Data products

Dario Fadda
(USRA)

Pipeline team

Bill Vacca

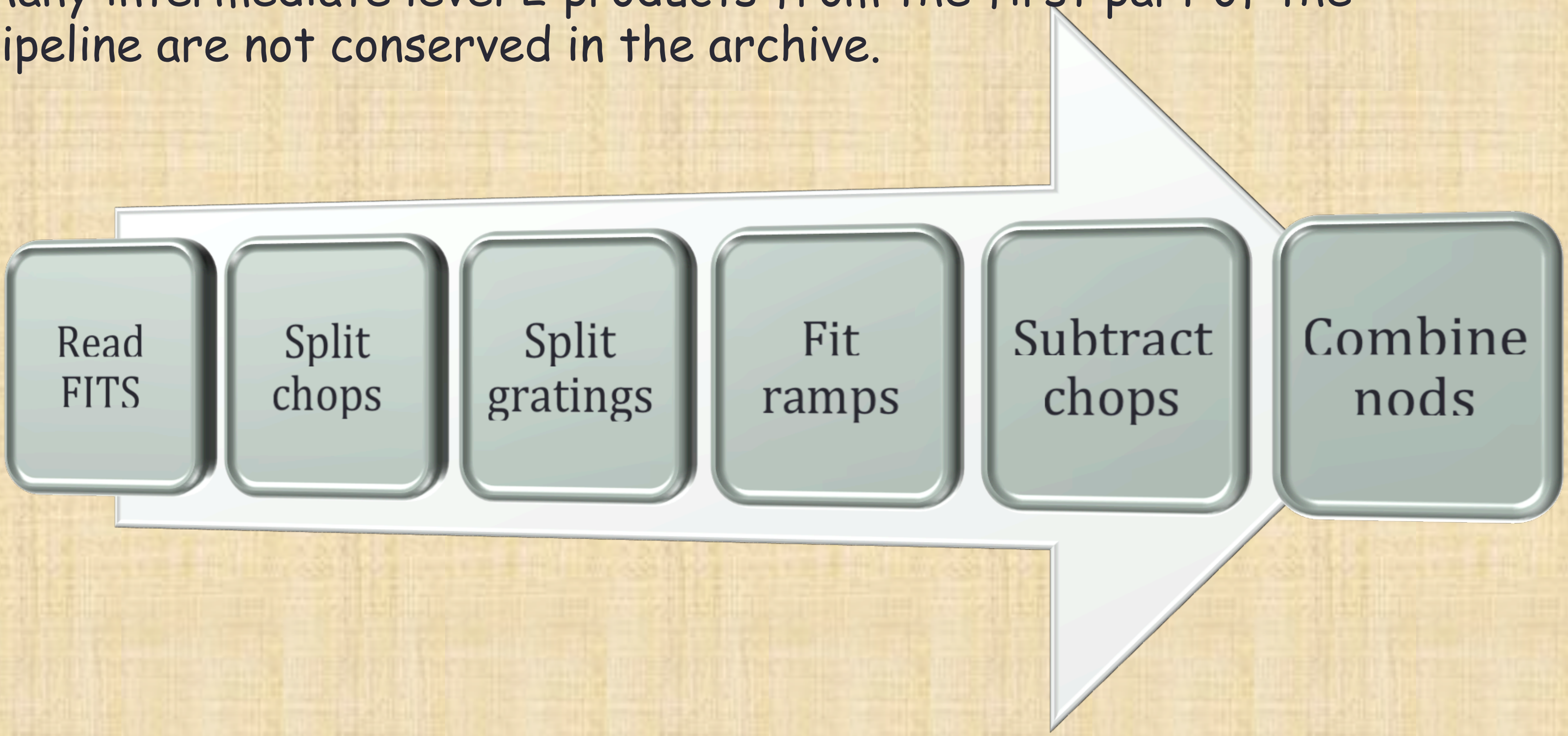
Melanie Clarke

Dario Fadda

Pipeline (levels 1 → 2)

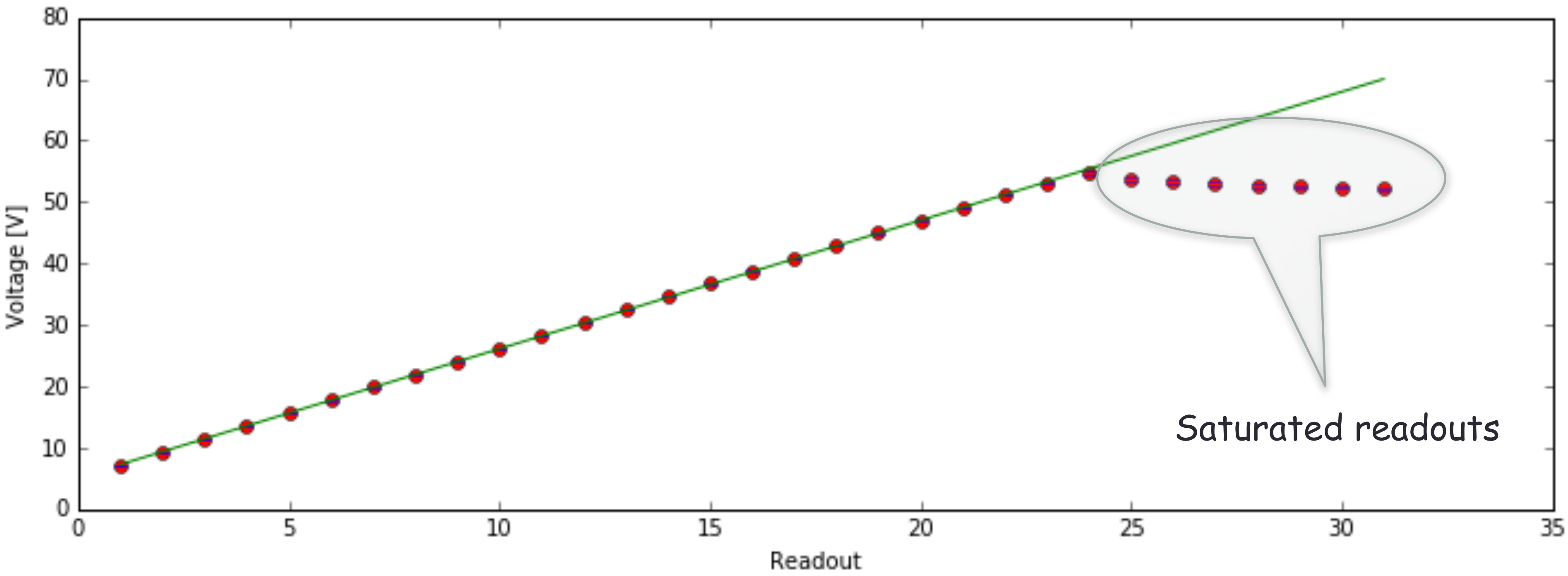
The pipeline consists in a sequence of modules.
For each module, files are created and read in the subsequent module. The raw files are retrievable as Level 1.

Many intermediate level 2 products from the first part of the pipeline are not conserved in the archive.



Fitting ramps

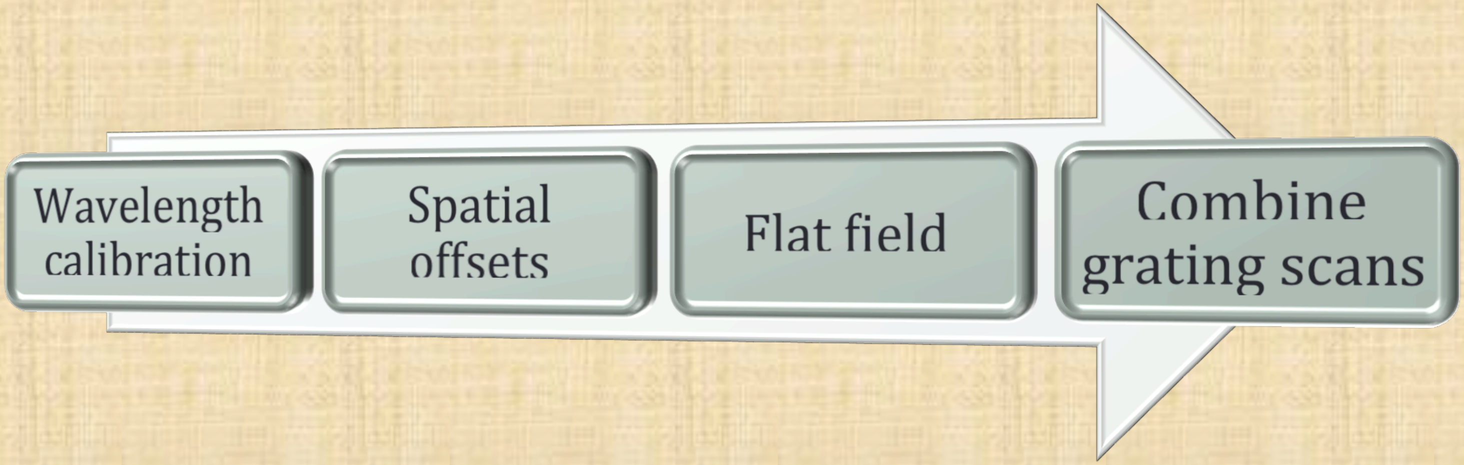
Ramps are fitted to estimate the flux on the detectors.
The pipeline uses a robust fit and excludes all saturated points from the fit to provide good estimates also for high flux observations.
Non-linearity is negligible (less than 1%).



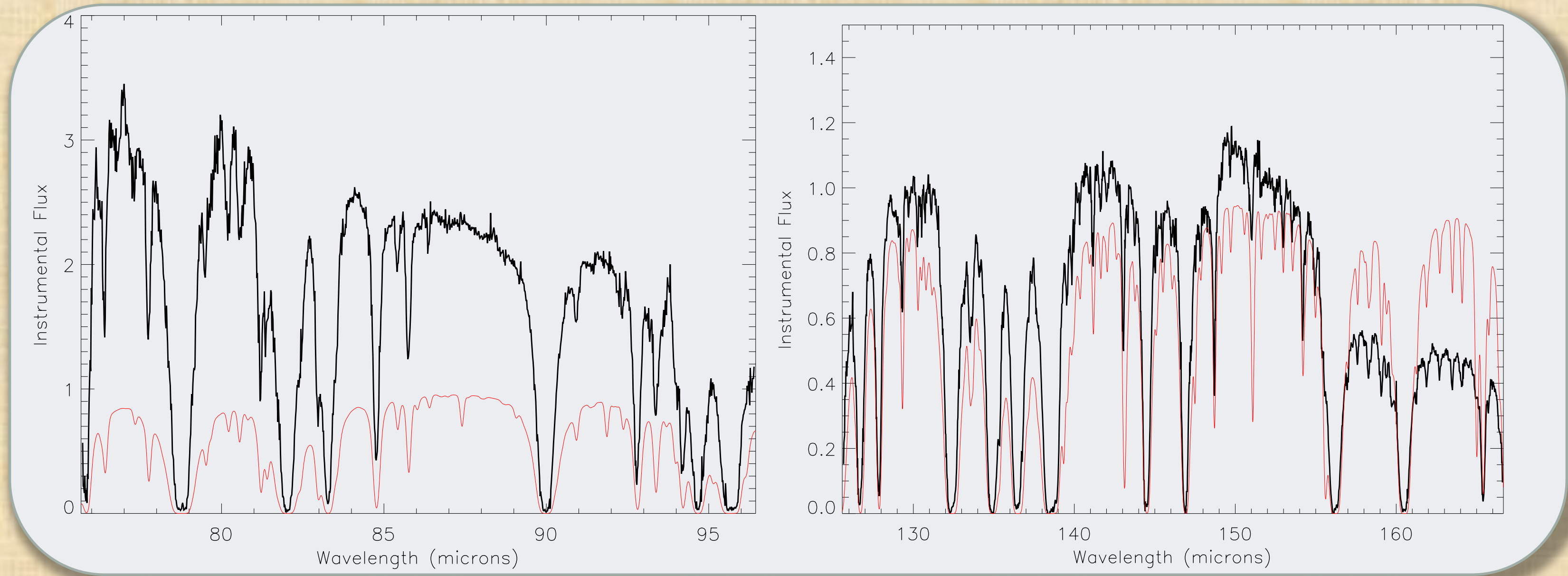
Pipeline (levels 1 → 2)

Once grating positions are translated into wavelengths and WCS is assigned to the FITS files, the data are flat-fielded and different grating scans are combined to obtain a series of spectra.

This results in the level 2 product in the archive which has the acronym SCM (for scan combined) in their names.



