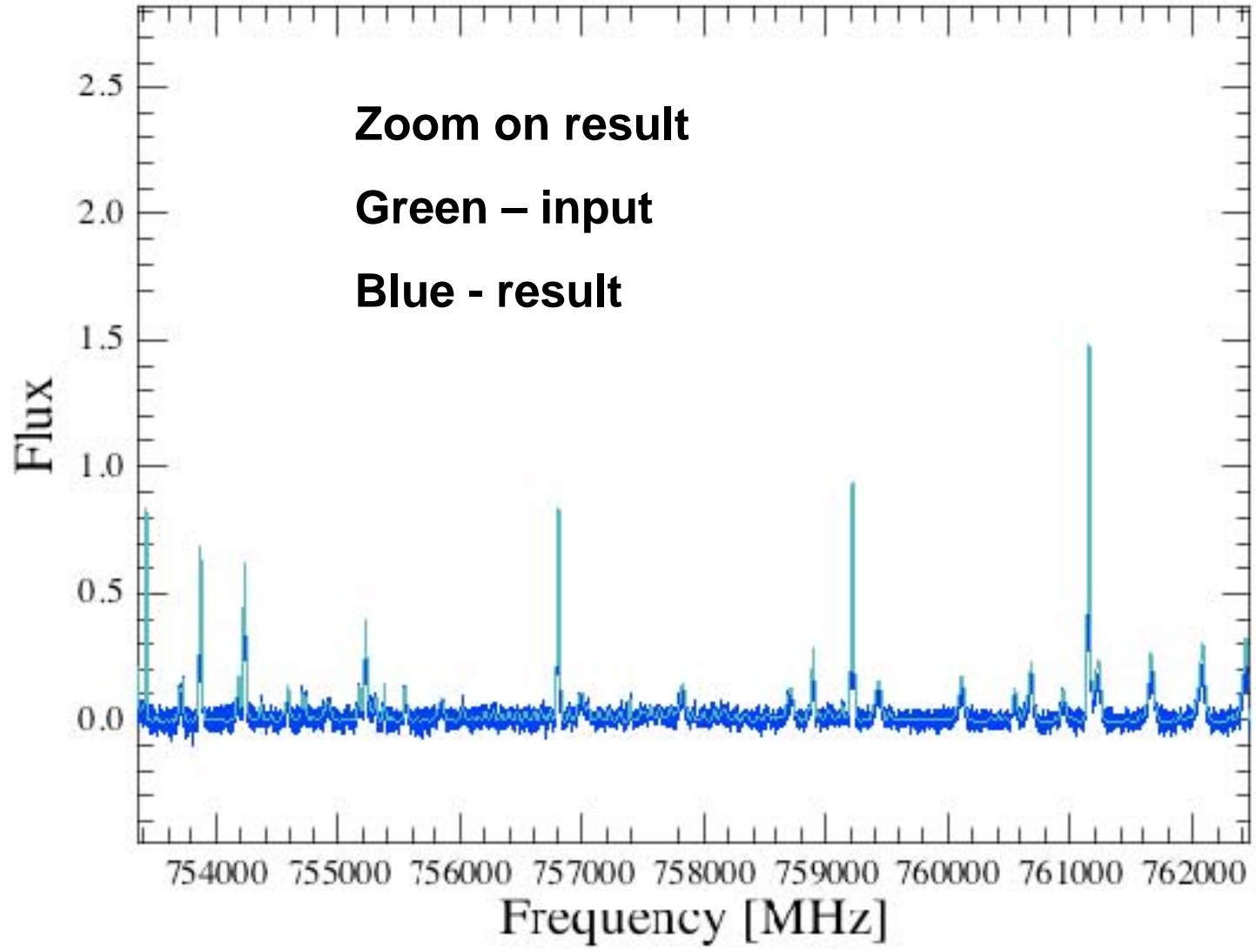
**Input spectrum
Green**

**Output Spectrum
Blue**





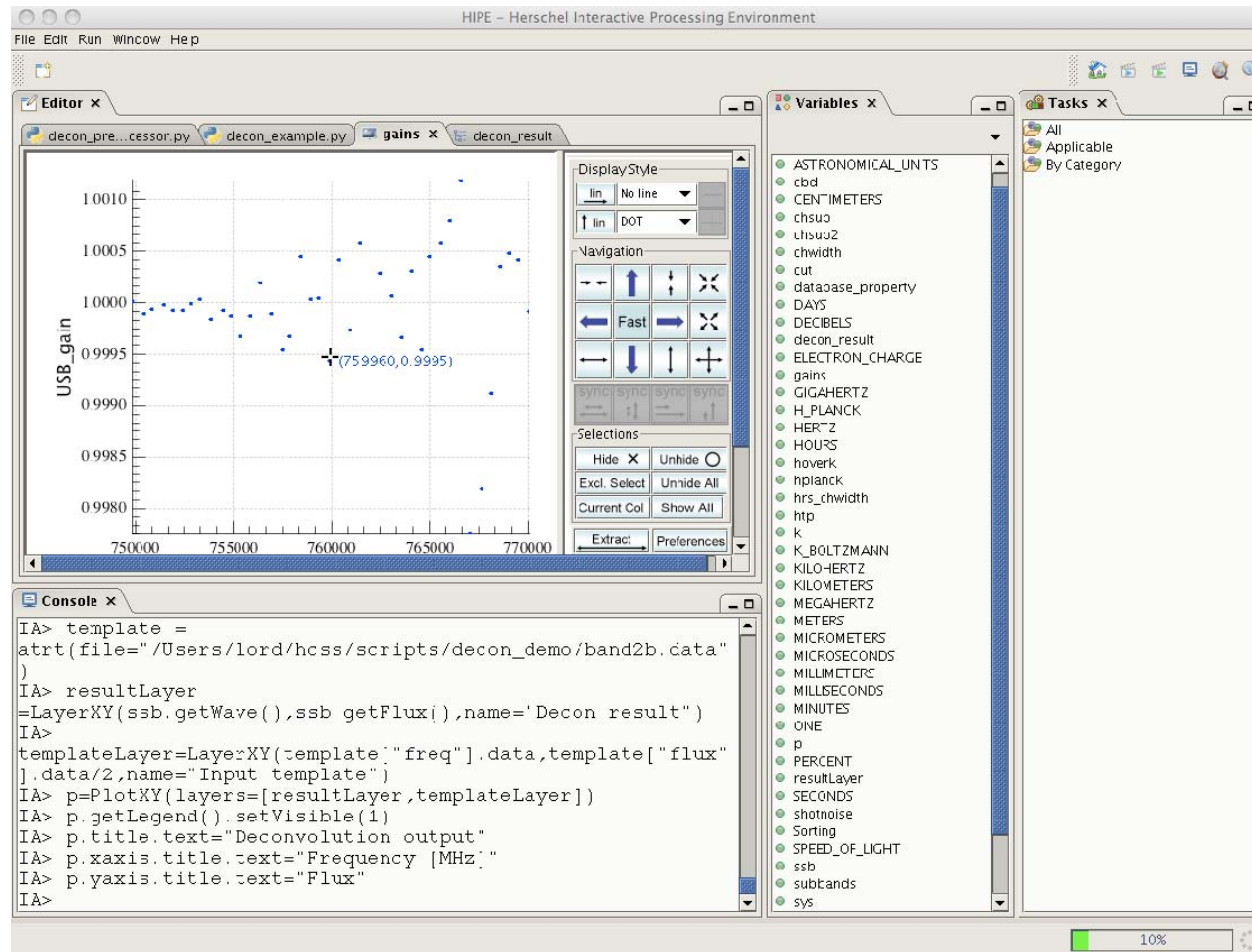
Deconvolution output



— Deconresult — Input template



Result – Gains for scans





Result – Gains Printed



HIPE – Herschel Interactive Processing Environment

File Edit Run Window Help

Editor x

gains decon_result decon_res...t["gain"] decon_res...t["gain"] x

Meta Data
None

Table Data
TableDataset

| Index | LO_freq | USB_gain | LSB_gain |
|-------|----------|-------------|-------------|
| 0 | 749863.0 | 1.000011... | 0.999951... |
| 1 | 750483.0 | 0.999899... | 0.999977... |
| 2 | 750864.0 | 0.999941... | 1.000031... |
| 3 | 751449.0 | 0.999983... | 0.999960... |
| 4 | 751911.0 | 0.999926... | 1.000004... |
| 5 | 752394.0 | 0.999924... | 0.999889... |
| 6 | 752862.0 | 0.999995... | 1.000051... |
| 7 | 753333.0 | 1.000036... | 0.999944... |
| 8 | 753864.0 | 0.999844... | 0.999767... |
| 9 | 754500.0 | 0.999931... | 1.000136... |
| 10 | 754895.0 | 0.999869... | 0.999913... |
| 11 | 755291.0 | 0.999675... | 1.000069... |
| 12 | 755899.0 | 0.999874... | 0.999920... |
| 13 | 756402.0 | 1.000196... | 1.000143... |
