NIFS Data Reduction

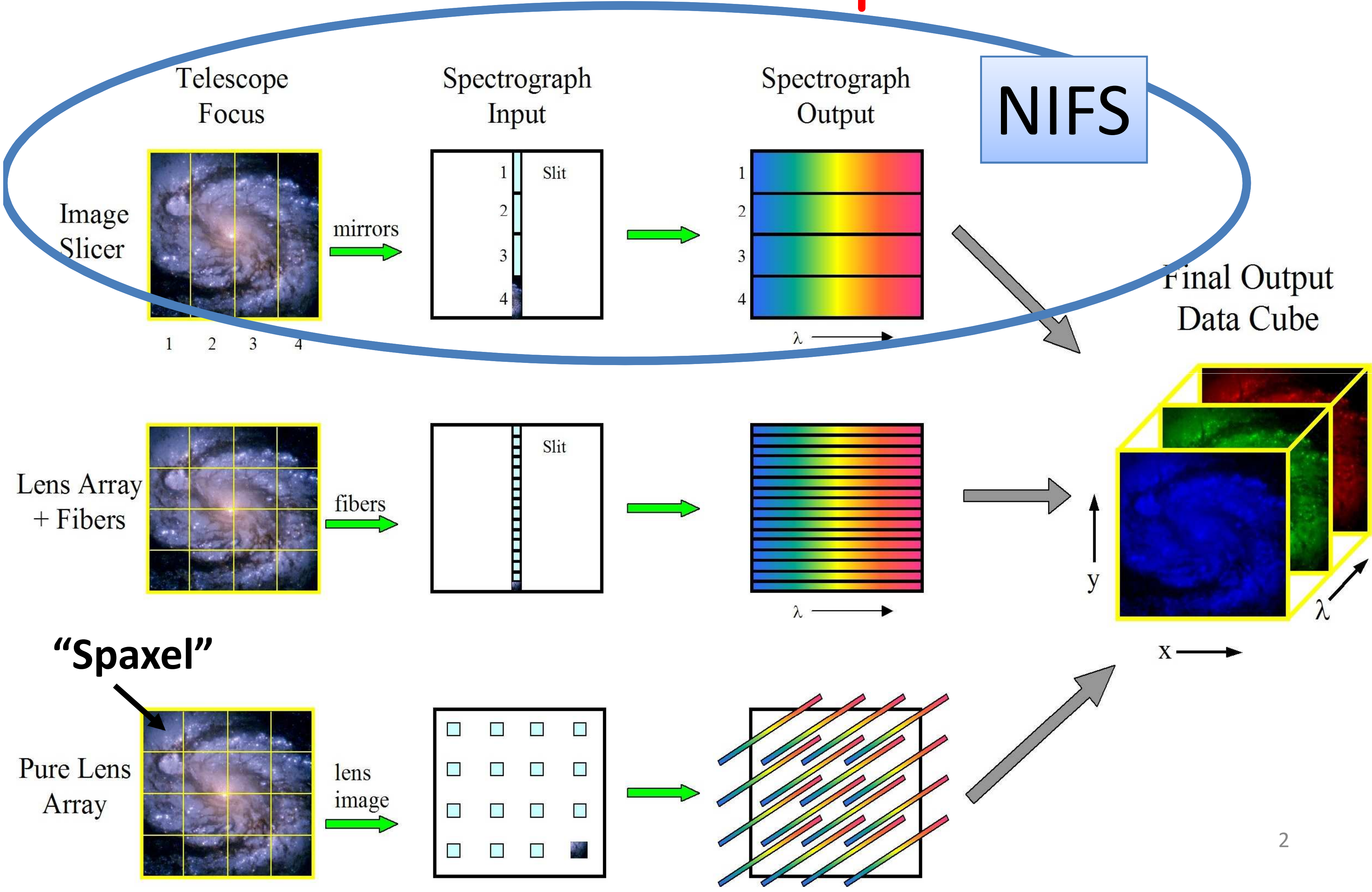
Richard McDermid

South American Gemini Data Workshop

São José dos Campos, Brazil,

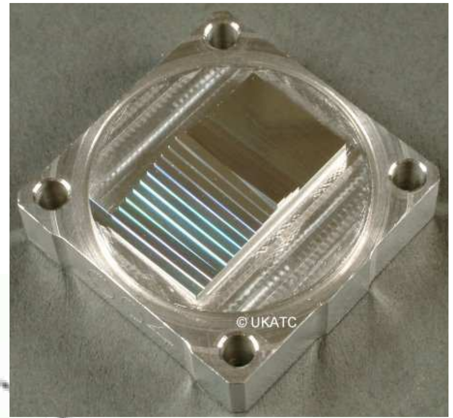
October 27-30, 2011

IFU Zoo: How to map 3D on 2D



IFU Techniques: Image Slicer

MIRI - JWST

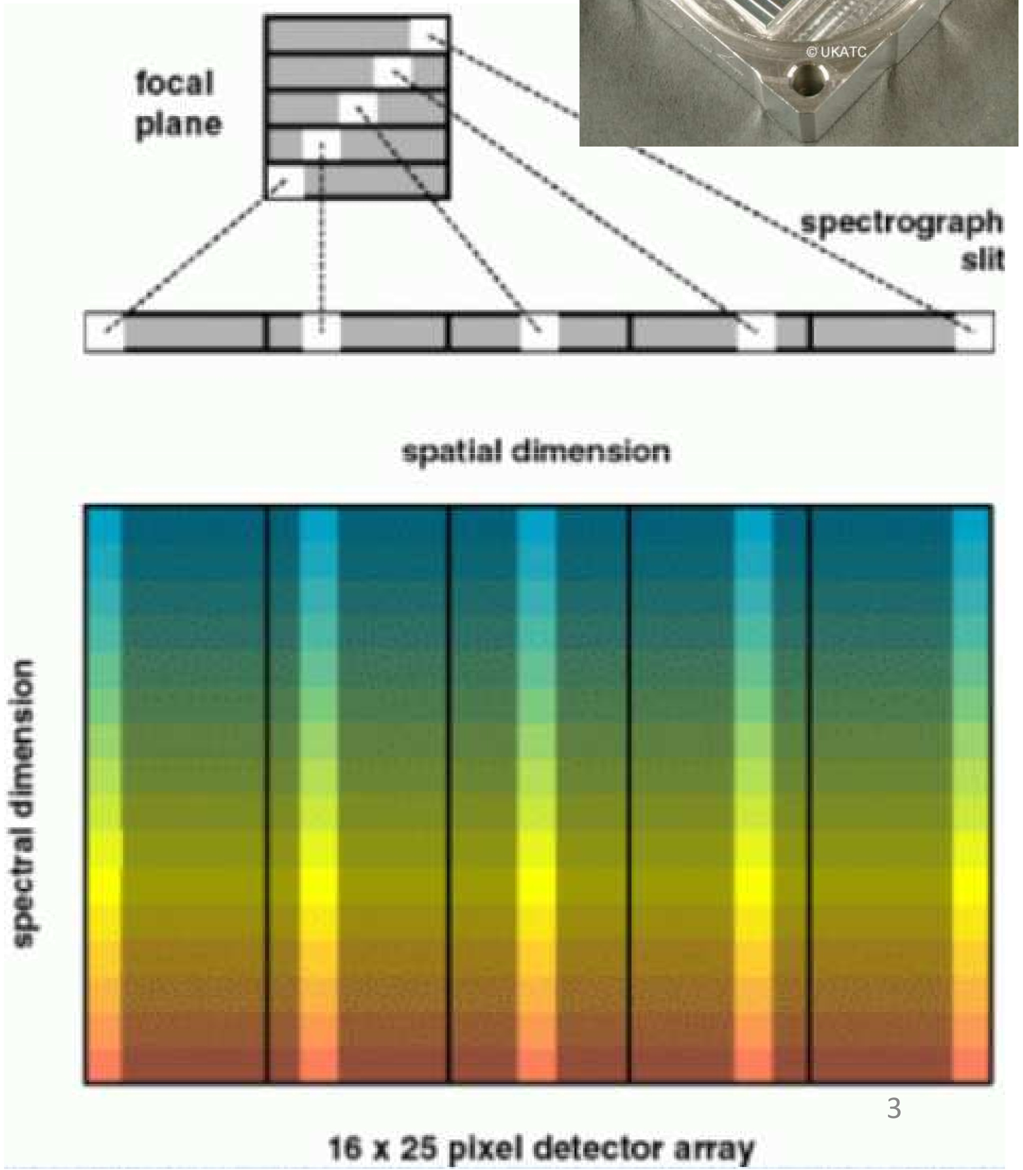


Pros:

- Compact design
- High throughput
- Easy cryogenics

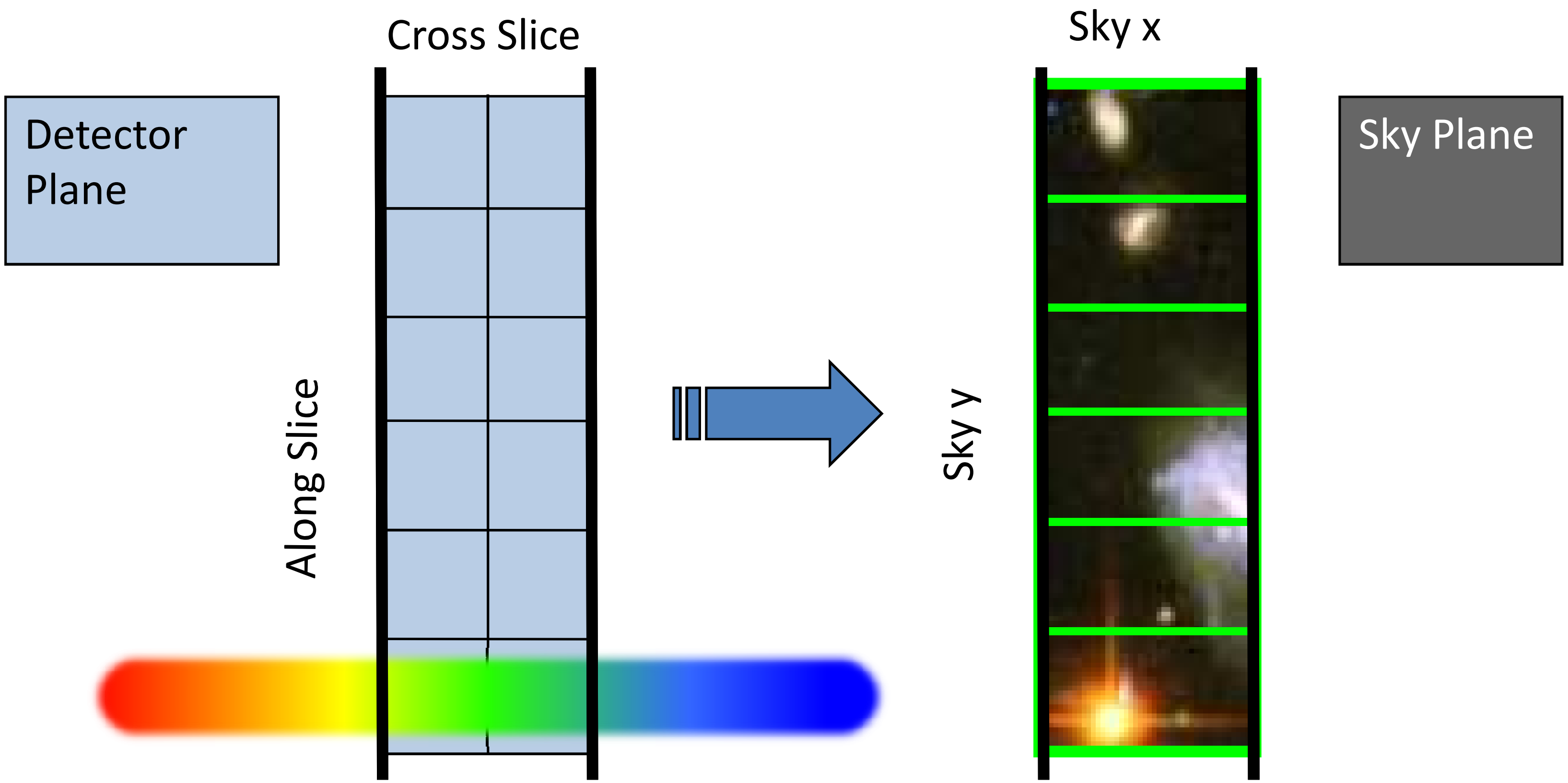
Cons:

- Difficult to manufacture



Rectangular Pixels

- NIFS has different (x,y) spatial sampling
- ***Along the slice*** is sampled by the ***detector***
- ***Across the slice*** is sampled by the ***slicer***
- Cross-slice sets spectral PSF - should be sampled on ~2 pixels
- Gives rectangular spaxels on the sky



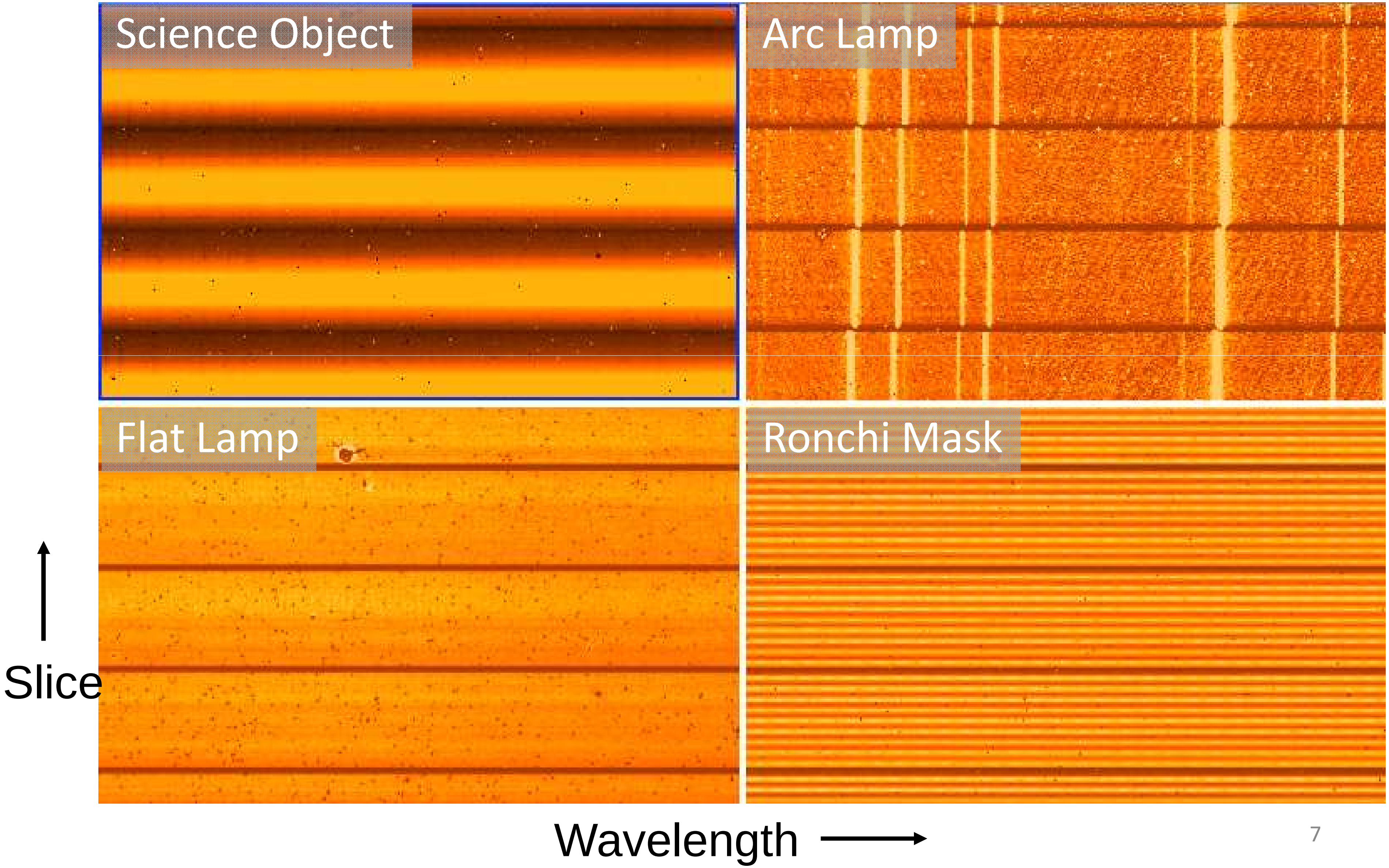
NIFS

- Near-infrared Integral Field Spectrograph
- Cryogenic slicer design
- Z,J,H,K bands, R~5,000
- One spatial setting:
  - 3”x3” FoV
  - 0.1”x0.04” sampling
- Optimized for use with AO
- Science: young stars, exo-planets, solar system, black holes, jets, stellar populations, hi-z galaxies….

Typical NIFS Observation

- ‘Before’ telluric star
  - NGS-AO
  - Acquire star
  - Sequence of on/off exposures
  - Same instrument config as science (inc. e.g. field lens for LGS)
- Science observation
  - Acquisition
  - Observation sequence:
    - Arc (grating position is not 100% repeatable)
    - Sequence of on/off exposures
- ‘After’ telluric (if science >~1.5hr)
- Daytime calibrations:
  - Baseline set:
    - Flat-lamps (with darks)
    - ‘Ronchi mask’ flats (with dark)
    - Darks for the arc
  - Darks for science (if sky emission to be used for wavelength calibration)

Typical NIFS Data



Arranging your files - suggestion

- Daycals/ - All baseline daytime calibrations
  - YYYYMMDD/ - cals from different dates
- Science/ - All science data
  - Obj1/ - First science object
    - YYYYMMDD/ - First obs date (if split over >1 nights)
      - Config/ - e.g. 'K' (if using multiple configs)
        - Telluric/ - telluric data for this science obs
    - Merged/ - Merged science and subsequent analysis
- Scripts/

NIFS Reduction: Example scripts

- Three IRAF scripts on the web:
  - Calibrations
  - Telluric
  - Science
- Form the basis of this tutorial
- Data set:
  - Science object (star)
  - Telluric correction star
  - Daytime calibrations
- Update the path and file numbers at the top of each script
- Excellent starting point for basic reduction