Wavelength calibration

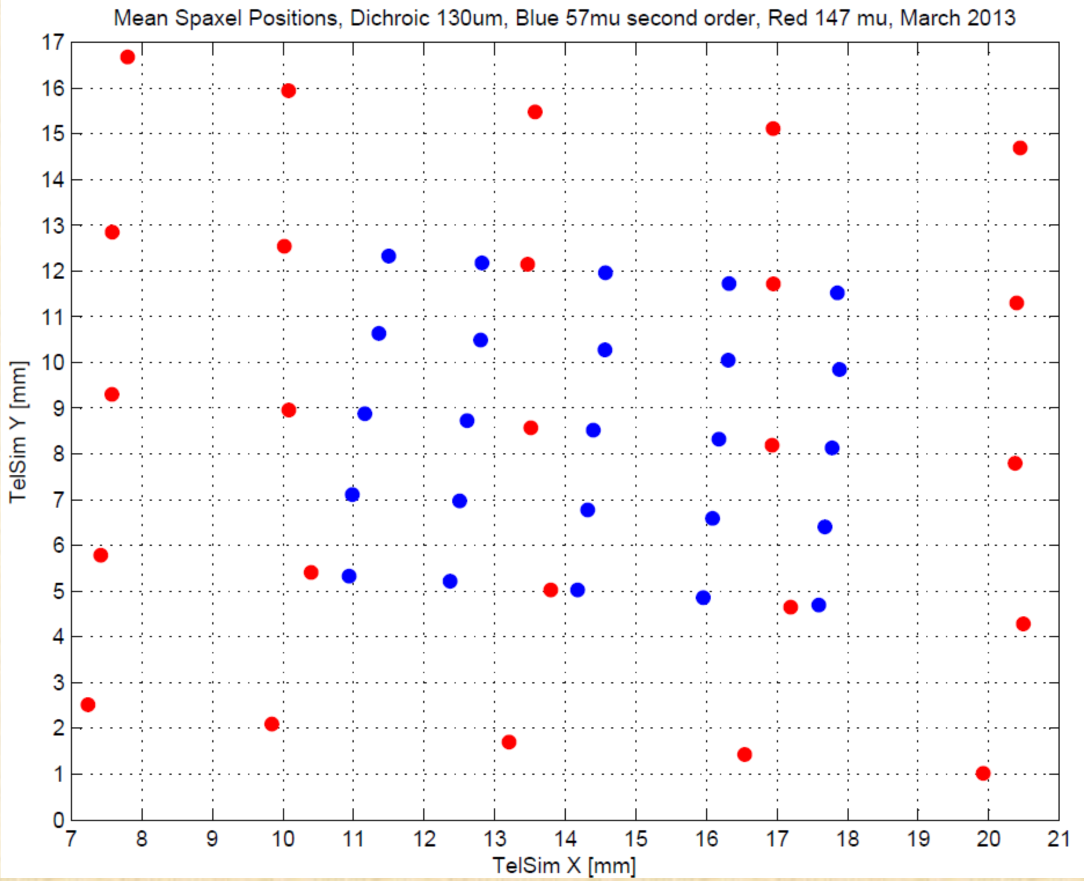


Mars spectrum (black) vs ATRAN (red)

Each of the 25 spaxels has its own wavelength calibration for the 16 spectral pixels. The wavelength calibration is good to 15% of a resolution element.

Each pixel has its own spectral width. This is taken into account when combining the different scans into a common wavelength grid.

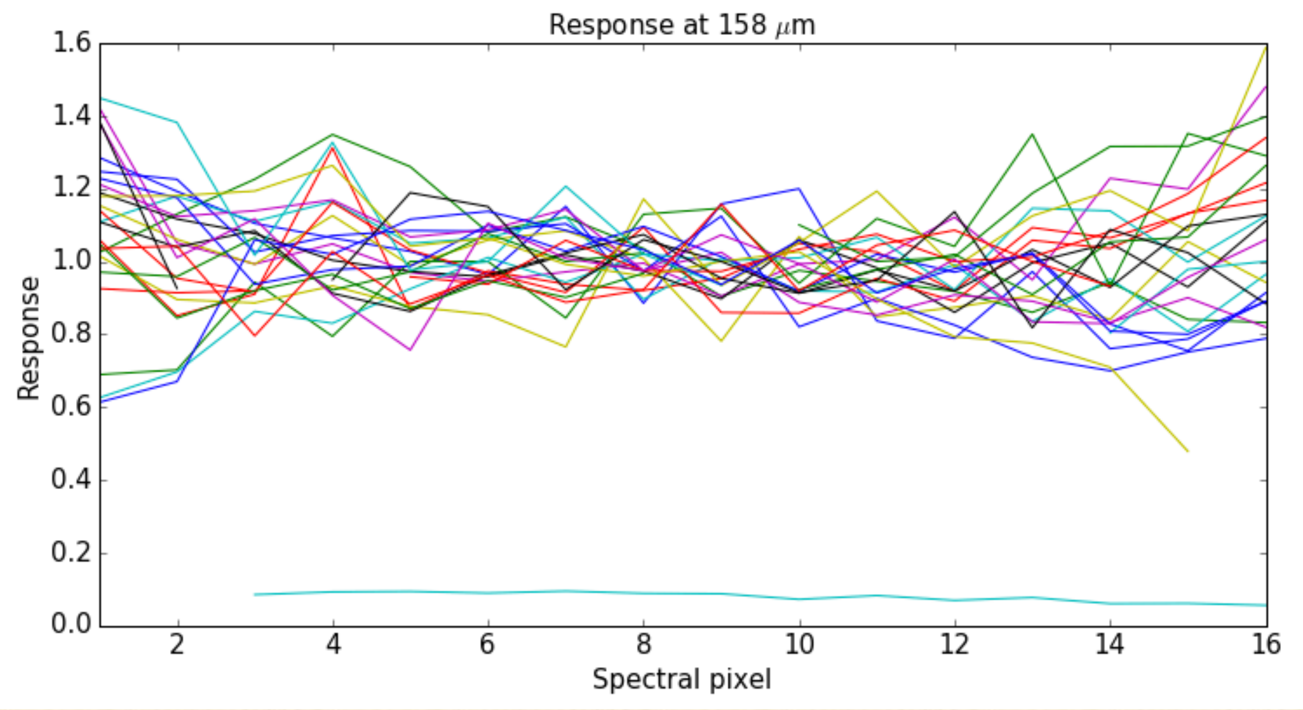
Spatial calibration



Spaxels are not on a regular grid. The relative sky position of each pixel is computed and stored. WCS for each cube is computed from dither offsets and reference position. The final cube has N up and E left.

Flat fields

Flat fields are 5 x 5 x 16 cubes computed in the lab for several key grating positions. For other cases, interpolated values are used.



We are in the process of computing flat fields from flight data. These new flats will be used in an upcoming data release.

SCM files

```
from astropy.io import fits
path = '/Users/dfadda/my_mounts/preview/LEVEL_4/FIFI-LS/2826/g102/'
ffile = 'F0283_FI_IFS_0401163_BLU_SCM_200167-200168.fits'
hdulist = fits.open(path+ffile)
hdulist.info()
print 'Extension 1:', hdulist[1].data.columns
```

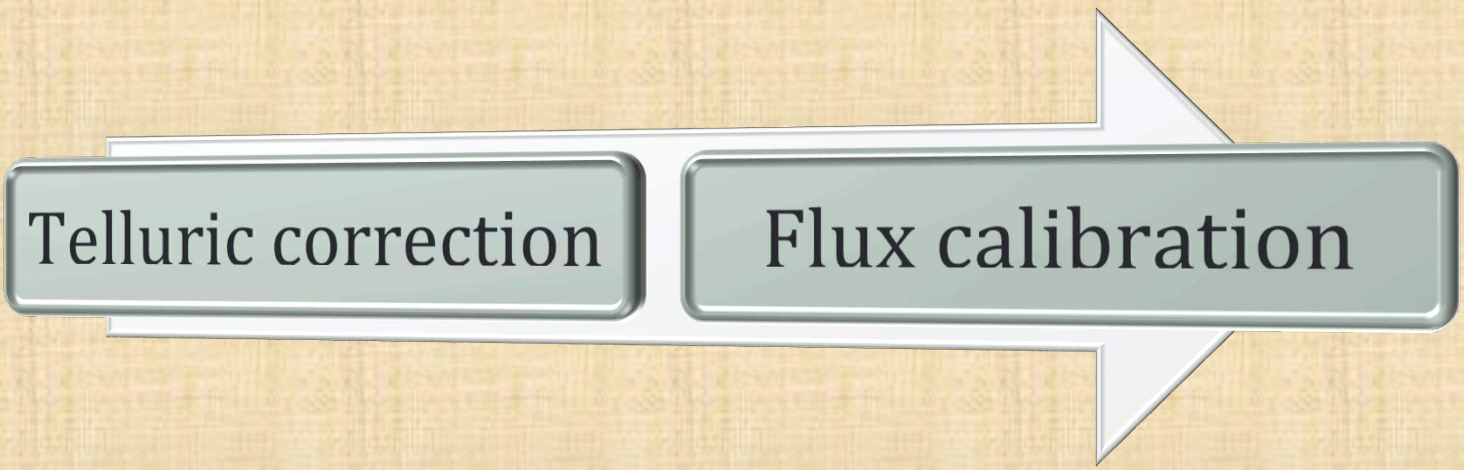
```
Filename: /Users/dfadda/my_mounts/preview/LEVEL_4/FIFI-LS/2826/g102/F0283_FI_IFS_0401163_BLU_
SCM_200167-200168.fits
No.    Name         Type      Cards   Dimensions   Format
0    PRIMARY     PrimaryHDU     278   ()
1                BinTableHDU     38   1R x 6C      [5040A, 1200D, 1200D, 1200D, 1200D, 1200D]

Extension 1: ColDefs(
    name = 'HEADER'; format = '5040A'; dim = '(80, 63)'
    name = 'DATA'; format = '1200D'; dim = '( 5, 5, 48)'
    name = 'STDDEV'; format = '1200D'; dim = '( 5, 5, 48)'
    name = 'LAMBDA'; format = '1200D'; dim = '( 5, 5, 48)'
    name = 'XS'; format = '1200D'; dim = '( 5, 5, 48)'
    name = 'YS'; format = '1200D'; dim = '( 5, 5, 48)'
)
```

Level 2 data are conserved as several $5x5xN_{\lambda}$ cubes.

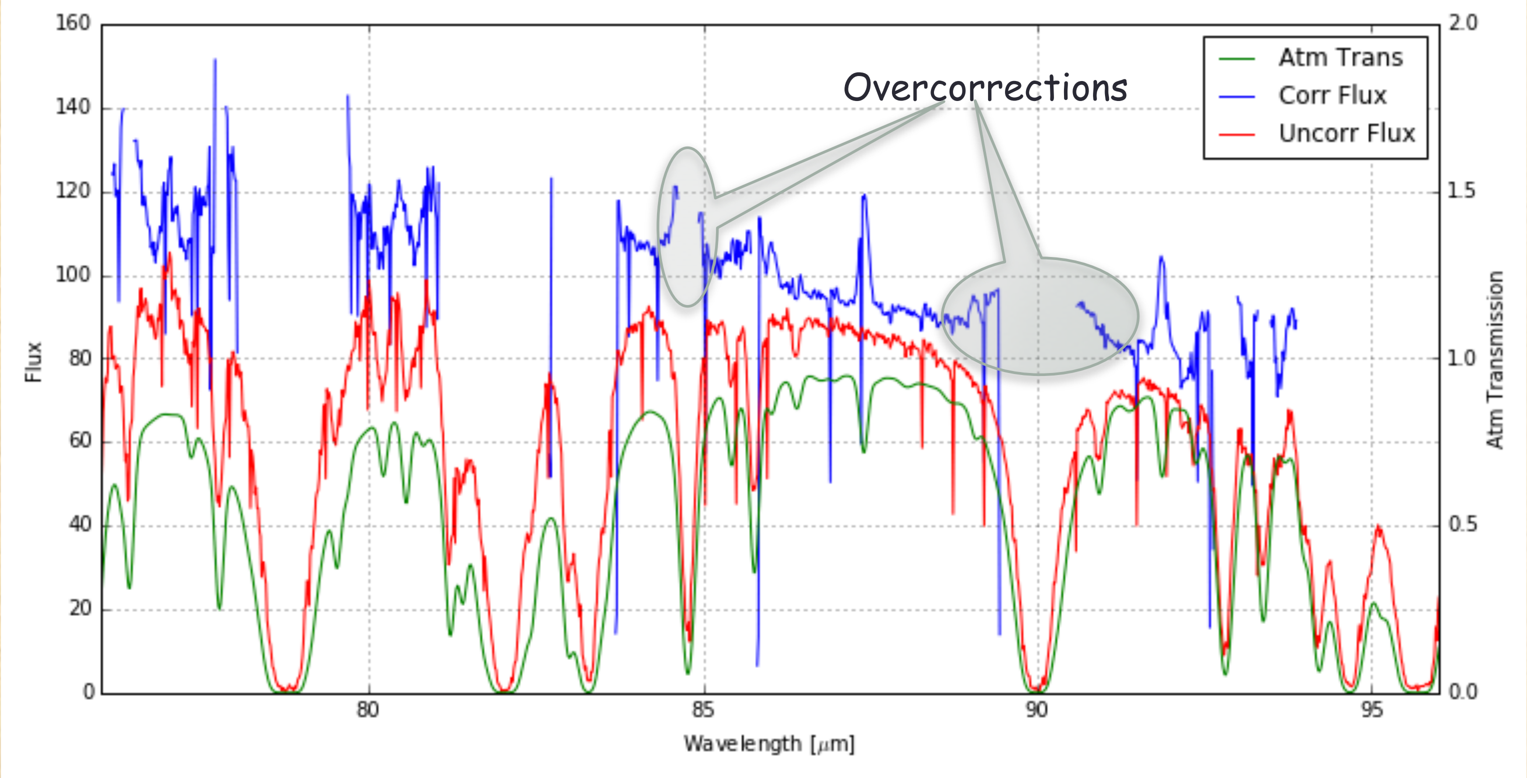
Pipeline (levels 2 → 3)

At this point, each file is corrected for atmospheric transmission and flux calibration is applied.

The resulting Level 3 products have the letters CAL (for calibrated) in their names.