| 14 | 756945.0 | 0.999900... | 1.000520... |
| 15 | 757494.0 | 0.999547... | 1.000231... |
| 16 | 757923.0 | 0.999670... | 0.999749... |
| 17 | 758462.0 | 1.000446... | 1.000071... |

Console x

```
IA> template =
atrt(file="/Users/lord/hcss/scripts/decon_demo/band2b.data"
)
IA> resultLayer
=LayerXY(ssb.getWave(),ssb.getFlux(),name='Decon result')
IA>
templateLayer=LayerXY(template["freq"].data,template["flux"
].data/2,name="Input template")
IA> p=PlotXY(layers=[resultLayer,templateLayer])
IA> p.getLegend().setVisible(1)
IA> p.title.text="Deconvolution output"
IA> p.xaxis.title.text="Frequency [MHz]"
IA> p.yaxis.title.text="Flux"
IA>
```

Variables x

- chsuo2
- chwidth
- cut
- database_property
- DAYS
- DECIBELS
- decon_result
- ELECTRON_CHARGE
- gains
- GIGAHERTZ
- H_PLANCK
- HERTZ
- HOURS
- hoverk
- hplanck
- hrs_chwidth
- htp
- k
- K_BULLZMANN
- KILOHERTZ
- KILOWETERS
- MEGAHERTZ
- METERS
- MICROMETERS
- MICROSECONDS
- MILLIMETERS
- MILLISECONDS
- MINUTES
- ONE
- p
- PERCENT
- resultLayer
- SECONDS
- shotnoise
- Sorting
- SPEED_OF_LIGHT
- ssb
- subbands
- sys
- template
- templateLayer
- TERAHERTZ
- zbd

Tasks x

- All
- Applicable
- By Category

10%