Lamp Calibrations

- Three basic calibrations:
  - Flat (DAYCAL)
    - Correct for transmission and illumination
    - Locate the spectra on the detector
  - Ronchi Mask (DAYCAL)
    - Spatial distortion
  - Arc (NIGHTCAL)
    - Wavelength calibration
- Each has associated dark frames
- May have multiple exposures to co-add
- DAYCAL are approx. 1 per observation date
- NIGHTCAL are usually once per science target, but can be common between targets if grating config not changed

Calibration 1: Flat-Field

- Step 1: Locate the spectra
  - Mask Definition File (MDF) provides relative location of slices on detector
  - Use nfprepare to match this to the absolute position for your data:

Input file

Path to data

Prefix for new output file

X-shift for MDF

```
nfprepare(calflat,rawpath=raw_data, outpref="s", shiftx=INDEF,
        shifty=INDEF, fl_vardq-, fl_corr-, fl_nonl-)
```

Y-shift for MDF

Do not create a variance extension

Do not correct for non-linearity

Do not try to flag non-linear pixels

  - Offset is stored in a new image
  - This exposure is then referenced in subsequent steps that need to know where the spectra are on the chip

Calibration 1: Flat-Field

- Step 2.1: Update flat images with offset value
- Step 2.2: Generate variance and data quality extensions
- Nfprepare is called again (once) to do both these tasks:

Input file list

Reference image with shift

```
nfprepare("@flatlist”, rawpath=raw_data, shiftim="s"//calflat,
   fl_vardq+, fl_int+, fl_corr-, fl_nonl-)
```

Create variance and data quality planes

Run interactively

- Apply same process to dark frames

Calibration 1: Flat-Field

- Step 2.3: Combine flats and darks using gemcombine:

Input file list

```
gemcombine("n//@flatlist",output="gn"//calflat,
   fl_dqpr+, fl_vardq+, masktype="none", logfile="nifs.log”)
```

Propagate DQ

Generate VAR/DQ planes

No pixel masking

Append outputs to a log file

- Repeat for darks…
- Now have 2D images with DQ and VAR extensions. Ready to go to 3D…

Calibration 1: Flat-Field

- Step 3.1: Extract the slices using nsreduce:

'cut' out the slices from the 2D image

Apply first order wavelength coordinate system

```
nsreduce("gn"//calflat, fl_nscut+, fl_nsappw+, fl_vardq+,
   fl_sky-, fl_dark-, fl_flat-, logfile="nifs.log”)
```

- Step 3.2: Create slice-by-slice flat field using nsflat:

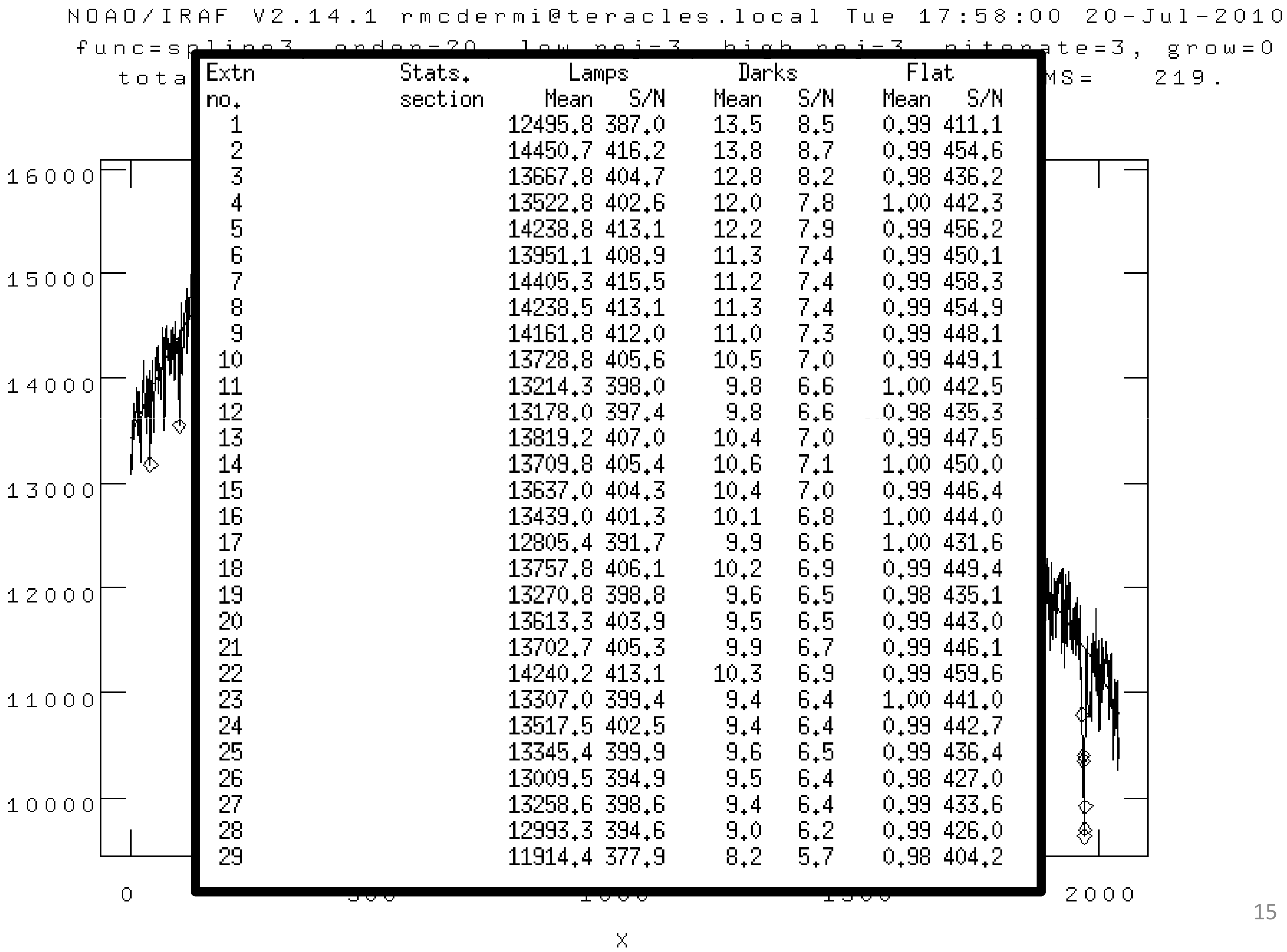
```
nsflat("rgn"//calflat, darks="rgn"//flatdark,
 flatfile="rn"//calflat//"_sflat”, darkfile="rn"//flatdark//"_dark",
 fl_save_dark+, process="fit”, thr_flo=0.15, thr_fup=1.55,
 fl_vardq+, logfile="nifs.log")
```

Output flat image

LOwer and UPper limits for 'bad' pixels

  - Divides each spectrum (row) in a slice by a fit to the average slice spectrum, with coarse renormalizing
  - Also creates a bad pixel mask from the darks

Calibration 1: Flat-Field



Extn no.	Stats. section	Lamps Mean	Lamps S/N	Darks Mean	Darks S/N	Flat Mean	Flat S/N
1		12495.8	387.0	13.5	8.5	0.99	411.1
2		14450.7	416.2	13.8	8.7	0.99	454.6
3		13667.8	404.7	12.8	8.2	0.98	436.2
4		13522.8	402.6	12.0	7.8	1.00	442.3
5		14238.8	413.1	12.2	7.9	0.99	456.2
6		13951.1	408.9	11.3	7.4	0.99	450.1
7		14405.3	415.5	11.2	7.4	0.99	458.3
8		14238.5	413.1	11.3	7.4	0.99	454.9
9		14161.8	412.0	11.0	7.3	0.99	448.1
10		13728.8	405.6	10.5	7.0	0.99	449.1
11		13214.3	398.0	9.8	6.6	1.00	442.5
12		13178.0	397.4	9.8	6.6	0.98	435.3
13		13819.2	407.0	10.4	7.0	0.99	447.5
14		13709.8	405.4	10.6	7.1	1.00	450.0
15		13637.0	404.3	10.4	7.0	0.99	446.4
16		13439.0	401.3	10.1	6.8	1.00	444.0
17		12805.4	391.7	9.9	6.6	1.00	431.6
18		13757.8	406.1	10.2	6.9	0.99	449.4
19		13270.8	398.8	9.6	6.5	0.98	435.1
20		13613.3	403.9	9.5	6.5	0.99	443.0
21		13702.7	405.3	9.9	6.7	0.99	446.1
22		14240.2	413.1	10.3	6.9	0.99	459.6
23		13307.0	399.4	9.4	6.4	1.00	441.0
24		13517.5	402.5	9.4	6.4	0.99	442.7
25		13345.4	399.9	9.6	6.5	0.99	436.4
26		13009.5	394.9	9.5	6.4	0.98	427.0
27		13258.6	398.6	9.4	6.4	0.99	433.6
28		12993.3	394.6	9.0	6.2	0.99	426.0
29		11914.4	377.9	8.2	5.7	0.98	404.2

Calibration 1: Flat-Field

- Step 3.3: Renormalize the slices to account for slice-to-slice variations using nsslitfunction:

Final flat-field correction frame

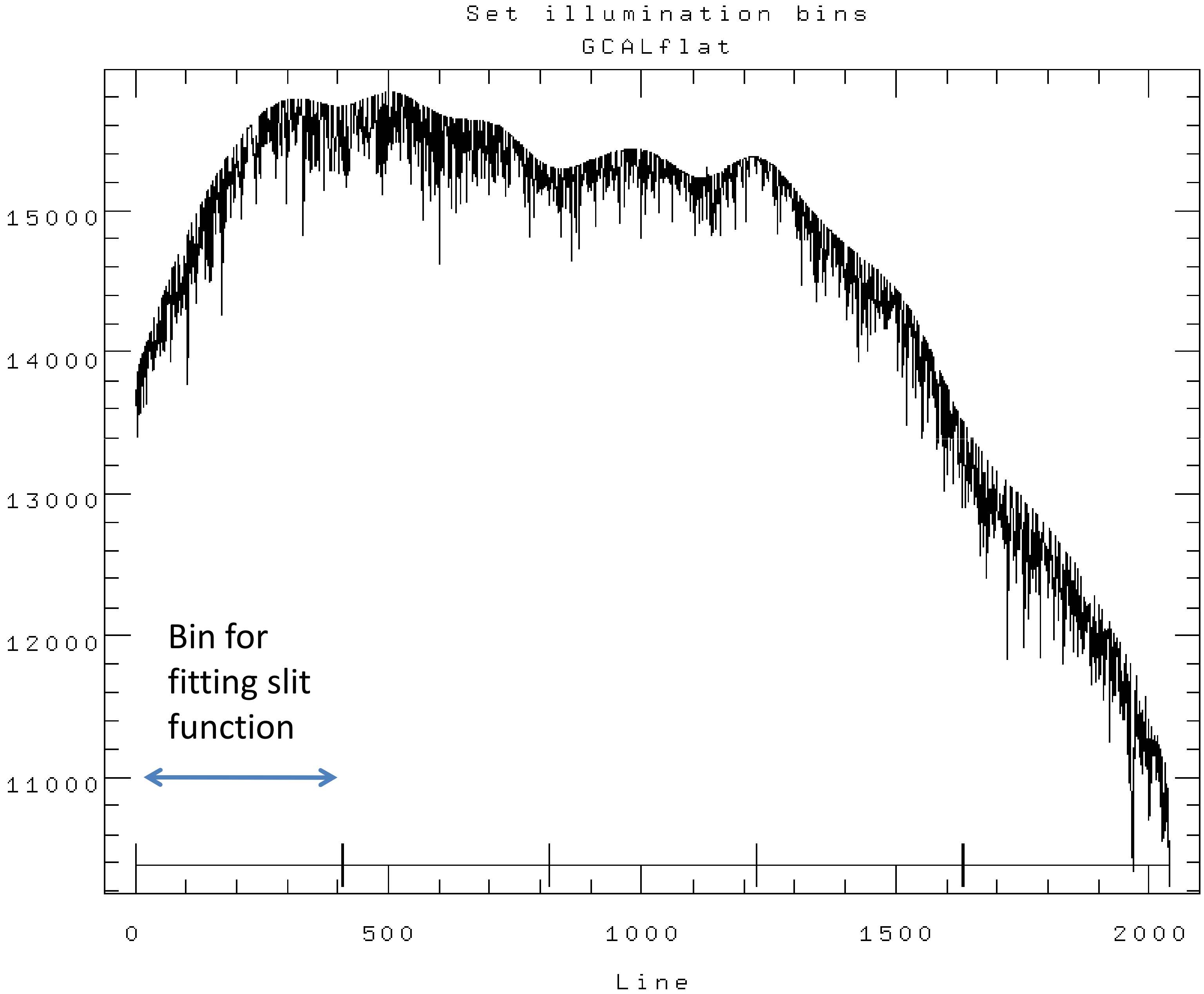
```
nsslitfunction("rgn"//calflat, "rn"//calflat//"_flat",
   flat="rn"//calflat//"_sflat”, dark="rn"//flatdark//"_dark",
   combine="median”, order=3, fl_vary-, logfile="nifs.log”)
```

Method to collapse in spectral direction

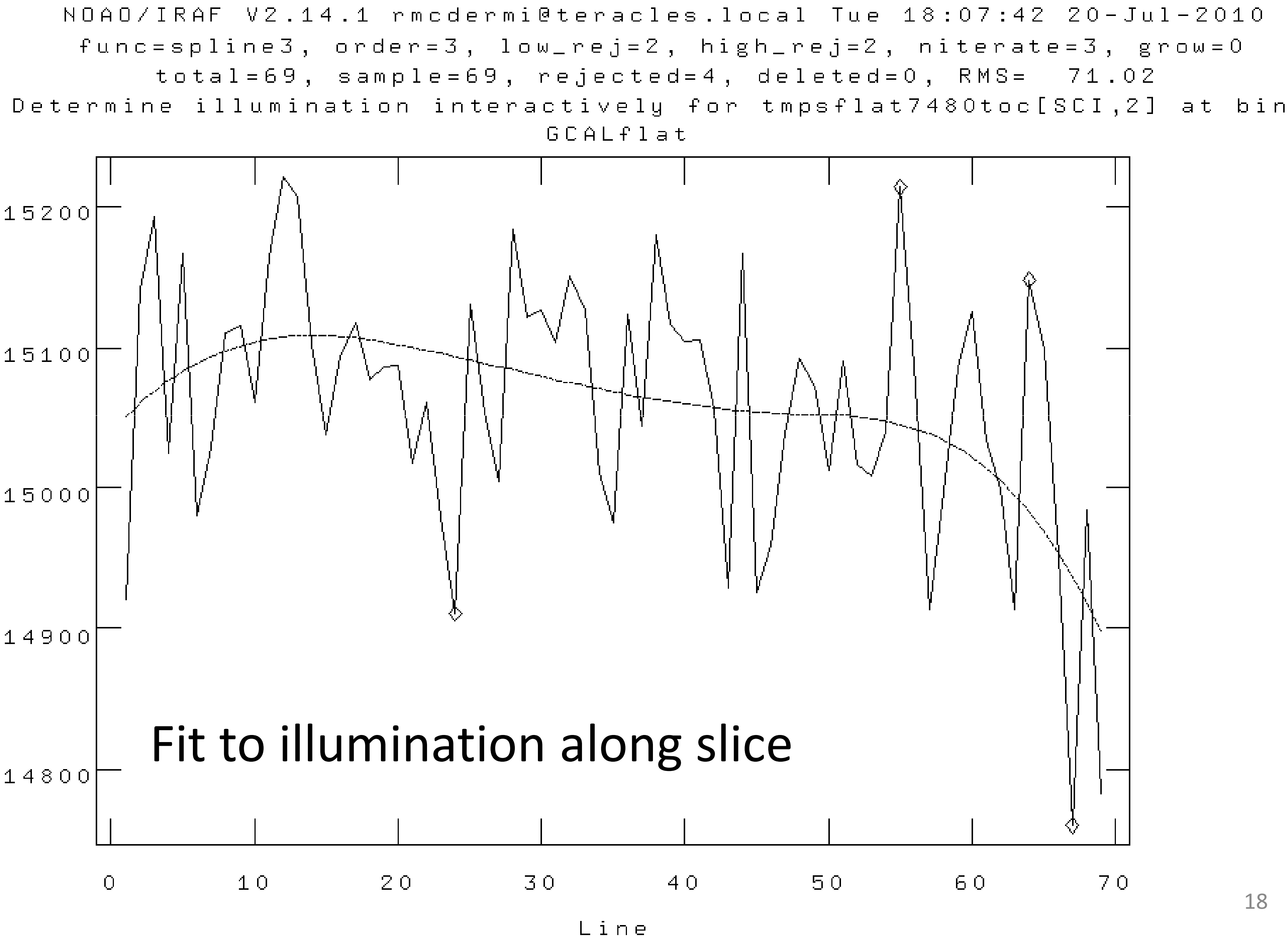
Order of fit across slices

  - Fits a function in spatial direction to set slice normalization
  - Outputs the final flat field, with both spatial and spectral flat information