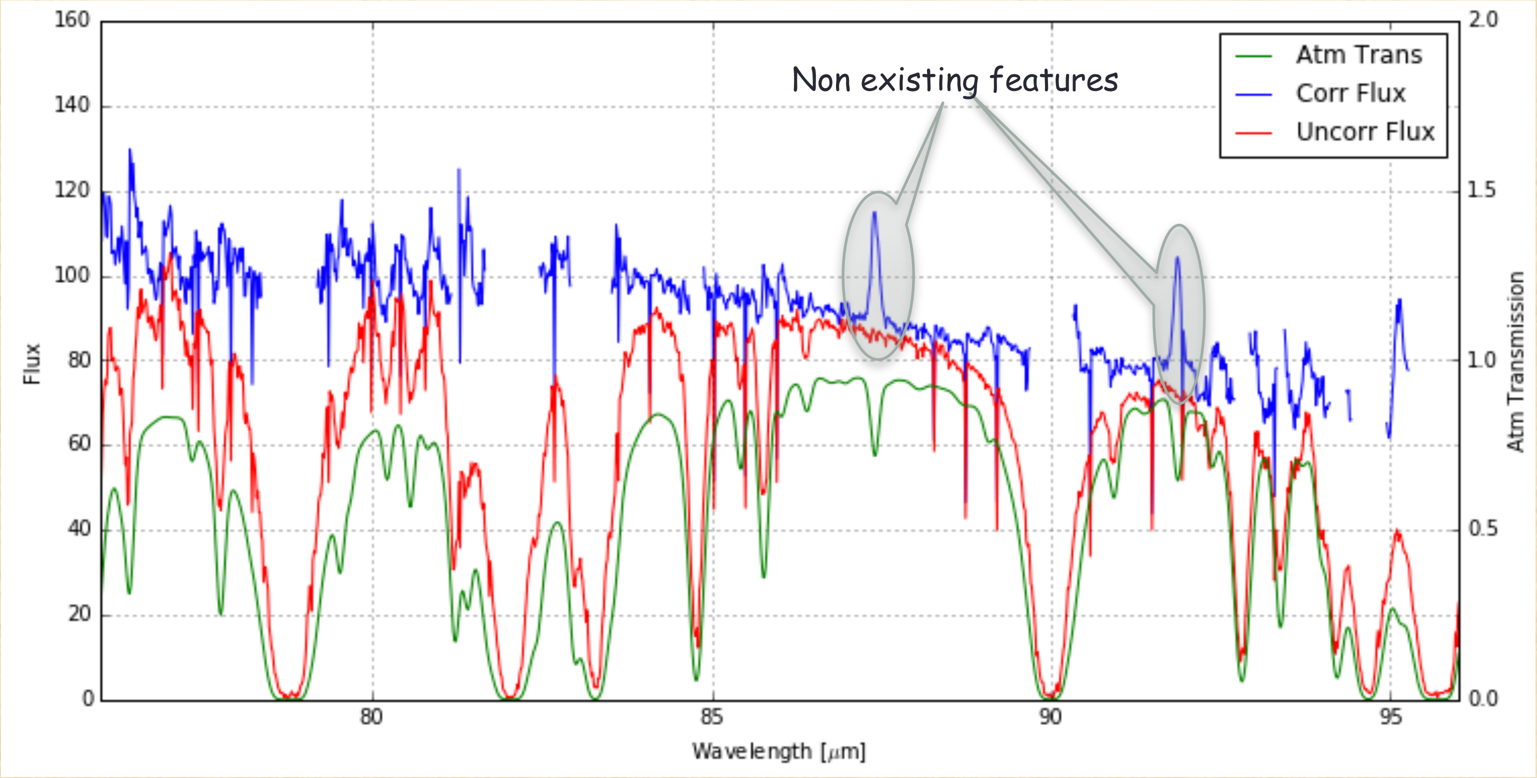
Telluric correction



A correction is applied to all the data with transmission greater than 60%. Below this value, data are blanked. The correction assumes a standard PWV value, since the real one is not yet measured. You can generate models of atmospheric transmission with ATRAN: https://atran.arc.nasa.gov/cgi-bin/atran/atran.cgi

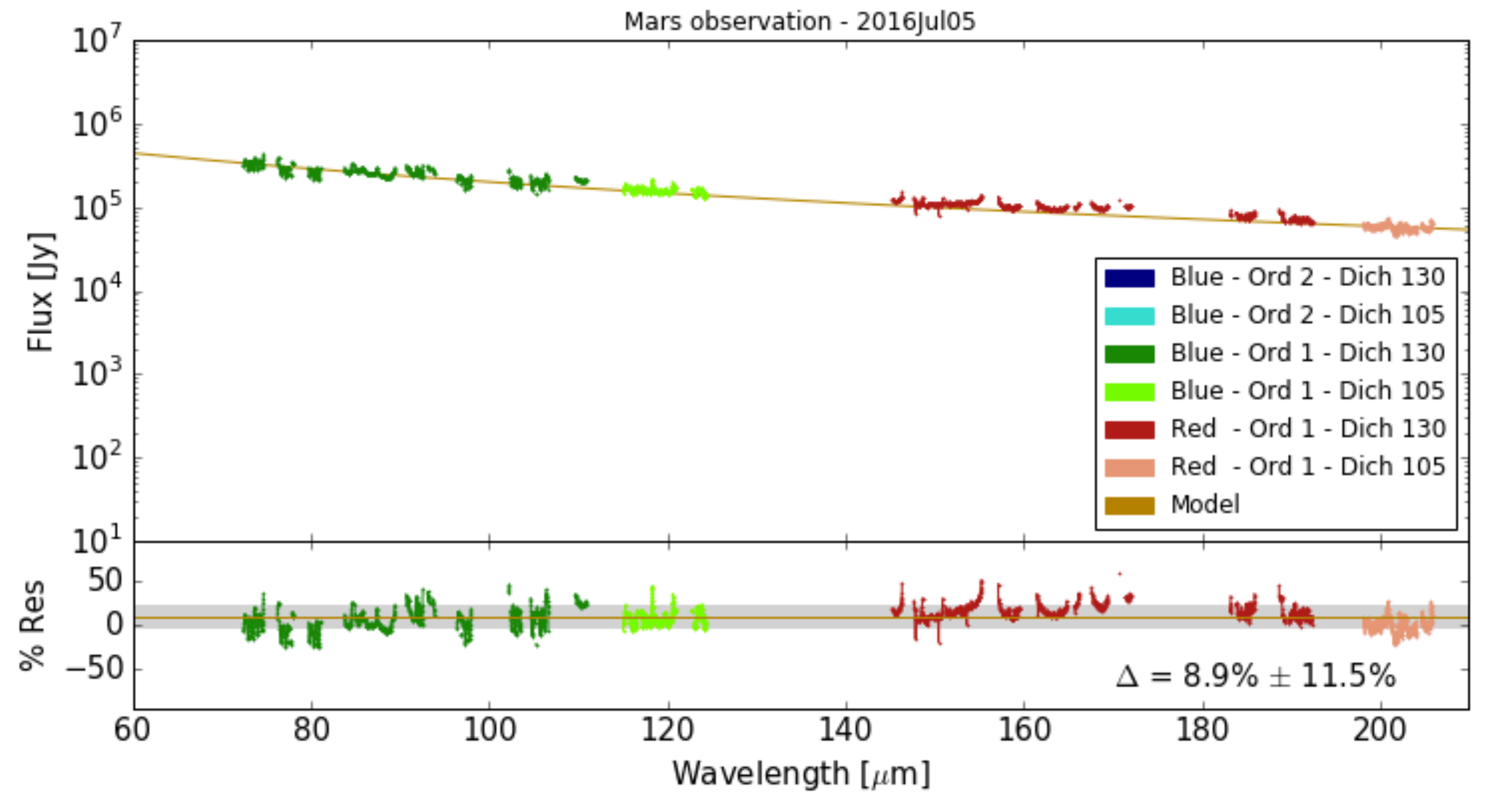
To allow the GIs to use the data below this threshold, the final cube has a extension with uncorrected data. A transmission correction should be applied to this data to get useful fluxes.

Telluric correction



We can correct the flux by using a lower water vapor value (5 for instance) and obtaining something not overcorrected.
Note that some ATRAN features do not appear in the observed data. So, it's a good precaution to plot the spectrum against the atmospheric transmission.
Spikes in the spectrum are edge effects of the wavelength interpolation between adjacent pieces of different spectroscopic observations.

Flux calibration



Based on Mars observation and theoretical spectrum by Lellouch & Amri from 60 μm to 300 μm, extended to 40 μm with black-body curve. Response derived for each combination of filter/orders and dichroics. Comparison with previous observations shows accuracy of 20%. This can be improved with knowledge of the PWV.

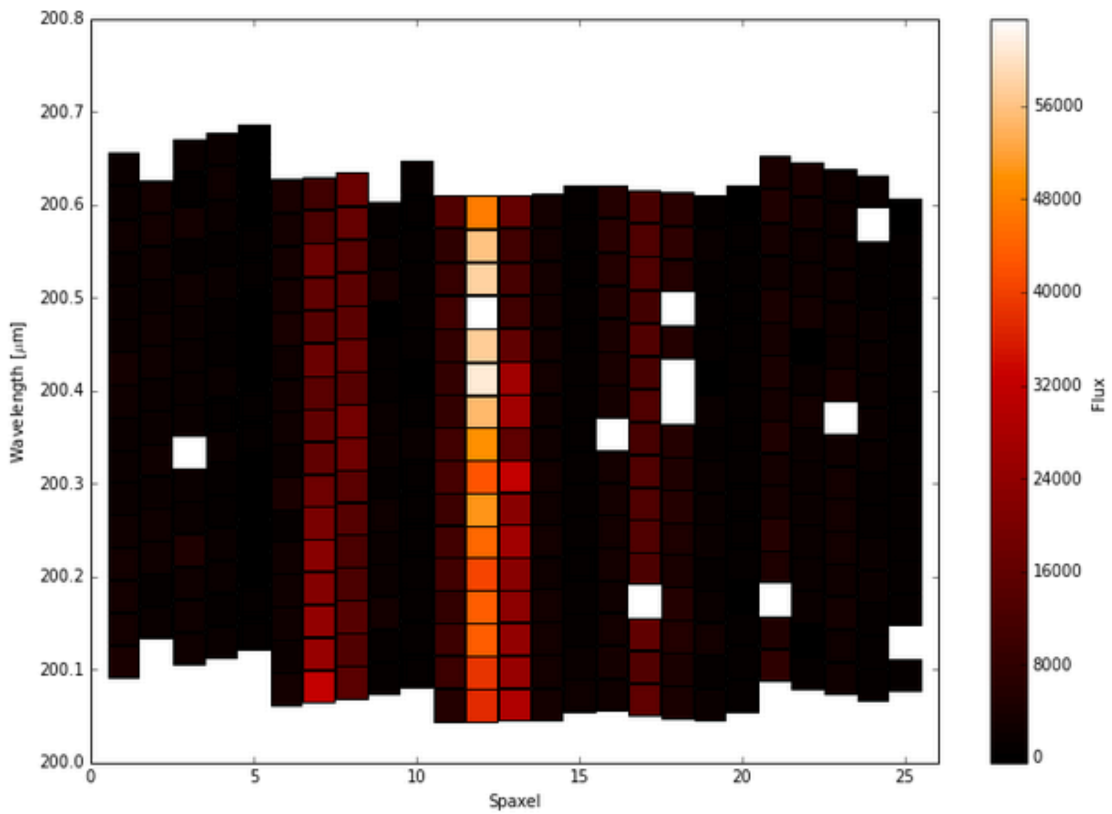
CAL files

```
from astropy.io import fits
path = '/Users/dfadda/my_mounts/preview/LEVEL_4/FIFI-LS/2826/g102/'
ffile = 'F0283_FI_IFS_0401163_BLU_CAL_200169-200170.fits'
hdulist = fits.open(path+ffile)
hdulist.info()
print 'Extension 1:', hdulist[1].data.columns
```

```
Filename: /Users/dfadda/my_mounts/preview/LEVEL_4/FIFI-LS/2826/g102/F0283_FI_IFS_0401163_BLU_
CAL_200169-200170.fits
No.    Name         Type      Cards   Dimensions   Format
0    PRIMARY     PrimaryHDU     282   ()
1                BinTableHDU     50   1R x 10C     [5040A, 1200D, 1200D, 1200D, 1200D, 1200D,
 1200D, 1200D, 1200D, 1200D]
Extension 1: ColDefs(
    name = 'HEADER'; format = '5040A'; dim = '(80, 63)'
    name = 'DATA'; format = '1200D'; dim = '( 5, 5, 48)'
    name = 'STDDEV'; format = '1200D'; dim = '( 5, 5, 48)'
    name = 'UNCORRECTED_DATA'; format = '1200D'; dim = '( 5, 5, 48)'
    name = 'UNCORRECTED_STDDEV'; format = '1200D'; dim = '( 5, 5, 48)'
    name = 'LAMBDA'; format = '1200D'; dim = '( 5, 5, 48)'
    name = 'XS'; format = '1200D'; dim = '( 5, 5, 48)'
    name = 'YS'; format = '1200D'; dim = '( 5, 5, 48)'
    name = 'ATRAN'; format = '1200D'; dim = '( 5, 5, 48)'
    name = 'RESPONSE'; format = '1200D'; dim = '( 5, 5, 48)'
)
```

The calibrated files contain the median atmospheric transmission and response used as well as the data uncorrected for atmospheric absorption.