Calibration 1: Flat-Field



Calibration 1: Flat-Field

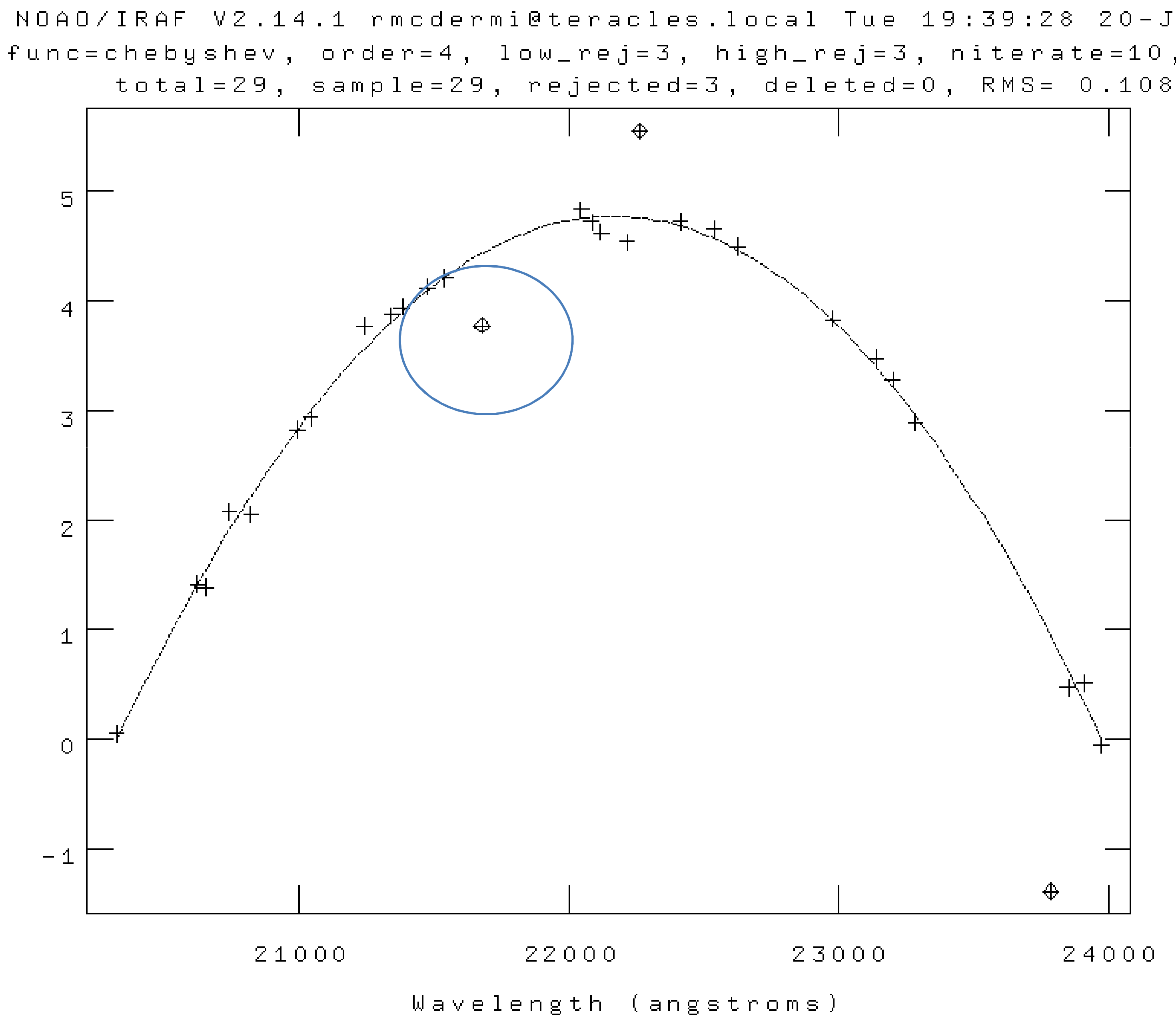


Calibration 2: Wavelength Calibration

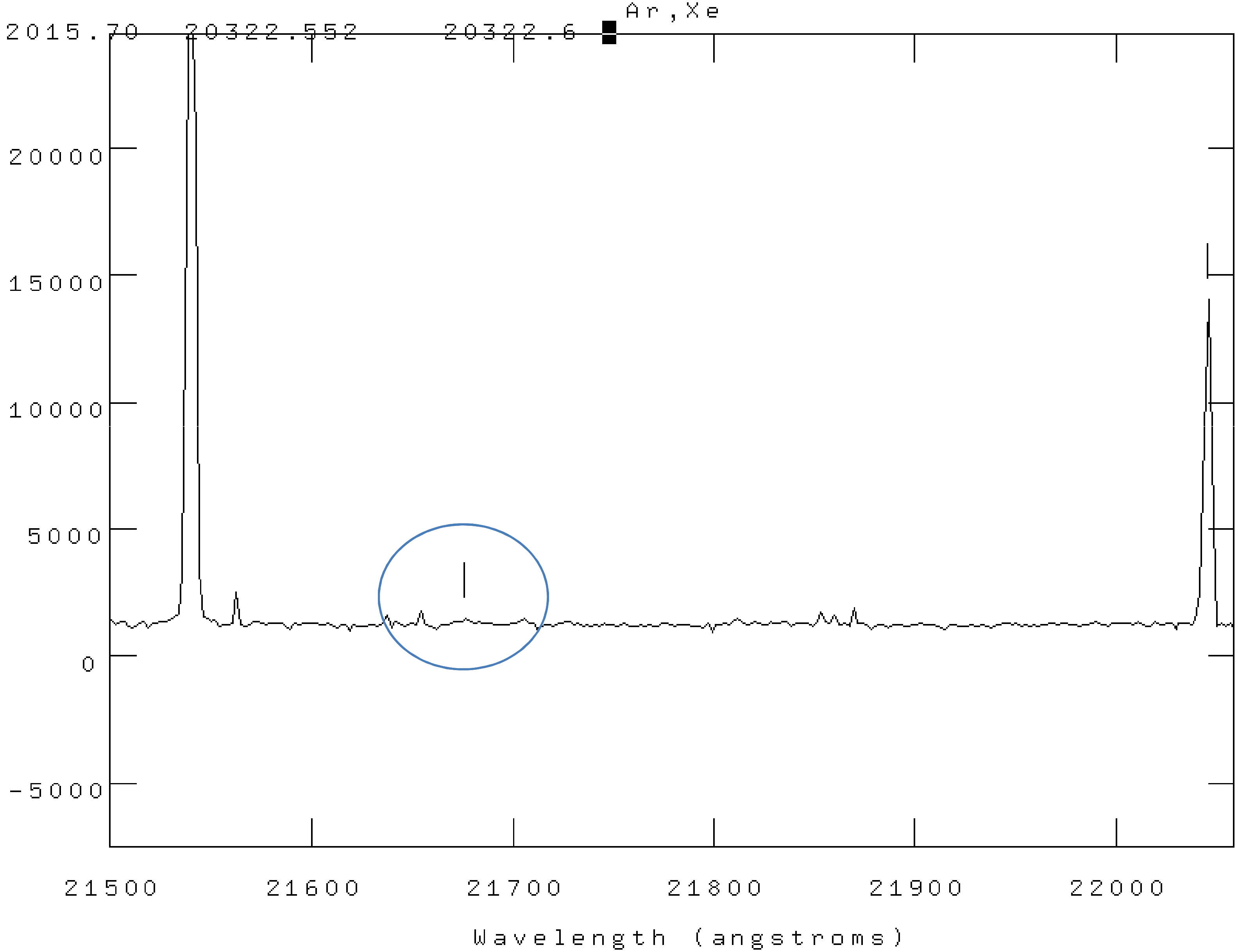
- Step 1: Repeat nfprepare, gemcombine and nsreduce -> extracted slices
- Step 2: Correctly identify the arc lines, and determine the dispersion function for each slice
  - Should run this interactively the first time through to ensure correct identification of lines and appropriate fit function
  - First solution is starting point for subsequent fits
  - Should robustly determine good solution for subsequent spectra
- Result is a series of files in a ‘database/’ directory containing the wavelength solutions of each slice

```
nswavelength("rgn"//arc, coordli=clist, nsum=10,
thresho=my_thresh, trace=yes, fwidth=2.0, match=-6, cradius=8.0,
fl_inter+, nfound=10, nlost=10, logfile="nifs.log”)
```