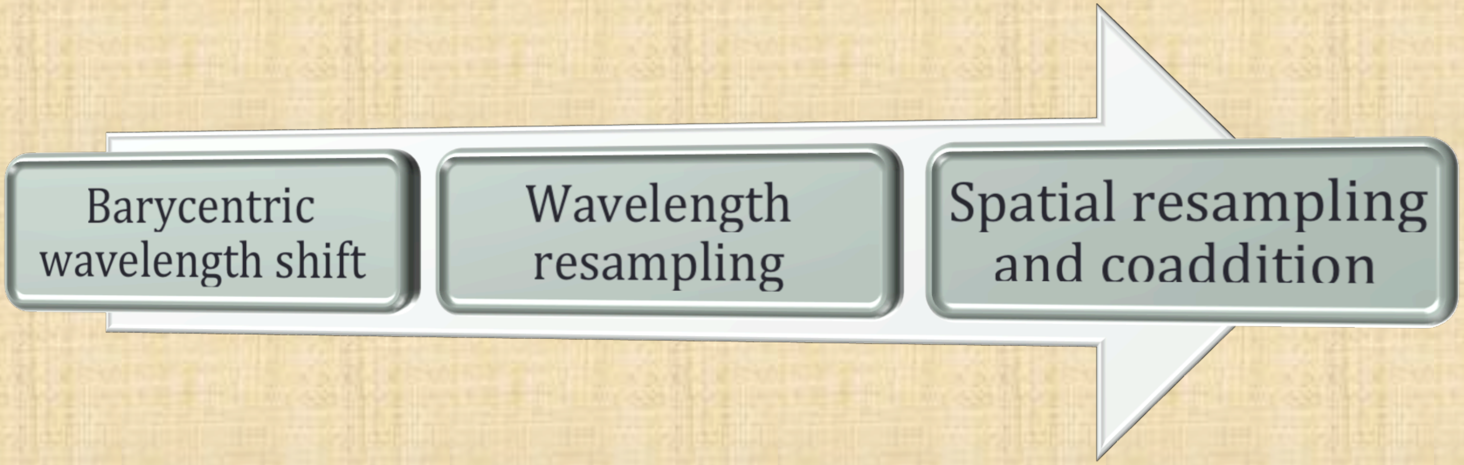
```
from astropy.io import fits
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
path = '/Users/dfadda/my_mounts/preview/2016-07-05_FI_F316/2621-ARC/g14/'
ffile = 'F0316_FI_IFS_90003321_RED_CAL_400079-400080.fits'
hdulist = fits.open(path+ffile); cols = hdulist[1].data.columns
wave = cols['LAMBDA'].array; flux = cols['DATA'].array; nl = wave.shape[1]

xw,yw = np.meshgrid(np.arange(1,26),np.arange(nl))
w = wave[0,:,:,:].reshape(nl,25); f = flux[0,:,:,:].reshape(nl,25)
plt.figure(figsize=(12,8.1))
plt.ticklabel_format(useOffset=False, style='plain')
cm = plt.cm.get_cmap('gist_heat')
sc=plt.scatter(xw,w,c=f,marker='s',s=380,cmap=cm)
cbar=plt.colorbar(sc); cbar.ax.set_ylabel('Flux')
plt.xlim([0,26]); plt.xlabel('Spaxel'); plt.ylabel('Wavelength [$\mu$m]')
plt.show()
```



Pipeline (levels 2 → 3)

The calibrated files are now resampled into a regular wavelength grid and each wavelength plane is projected into a rectilinear spatial grid.

The files with WGR in their names (for wavelength grid) contain cubes after wavelength resampling.



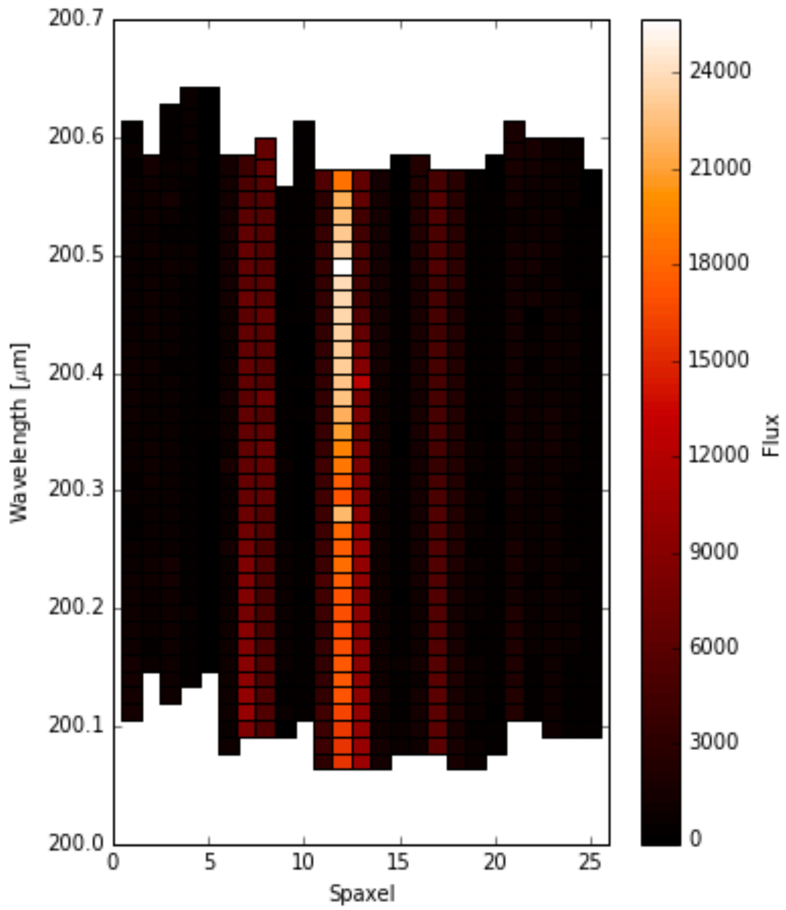
WGR files

```
from astropy.io import fits
path = '/Users/dfadda/my_mounts/preview/LEVEL_4/FIFI-LS/2826/g102/'
ffile = 'F0283_FI_IFS_0401163_BLU_WGR_200161-200162.fits'
hdulist = fits.open(path+ffile)
hdulist.info()
```

```
Filename: /Users/dfadda/my_mounts/preview/LEVEL_4/FIFI-LS/2826/g102/F0283
_FI_IFS_0401163_BLU_WGR_200161-200162.fits
No.    Name         Type      Cards   Dimensions   Format
0    PRIMARY     PrimaryHDU     292   ()
1    FLUX        ImageHDU         8   (56, 25)     float64
2    ERROR       ImageHDU         8   (56, 25)     float64
3    UNCORRECTED_FLUX  ImageHDU         8   (56, 25)     float64
4    UNCORRECTED_ERROR  ImageHDU         8   (56, 25)     float64
5    WAVELENGTH  ImageHDU         7   (56,)        float64
6    X           ImageHDU         7   (25,)        float64
7    Y           ImageHDU         7   (25,)        float64
8    TRANSMISSION  ImageHDU         7   (56,)        float32
9    RESPONSE    ImageHDU         7   (56,)        float32
```

Starting from WGR files, each extension contain different data.
Data are, in this case, re-gridded along 56 wavelength values for each of the 25 spaxels. There are now only 25 positions.

```
path = '/Users/dfadda/my_mounts/preview/2016-07-05_FI_F316/2621-ARC/g14/'
ffile = 'F0316_FI_IFS_90003321_RED_WGR_400079-400080.fits'
hdulist = fits.open(path+ffile)
f = hdulist[1].data; w = hdulist[5].data; nl=len(w)
xw,yw = np.meshgrid(np.arange(nl),np.arange(1,26))
ww,wz = np.meshgrid(w,np.arange(1,26))
plt.figure(figsize=(5.5,7))
plt.ticklabel_format(useOffset=False, style='plain')
cm = plt.cm.get_cmap('gist_heat')
sc=plt.scatter(yw,ww,c=f,marker='s',s=100,cmap=cm)
cbar=plt.colorbar(sc); cbar.ax.set_ylabel('Flux')
plt.xlim([0,26]); plt.xlabel('Spaxel'); plt.ylabel('Wavelength [$\mu$m]')
plt.show()
```



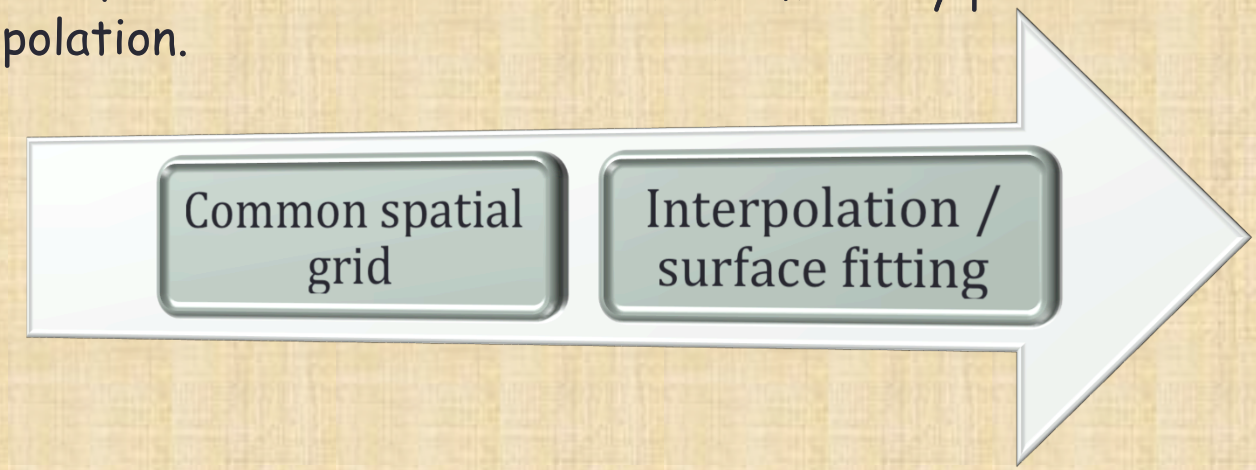
Pipeline (levels 3 → 4)

The final step consists in obtaining a cube with regular spatial grid over which all the data are coadded.
The final cube is conserved with the acronym WXY (for wavelength & spatial rebinned) in their names.

Two methods for spatial resampling can be used:

- Interpolation (using IDL radial basis functions)
- Local polynomial fitting

In the archive usually the default method is fitting.
In case of undithered observations we manually process data using interpolation.



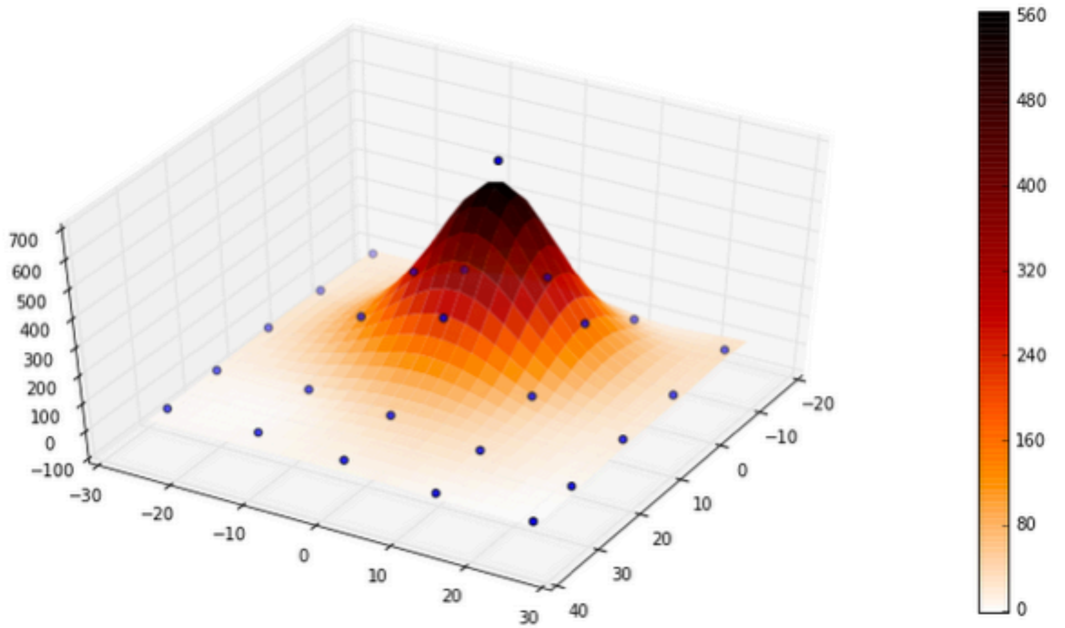
WXY file

```
from astropy.io import fits
path = '/Users/dfadda/my_mounts/preview/LEVEL_4/FIFI-LS/2826/g102/'
ffile = 'F0283_FI_IFS_0401163_BLU_WXY_100096-200218.fits'
hdulist = fits.open(path+ffile)
hdulist.info()
```

```
Filename: /Users/dfadda/my_mounts/preview/LEVEL_4/FIFI-LS/2826/g102/F0283
_FI_IFS_0401163_BLU_WXY_100096-200218.fits
No.    Name         Type      Cards   Dimensions   Format
0    PRIMARY     PrimaryHDU     311   ()
1    FLUX        ImageHDU        28   (381, 349, 56)   float64
2    ERROR       ImageHDU        28   (381, 349, 56)   float64
3    UNCORRECTED_FLUX  ImageHDU        28   (381, 349, 56)   float64
4    UNCORRECTED_ERROR  ImageHDU        28   (381, 349, 56)   float64
5    WAVELENGTH  ImageHDU         7   (56,)         float64
6    X           ImageHDU         7   (381,)        float64
7    Y           ImageHDU         7   (349,)        float64
8    TRANSMISSION  ImageHDU         7   (56,)         float32
9    RESPONSE    ImageHDU         7   (56,)         float32
10   EXPOSURE_MAP  ImageHDU        28   (381, 349, 56)   int16
```

Finally, all the WGR files are combined in a single cube which spatial size depends on the observation. Default pixel sizes are 1 and 2 sq. arcsec., for the blue and red arrays, respectively.

```
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D
from matplotlib import cm
path = '/Users/dfadda/my_mounts/preview/2016-07-06_FI_F317/2622-ARC/g22/'
ffile = 'F0317_FI_IFS_90003321_RED_WXY_200376-200443.fits'
hl = fits.open(path+ffile)
x=hl[6].data; y=hl[7].data; x,y = np.meshgrid(x,y)
f = hl[1].data; w = hl[5].data; n = 200
ax = Axes3D(plt.figure(figsize=(10,5))); ax.view_init(azim=30,elev=45)
surf=ax.plot_surface(x,y,f[n,:,:], rstride=1, cstride=1, linewidth=0,
                alpha=0.9, cmap=cm.gist_heat_r)
plt.colorbar(surf)
hl.close()
# Overplot data from WGR files
import os, fnmatch
wgr = fnmatch.filter(os.listdir(path),"*WGR*.fits")
for wgrfile in wgr:
    hlf = fits.open(path+wgrfile)
    xs=hlf[6].data; ys=hlf[7].data; fs = hlf[1].data[:,n]/36. # Area ratio
    ax.scatter(xs,ys,fs,c='blue',marker='o')
    hlf.close()
plt.show()
print "wavelength ", w[n], "um"
```




```
wavelength  193.161951427 um
```

Multi-mission grouping

In the case an observation is performed across several flights, data are first processed for each single flight and then combined at the last step of the pipeline.

The grouping is done using the keyword "FILEGPID" which is assigned manually before processing the data.

G.I. will find sometimes data of nearby observations grouped in a single final cube if the spatial and wavelength overlap is significant. If this grouping is not desirable for science (such is the case of repeated observations to detect variability) the G.I. should contact the SOFIA Science Center to split the data in multiple final cubes.

Interacting with the pipeline

Several interactive passages are done during the reduction:

- Exclude files which are of low quality (e.g. bad atmospheric transmission)
- Group files from different missions (FileGpID keyword)
- Change the threshold to reject bad ramp fits
- Use simple interpolation for the final spatial projection (typically with staring observation)
- Change the kernel width of the surface fitting for the spatial projection (in case of highly concentrated sources)

Typically these choices are done during the QA.

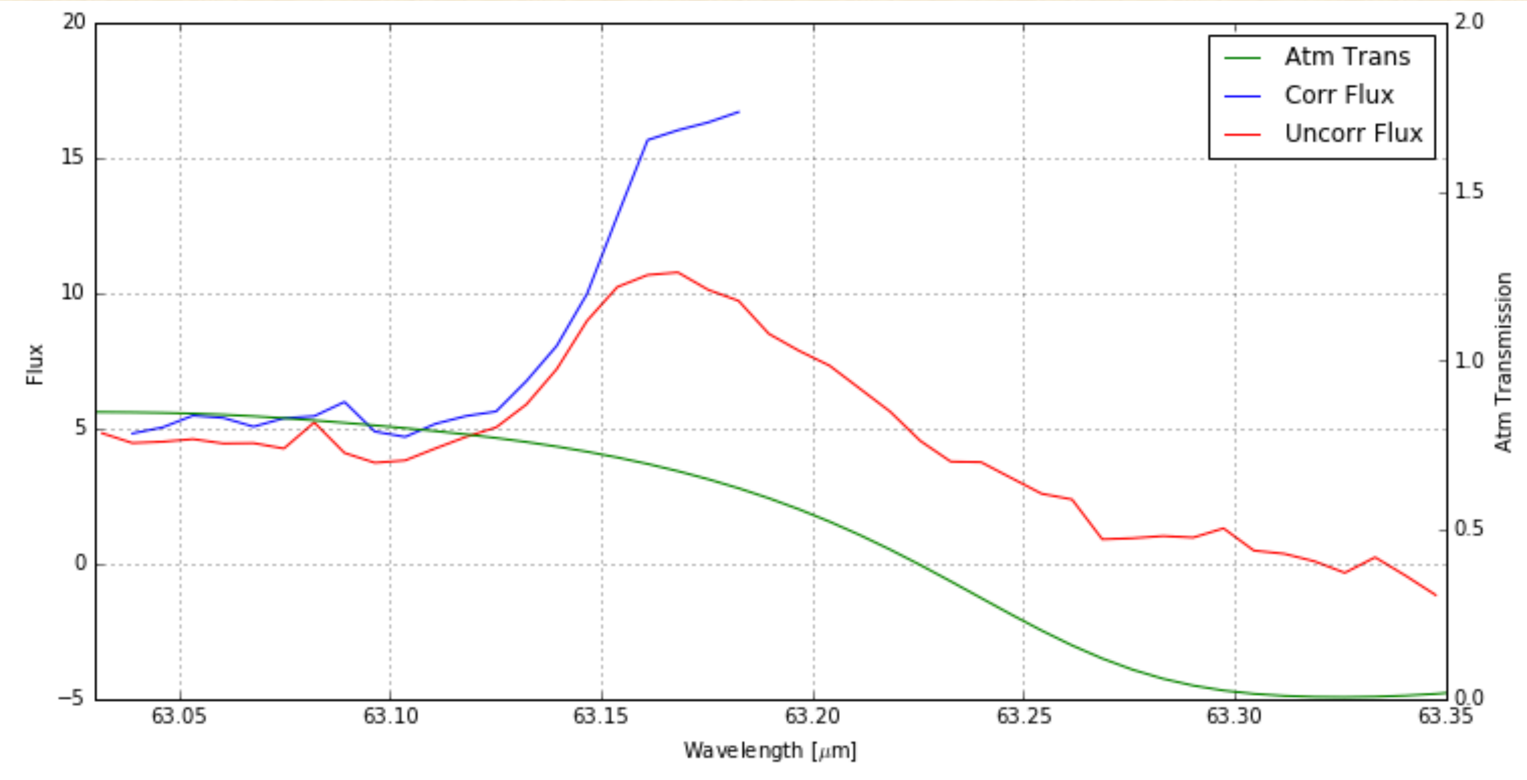
It is nevertheless useful to know about these possibilities in case the G.I. notice something strange in their data.

Caveat

A laundry list of possible problems:

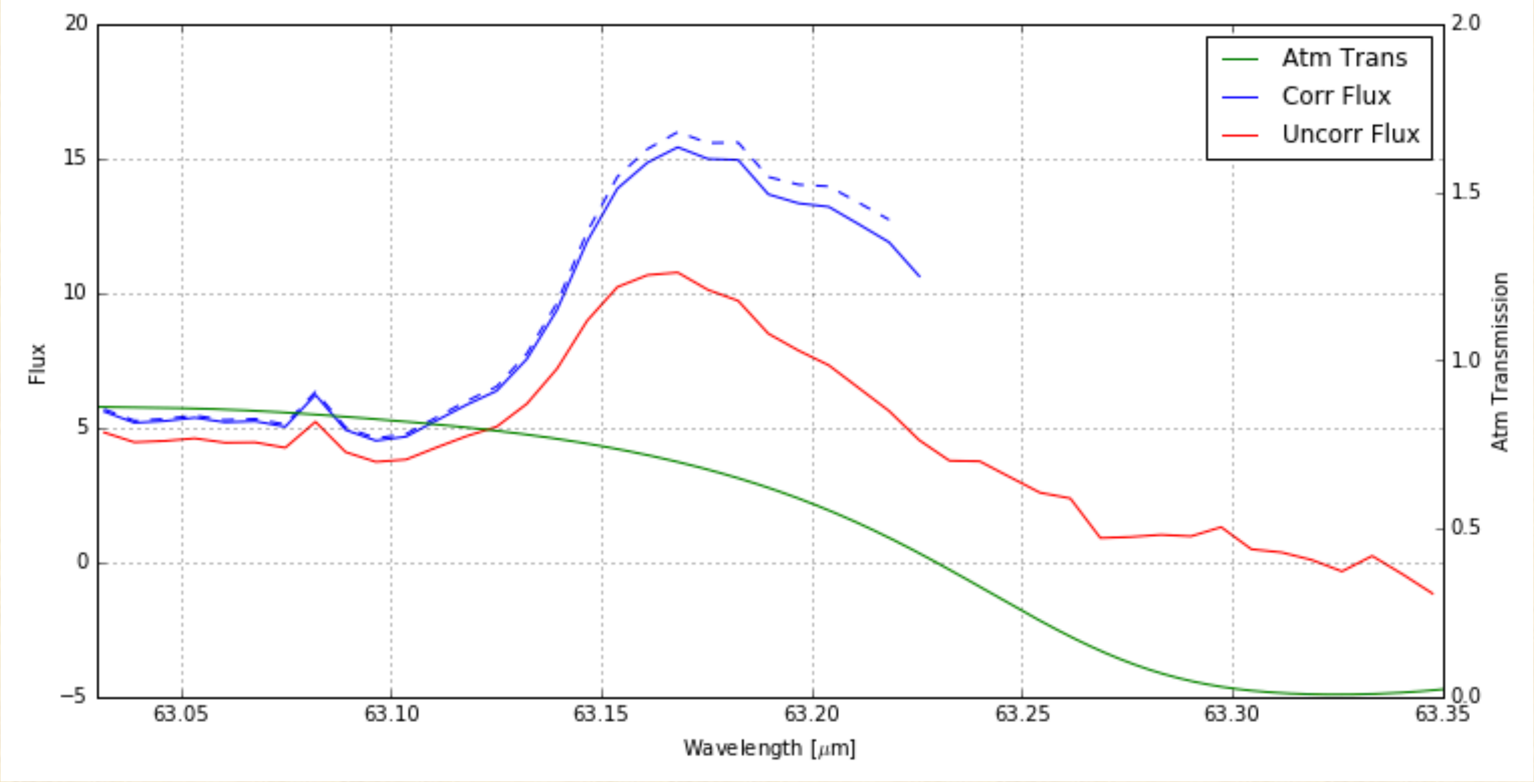
- Telluric lines: NaNs and overcorrections
- Spatial resampling – interpolation vs polynomial fit
- Negative continuum (bad reference position)
- Bad flats
- Ghosts for bright objects

Telluric correction



In the corrected data lines can be cut short because the absorption becomes important (transmission < 60%).
In this cases, we can use the uncorrected flux after correcting it with a lower transmission threshold.

Telluric correction



In this case we recomputed the transmission using ATRAN and values in the file header: RESOLUN, LAT_STA, LAT_END, ZA_START, ZA_END.
Better statistics for these keywords can be obtained from the WGR files.
We put the threshold to 40% to recover more of the line, be able to fit it, and estimate a flux.
Note that the correction works only approximately in case of unresolved/narrow features.

Pitfalls of spatial resampling

The last step of the pipeline which involves spatial resampling and coaddition of the data is the most critical one.

In the pipeline there are two methods:

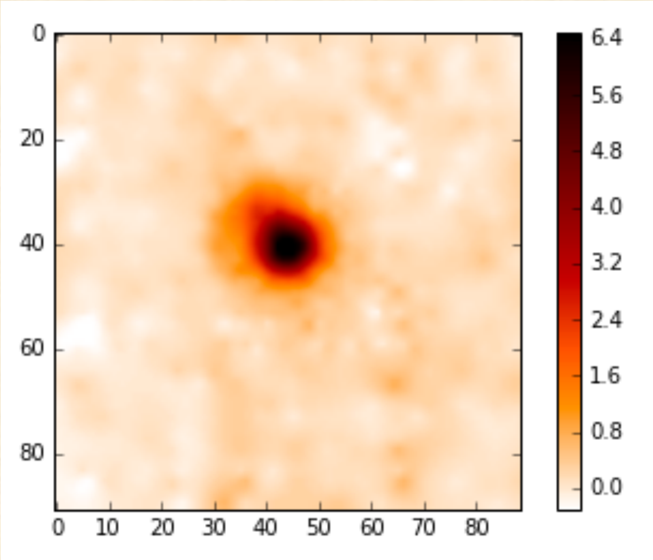
- **Fit** a 2D local surface fitting with a 2nd degree polynomial weighted using flux errors.
- **Interpolation** spatial interpolation on a regular grid and coaddition with IDL code griddata using radial basis function.

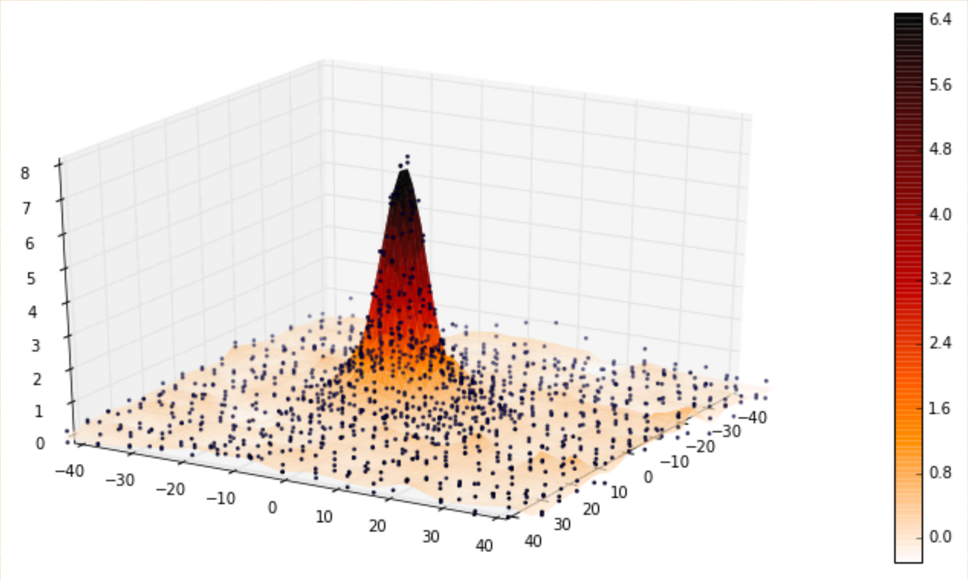
Fitting usually provides smoother images. Since the smoothing kernel is fixed, the result depends on the choice of the kernel and the weighting of the data. The interpolation, on the other hand, weights all the data in the same way.