Calibration 2: Wavelength Calibration



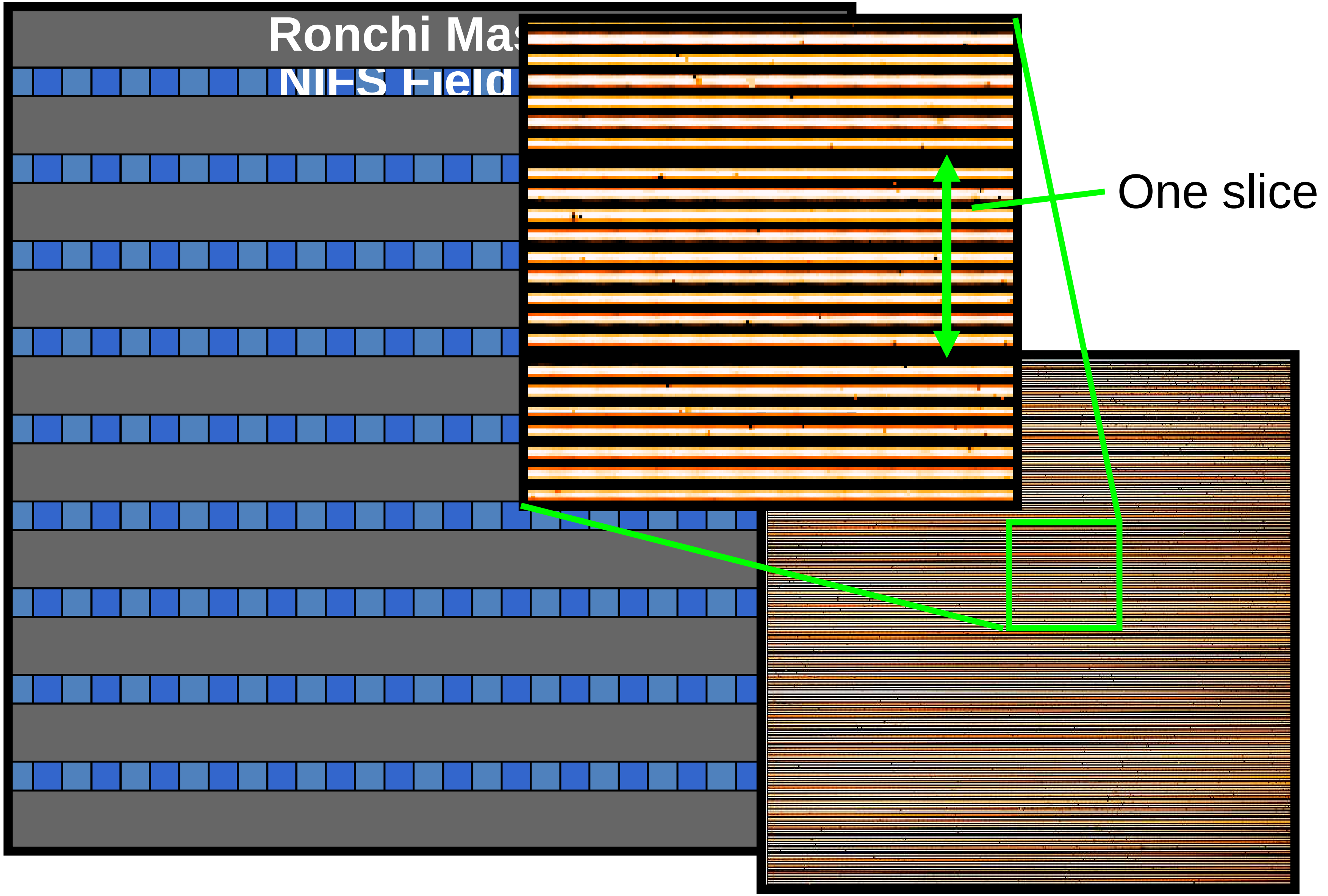
Calibration 2: Wavelength Calibration



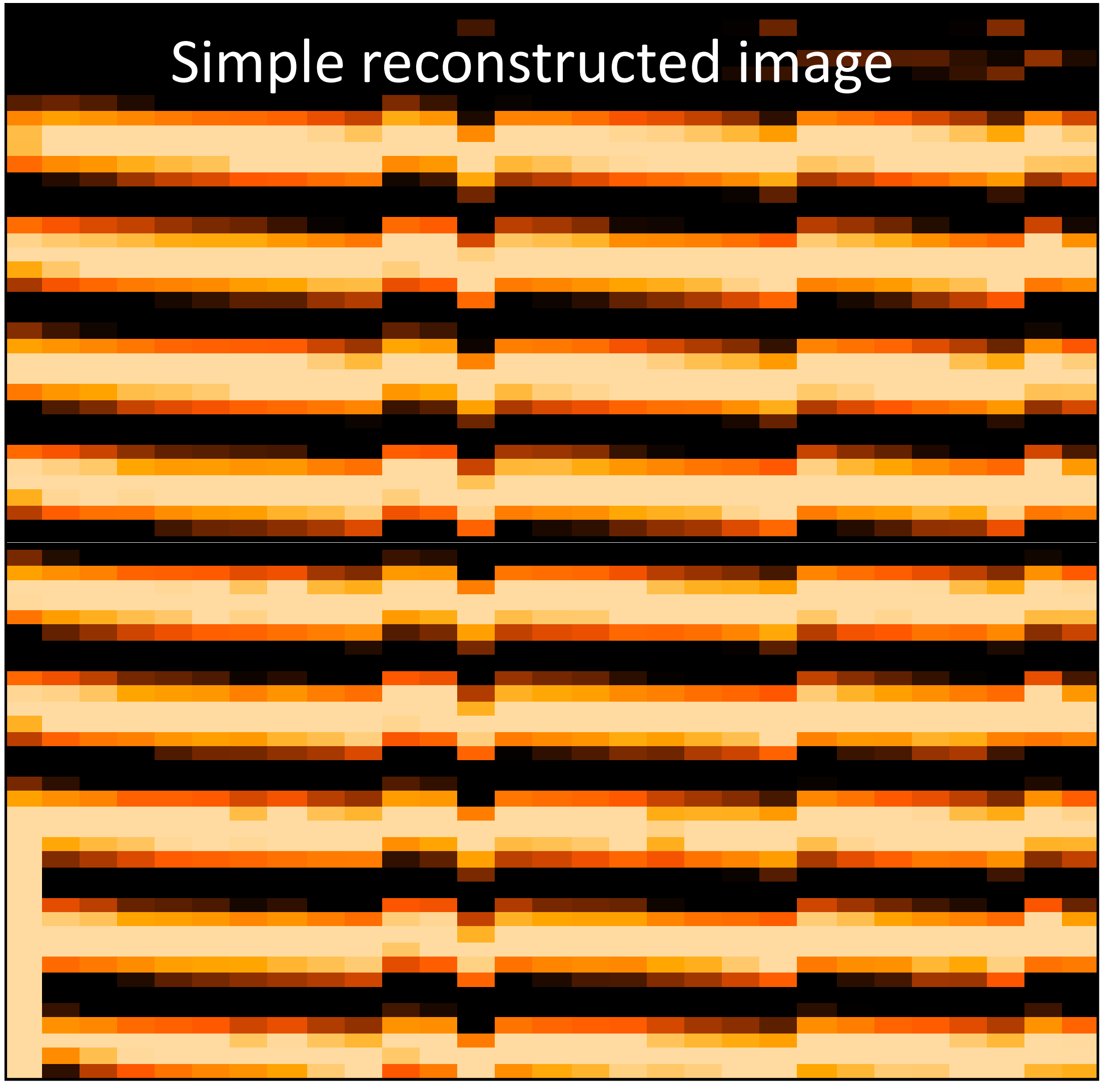
Calibration 3: Spatial Distortion

- Need to correct for distortions along the slices, and registration between slices
- This is done using the Ronchi mask as a reference
- Analogous to wavelength calibration, but in spatial domain

NIFS: Ronchi Mask



NIFS: Ronchi Mask



Transformation to make lines straight gives geometric correction

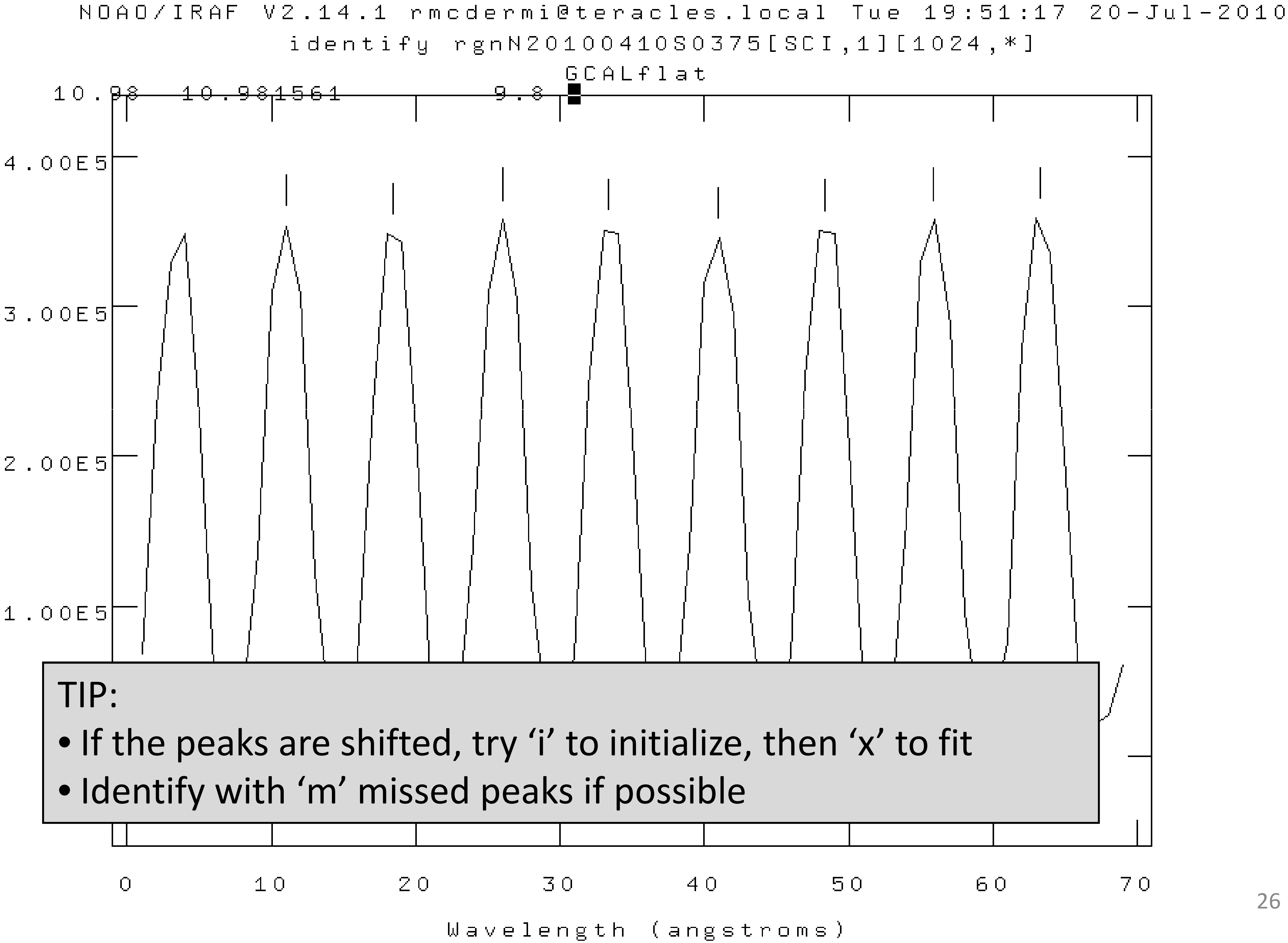
Calibration 3: Spatial Distortion

- Step 1: Repeat nfprepare, gemcombine and nsreduce -> extracted slices
- Step 2: run nfsdist
  - Reference peaks are very regular, so easy to fall foul of aliasing when run automatically
  - Recommend running interactively for each daycal set

```
nfsdist("rgn"//ronchiflat, fwidth=6.0, cradius=8.0, glshift=2.8,
minsep=6.5, thresh=2000.0, nlost=3, fl_int+, logfile="nifs.log”)
```