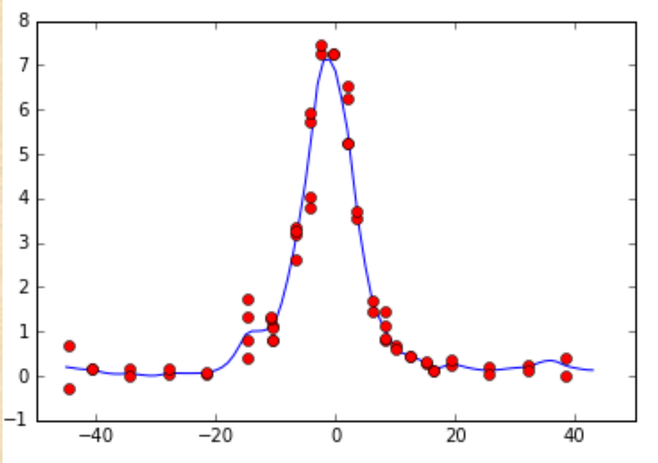
The default values for fitting are generally correct. However, in the case of very concentrated sources, the pipeline can oversmooth. We recently modified the pipeline (vers 1.3.2) to not propagate the response errors into the weights. This was causing the high flux points to be neglected since they had very low weights associated.

It is advisable to check your data against pre-projected data (WGR files) to know if the final flux estimate is reliable.

Interpolation (Python)





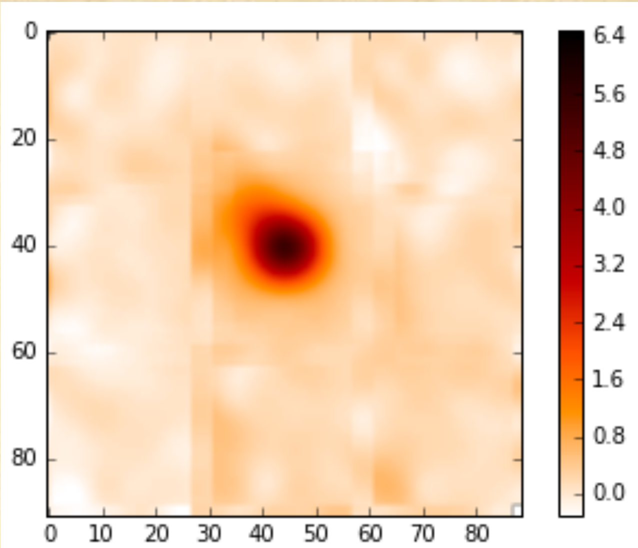


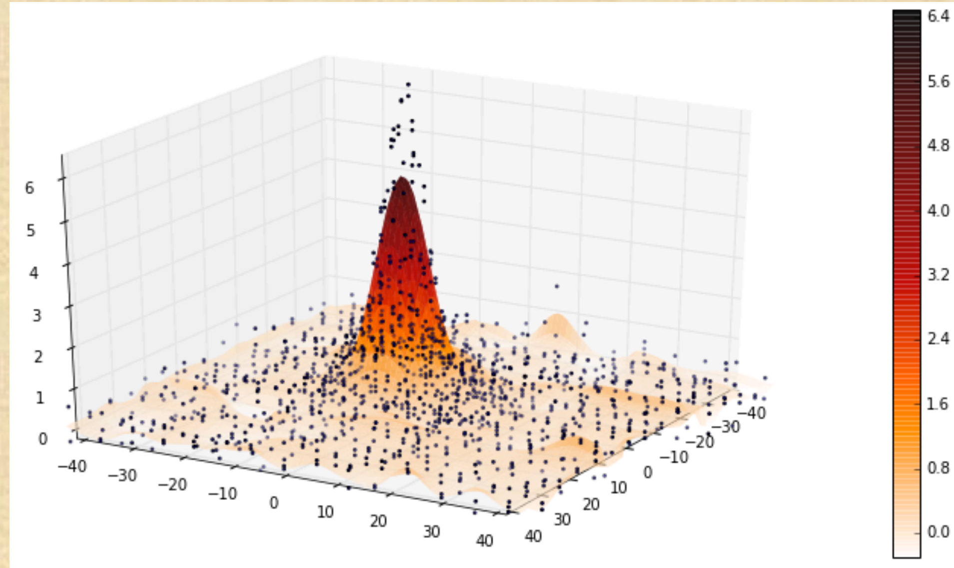
Resampling of WGR data (dots) using Python scipy libraries:

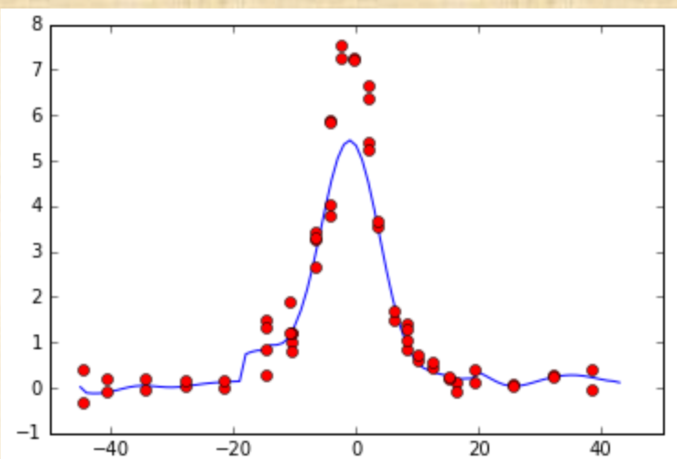
- scipy.interpolate.rbf (radial basis functions with minimal smoothing).

Slice at 88.5 um.
Total flux is: 1116

Interpolation (IDL)





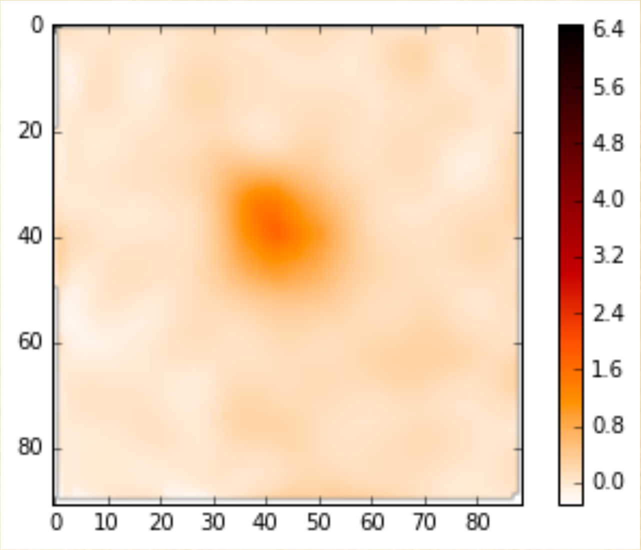


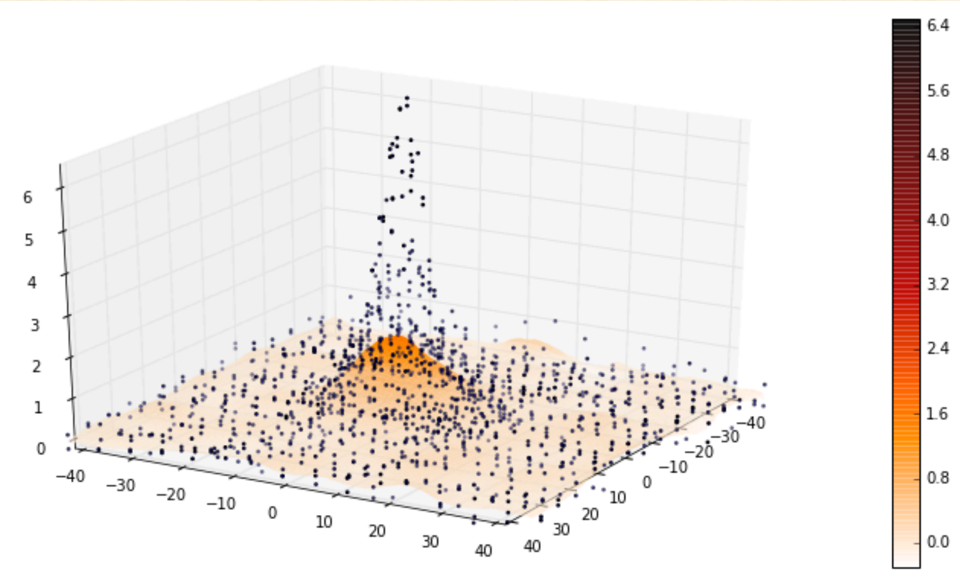
Resampling of WGR data (dots) using the FIFI-LS pipeline:

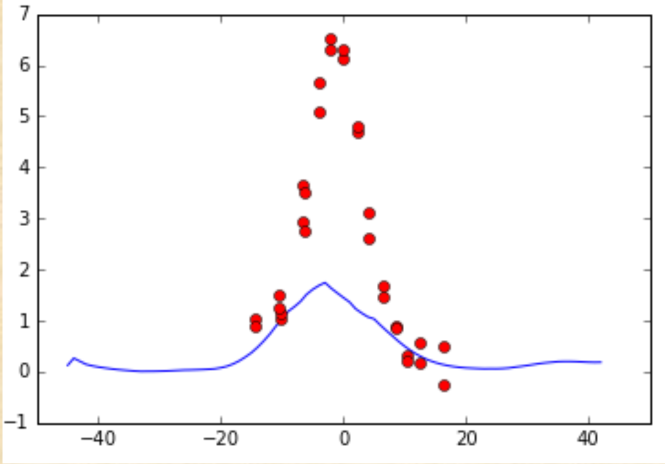
- IDL interpolation with radial basis functions

Slice at 88.5 um.
Total flux is: 1078

Polynomial fit 1.3.1 (default)





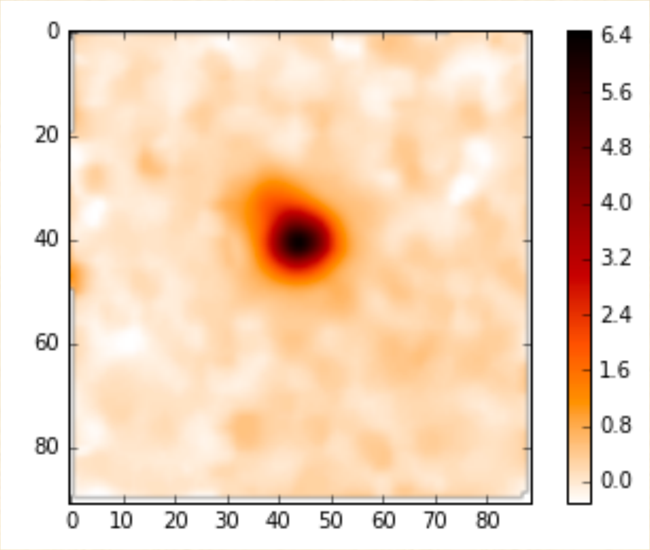


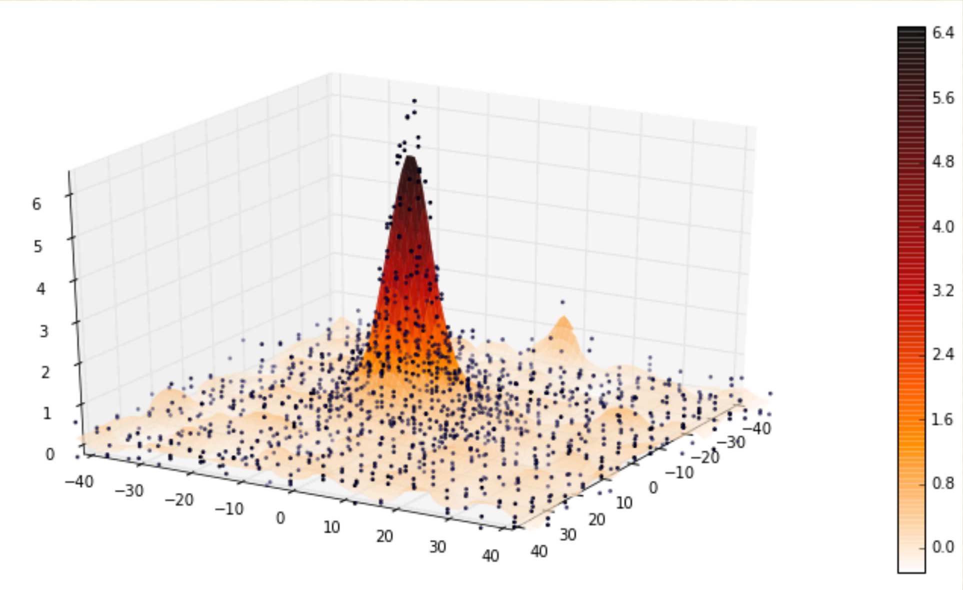
Interpolation of WGR data (dots) using the FIFI-LS pipeline 1.3.1:

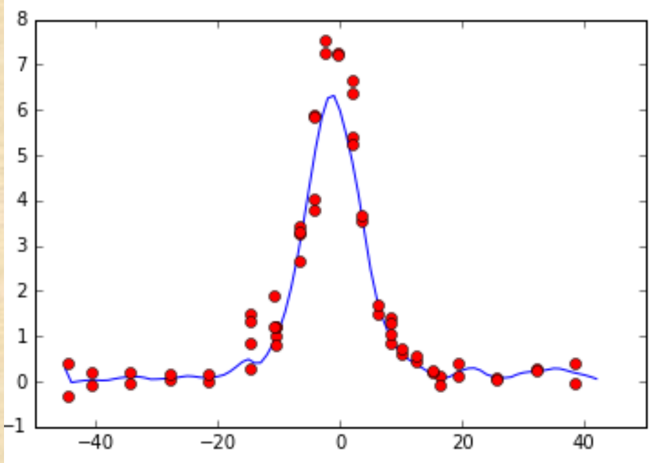
- polynomial 2D fit
- smoothing kernel = 2xFWHM (default)

Slice at 88.5 um.
Total flux is: 555

Polynomial fit (1.3.1 - manual)





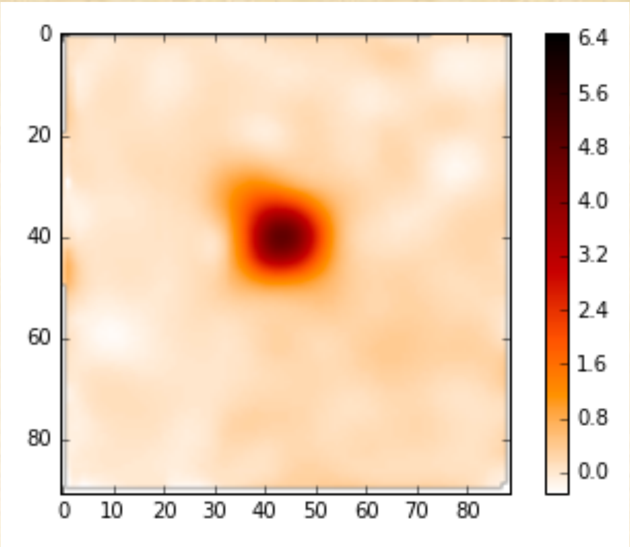


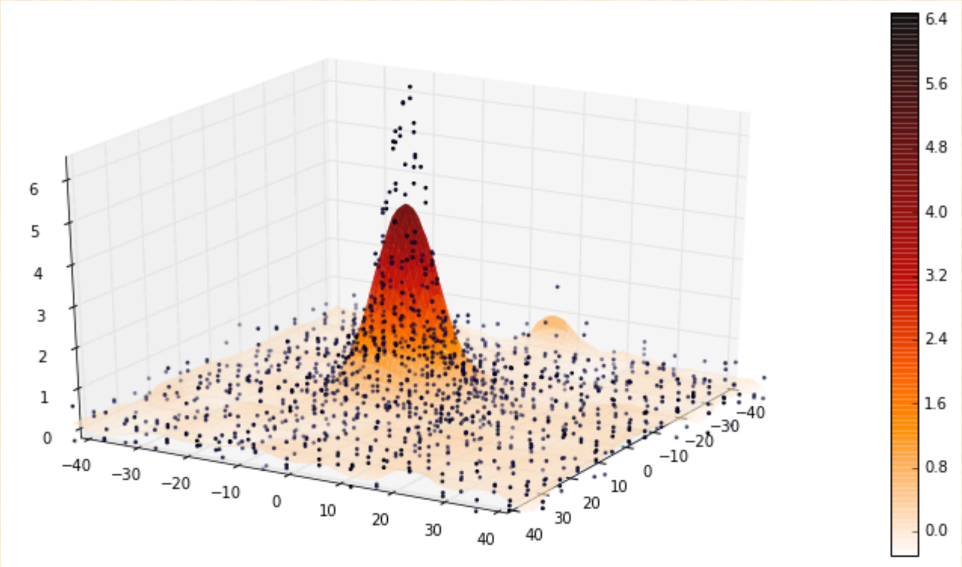
Resampling of WGR data (dots) using the FIFI-LS pipeline:

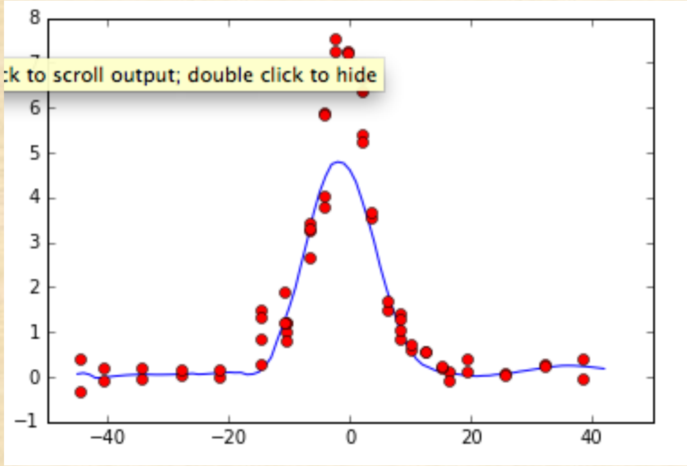
- polynomial 2D fit with
- smoothing kernel = 1 x FWHM

Slice at 88.5 um.
Total flux is: 1037

Polynomial fit 1.3.2 (default)





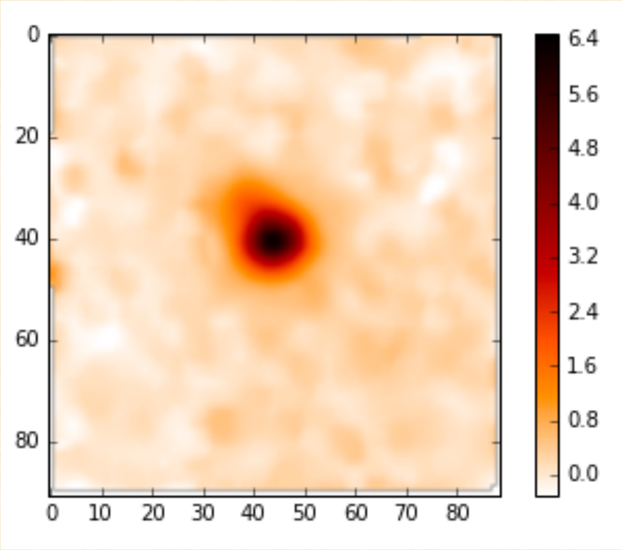


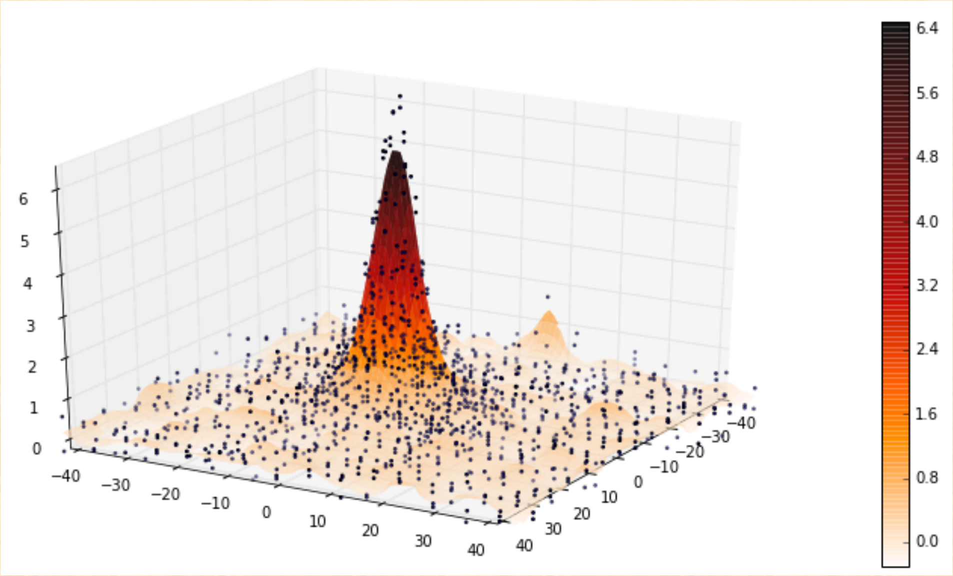
Resampling of WGR data (dots) using the FIFI-LS pipeline 1.3.2:

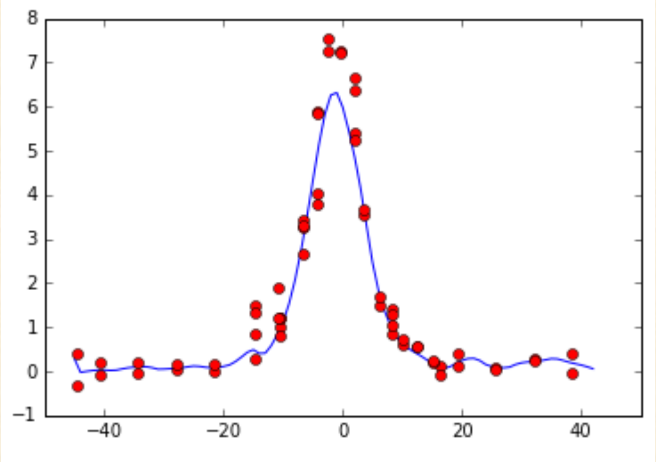
- polynomial 2D fit
- smoothing kernel = 2xFWHM (default)

Slice at 88.5 um.
Total flux is: 1043

Polynomial fit 1.3.2 (manual)





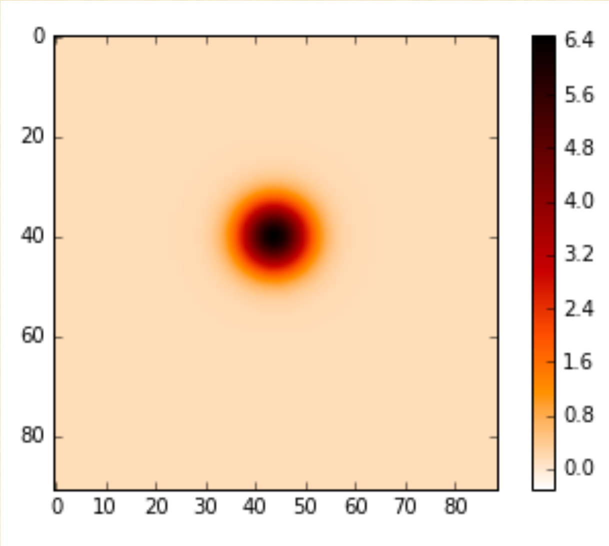


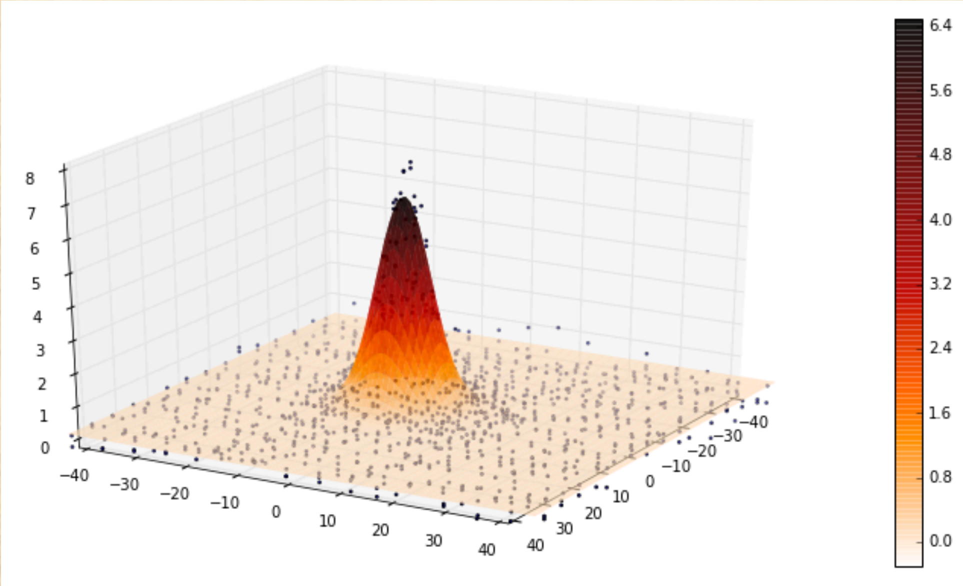
Resampling of WGR data (dots) using the FIFI-LS pipeline:

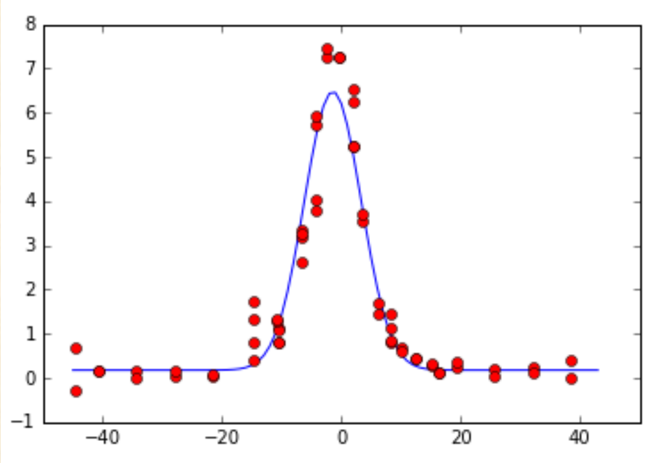
- polynomial 2D fit
- smoothing kernel = 1 x FWHM

Slice at 88.5 um.
Total flux is: 1040

2D Gaussian fit







2D Gaussian fit of WGR data (dots) in Python.