- TIP: apply the distortion correction to the Ronchi frame itself, and check its OK

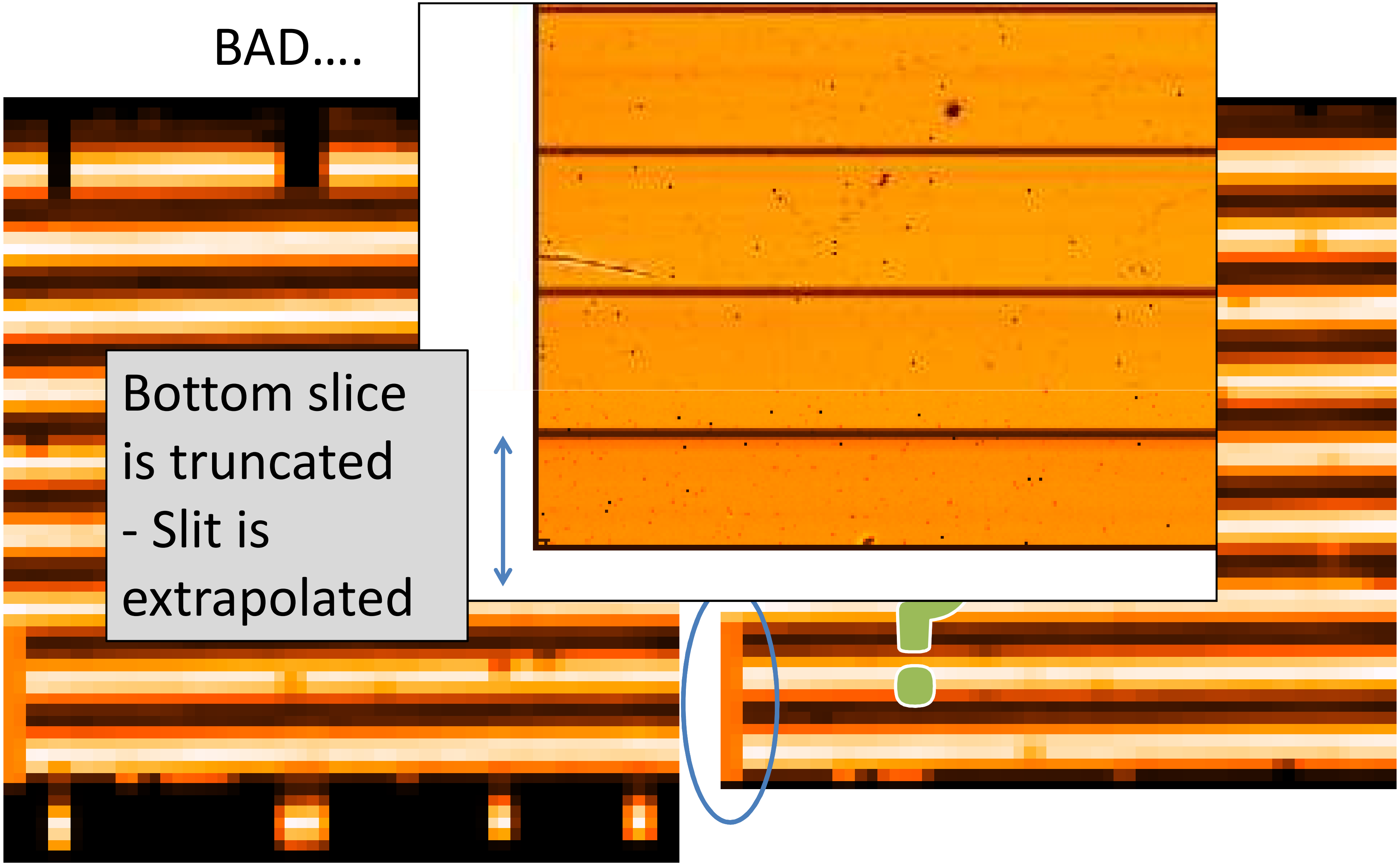
Calibration 3: Spatial Distortion



TIP:

- If the peaks are shifted, try ‘i’ to initialize, then ‘x’ to fit
- Identify with ‘m’ missed peaks if possible

Calibration 3: Spatial Distortion



Lamp Calibrations: Summary

You now have:

1. Shift reference file: "s"+calflat
2. Flat field: "rn"+calflat+"_flat"
3. Flat BPM (for DQ plane generation): "rn"+calflat+"_flat_bpm.pl”
4. Wavelength referenced Arc: "wrn"+arc
5. Spatially referenced Ronchi Flat: "rn"+ronchiflat

Notes:

- 1-3 are files that you need
- 4 & 5 are files with associated files in the ‘database/’ dir
- Arcs are likely together with science data

Telluric Star

- Similar to science reduction up to a point:
  - Sky subtraction
  - Spectra extraction => 3D
  - Wavelength calibration
  - Flat fielding
- Then extract 1D spectra, co-add separate observations, and derive the telluric correction spectrum

Telluric Star

- Preliminaries:
  - Copy the calibration files you will need into telluric directory:
    - Shift file
    - Flat
    - Bad pixel mask (BPM)
    - Ronchi mask + database dir+files
    - Arc file + database dir+files
  - Make two files listing filenames with (‘object’) and without (‘sky’) star in field

Telluric Star

- Step 1.1: Run nfprepare, making use of the shift file and BPM
- Step 1.2: Combine the blank sky frames:
  - Skies are close in time
  - Use gemcombine and your list of sky frames to create a median sky
- Step 1.3: Subtract the combined sky from each object frame with gemarith

Telluric Star

- Step 2.1: Run nsreduce, this time including the flat:

```
nsreduce("sn@telluriclist",outpref="r",
flatim=cal_data//"rn"//calflat//"_flat”, fl_nscut+, fl_nsappw-,
fl_vardq+, fl_sky-, fl_dark-, fl_flat+, logfile=log_file)
```

- Step 2.2: Replace bad pixels with values interpolated from fitting neighbours

```
nffixbad("rsn@telluriclist",outpref="b",logfile=log_file)
```