Slice at 88.5 um.
Total flux is: 889

Spatial resampling summary

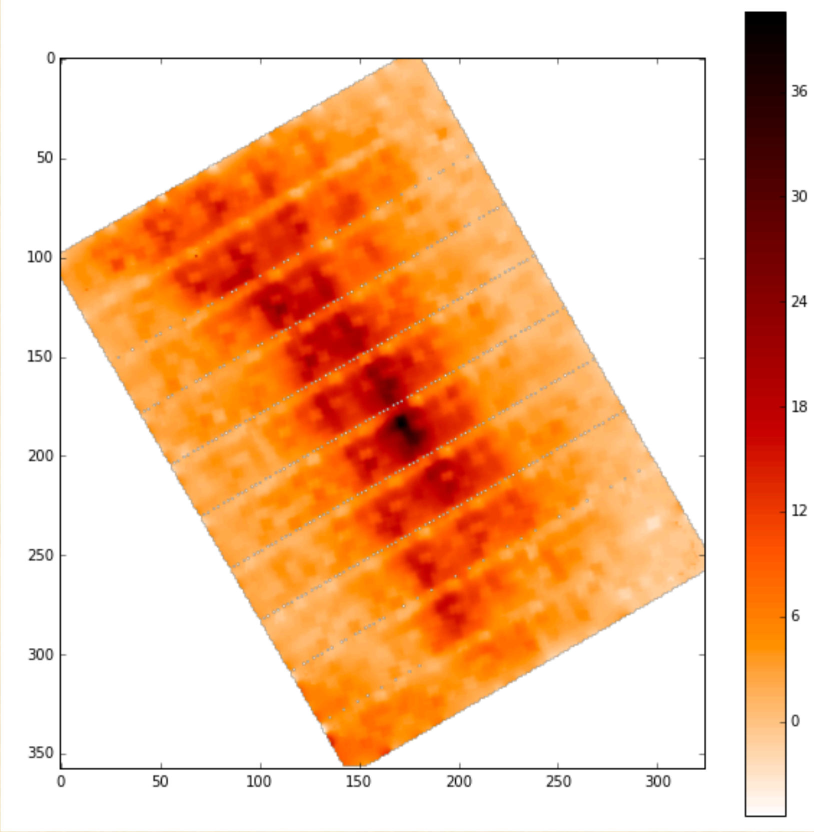
In this particular case we have seen that the default pipeline result could be significantly wrong.
In particular, if we normalize all the fluxes to the best fit done in Python with Radial Basis Functions, we have:

Pipeline (IDL) interpolation	**97.0%**
Pipeline 1.3.1 fit (default)	**50.0%**
Pipeline 1.3.1 fit (narrower kernel)	**93.3%**
Pipeline 1.3.2 fit (default)	**93.9%**
Pipeline 1.3.2 fit (narrower kernel)	**93.6%**
2D Gaussian fit	**80.0%**

In general the pipeline interpolation gives good flux estimates and it has been used for calibration.
Nice smooth maps are produced with polynomial fitting, although caution should be used for flux estimates in the case of peaked sources done with the pipeline 1.3.1.

Poorly dithered data

Simple interpolation is preferable is the case of data with very little dithering. There is very little advantage in smoothing data without redundancy. Since interpolation does only local smoothing, it is possible to better see defects in the reduction.

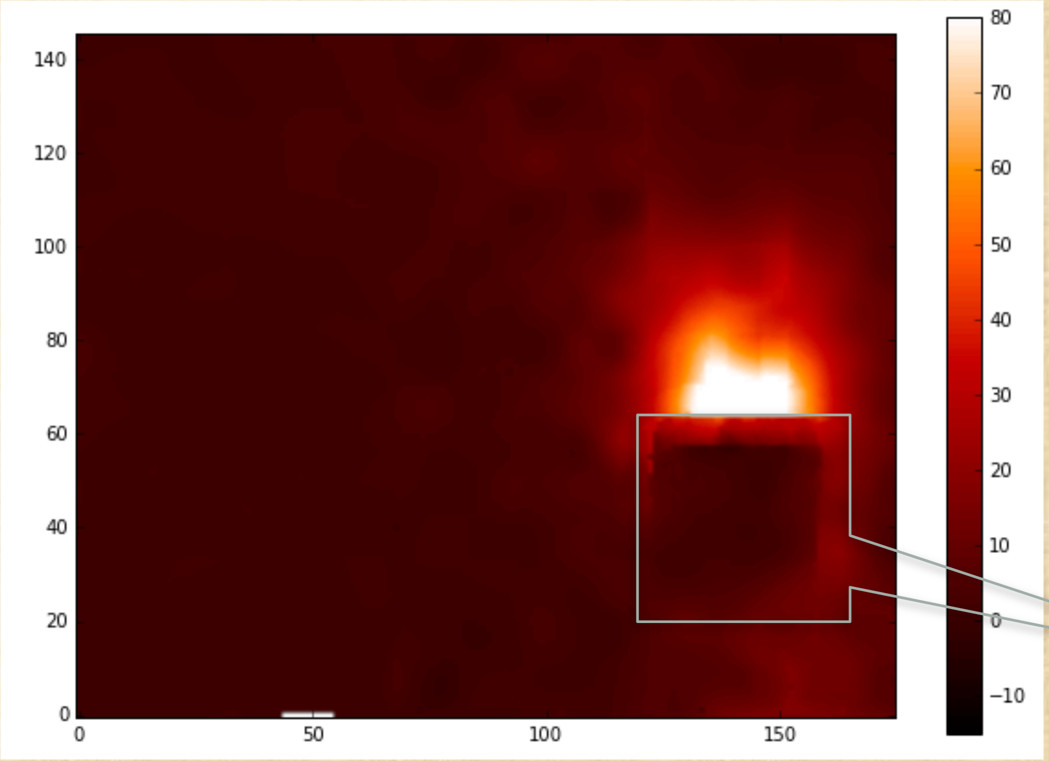


In this image we can see a raster scan with single redundancy reduced with interpolation.

Residuals from flats are clearly visible. Also, the image shows clearly that there is no overlap between parallel scans.

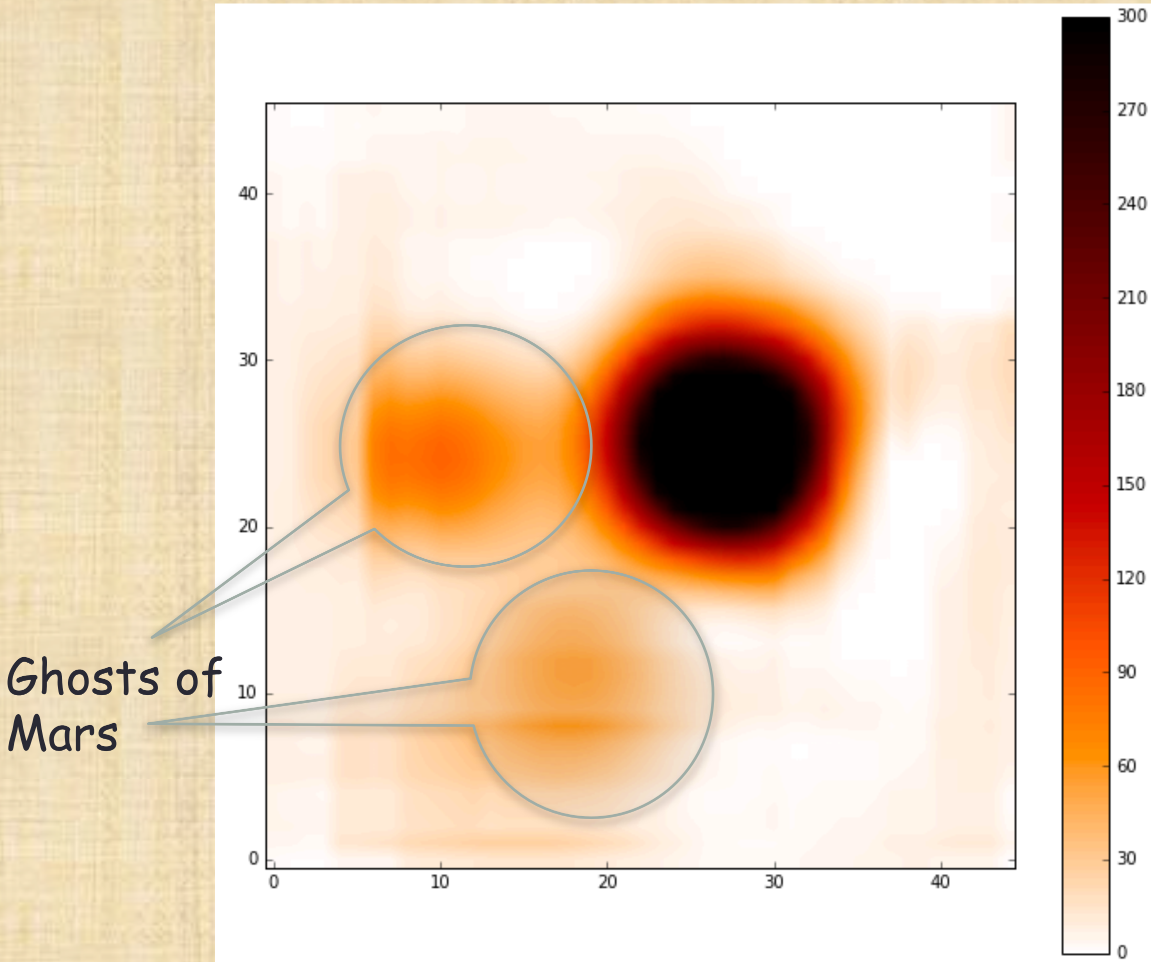
Bad chopping

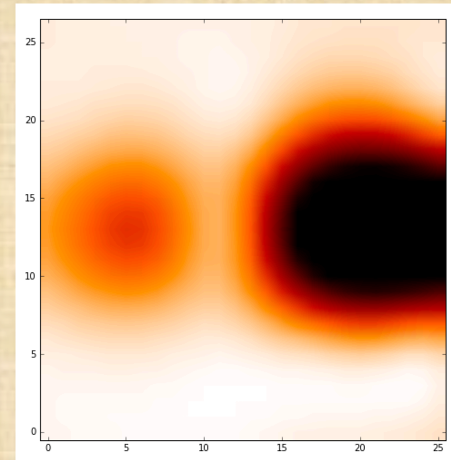
Negative values of the continuum indicate an unlucky choice of the reference chopping position.

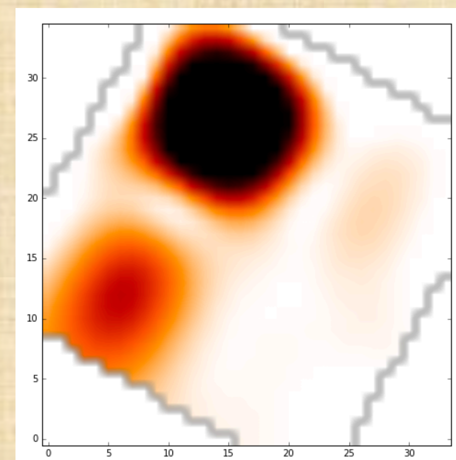


Bad chop here

Ghosts

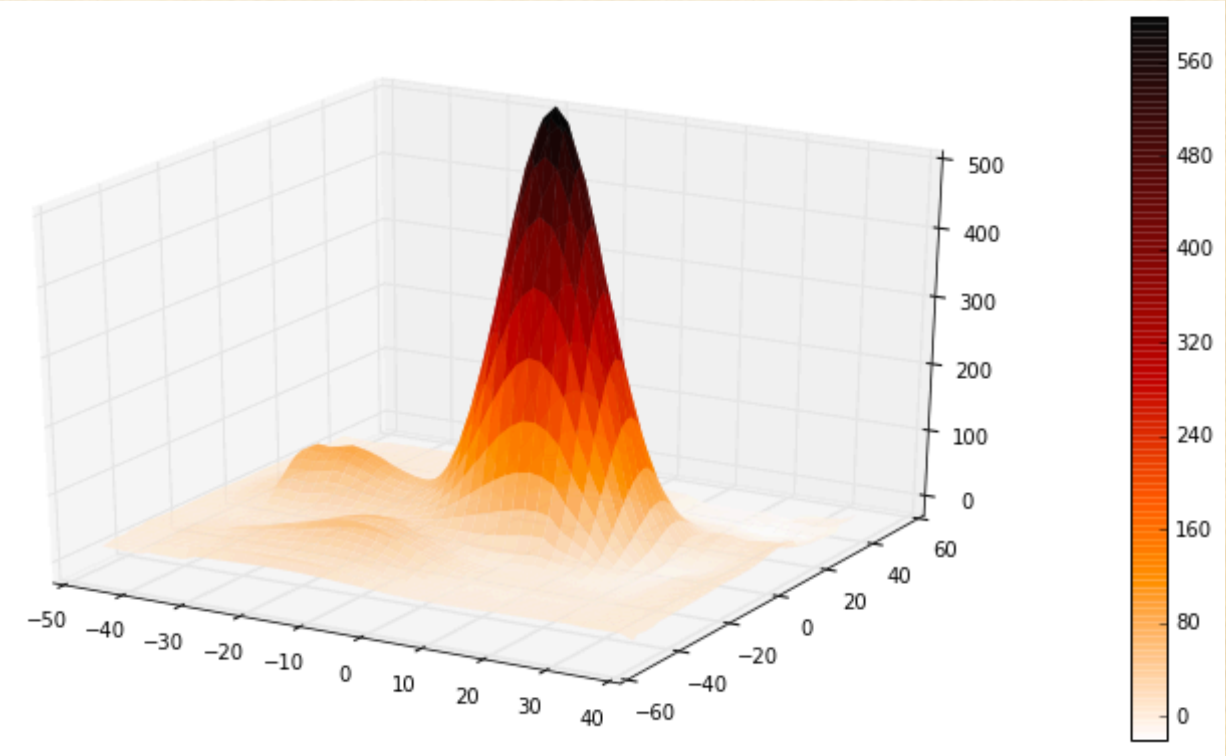






In presence of bright sources (Mars in this example at 130μm), ghosts can appear in the image. On the right, single pointings with the ghost.

Ghosts



In the single pointing the ghost accounts for 20% of the total flux. In the final image each ghost contribute approximately 10% of the total flux. Ghosts are taken into account for calibration.

What's next ?

- We just archived products from pipeline version 1.3.1 which includes flux calibration and telluric corrections
- We will reprocess the data starting next week with version 1.3.2 to get better spatial resampling
- We are currently analyzing a wealth of data taking during several flights with the scope of computing in-flight flats. For the moment we have produced:
  - A new set of bad pixel masks for each flight series
  - A study of the saturation point of the detectors
  - An estimate of non-linearity of the detectors
  - We are currently working on defining new flats. Since during the last flight series flats have sensibly changed, this will lead to a new reprocessing and data release early next year.