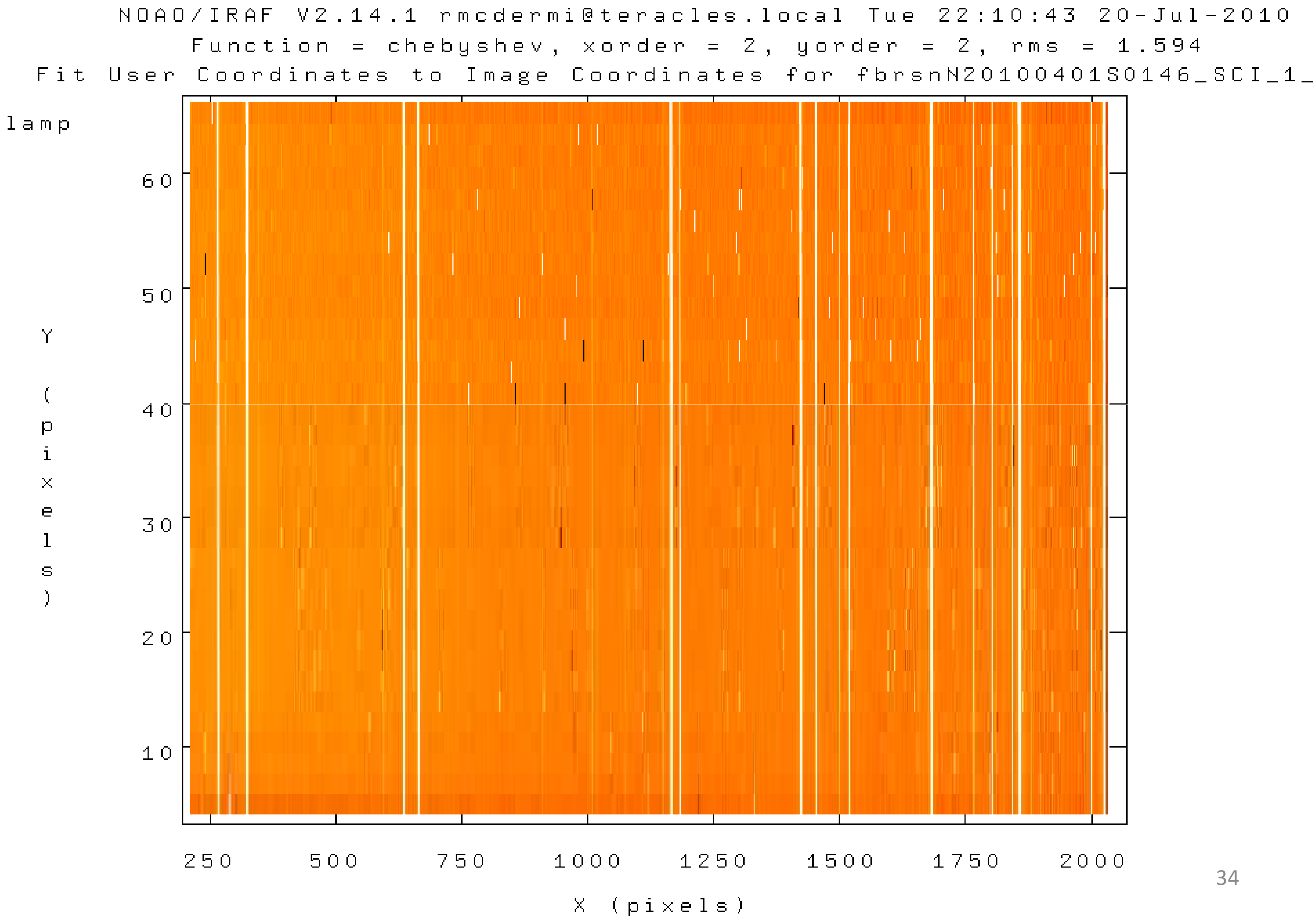
  – Uses the Data Quality (QD) plane

Telluric Star

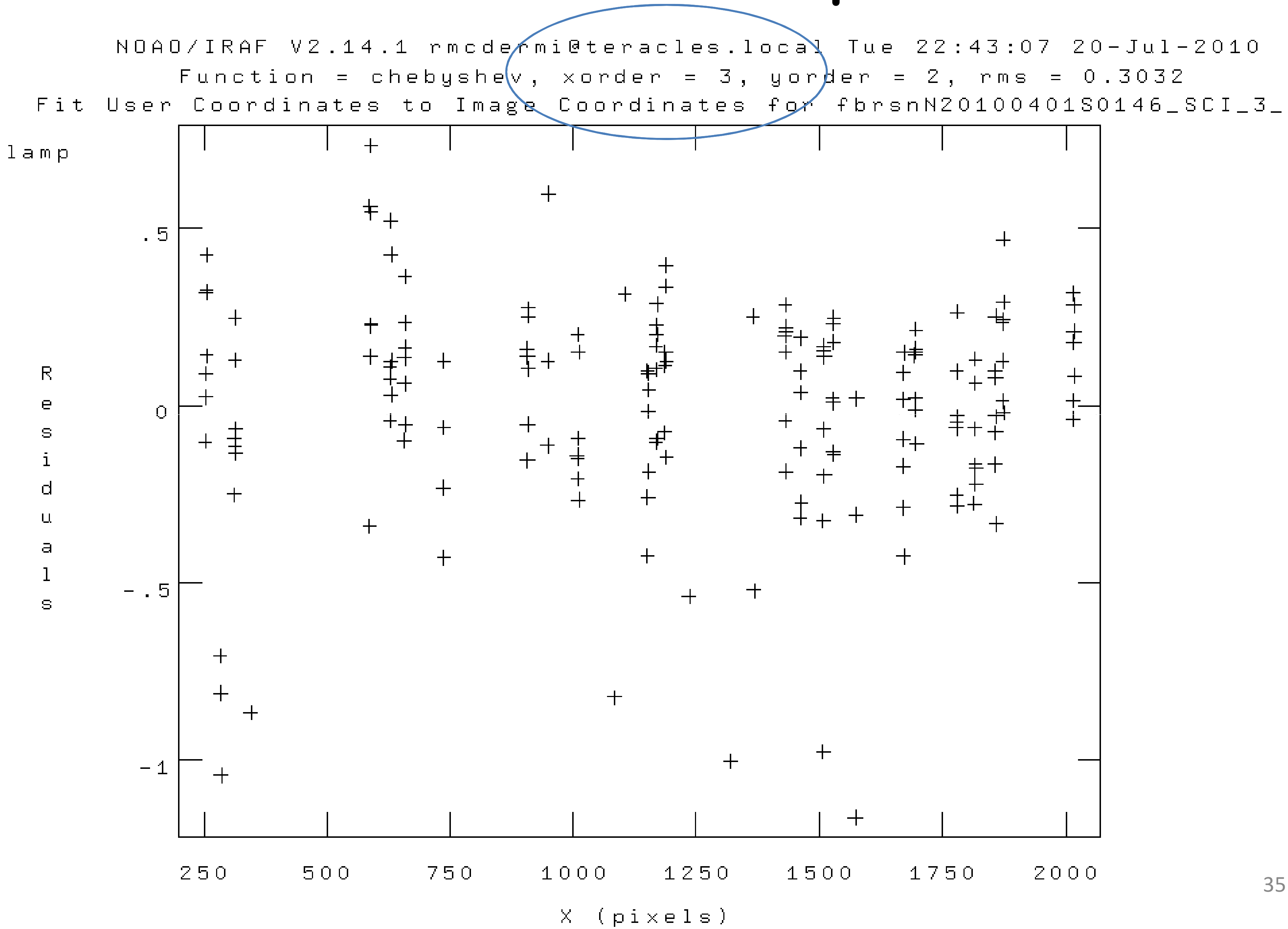
- Step 3.1: Derive the 2D spectral and spatial transformation for each slice using nsfitcoords
  - This combines the ‘1D’ dispersion and distortion solutions derived separately from nswavelength and nsdist into a 2D surface that is linear in wavelength and angular scales
  - The parameters of the fitted surface are associated to the object frame via files in the database directory

```
nsfitcoords("brsn@telluriclist", outpref="f", fl_int+,
lamptr="wrgn"//arc, sdisttr="rgn"//ronchiflat, lxorder=3,
lyorder=3, sxorder=3, syorder=3, logfile=log_file)
```

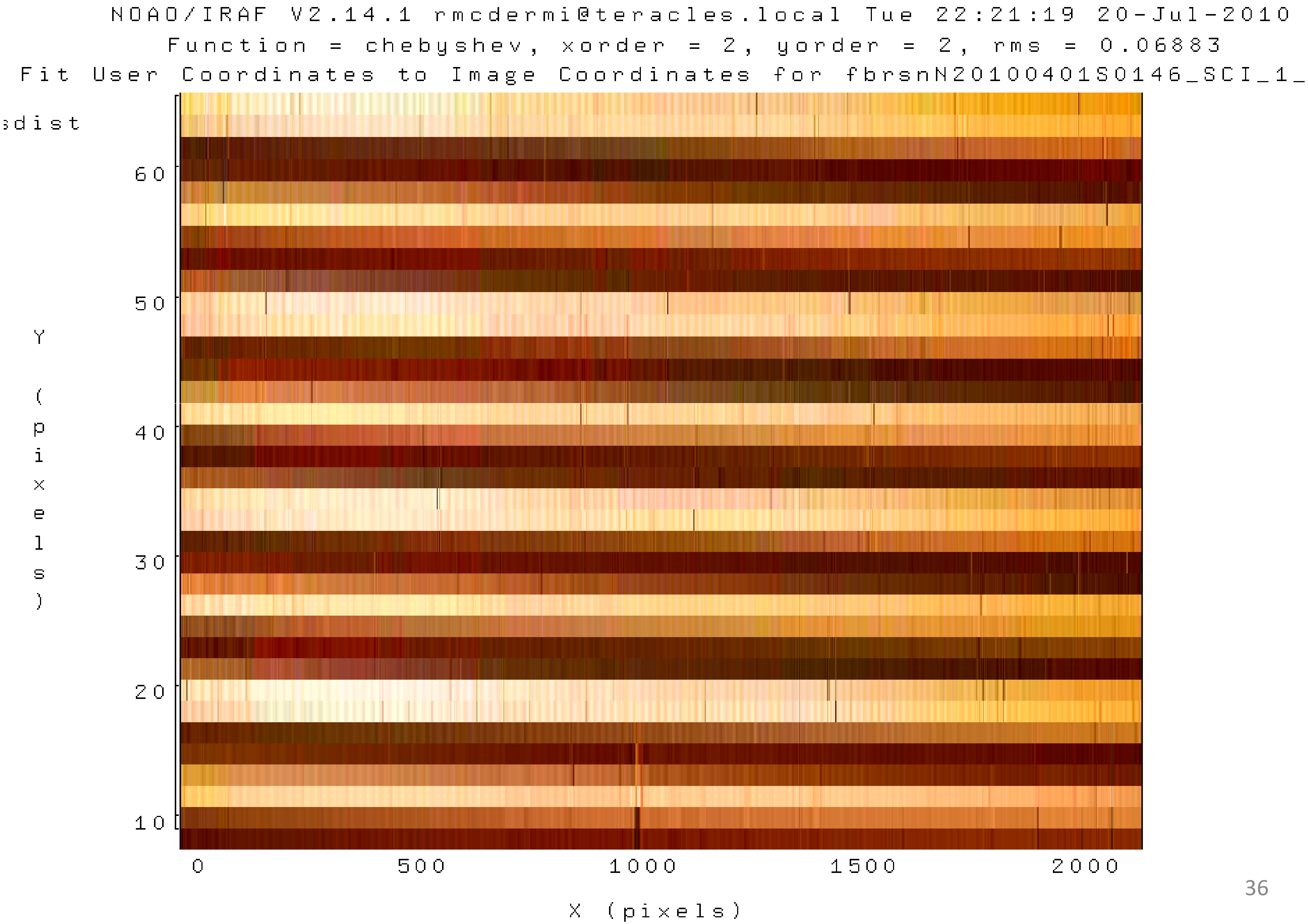
Nsfitcoords - spectral



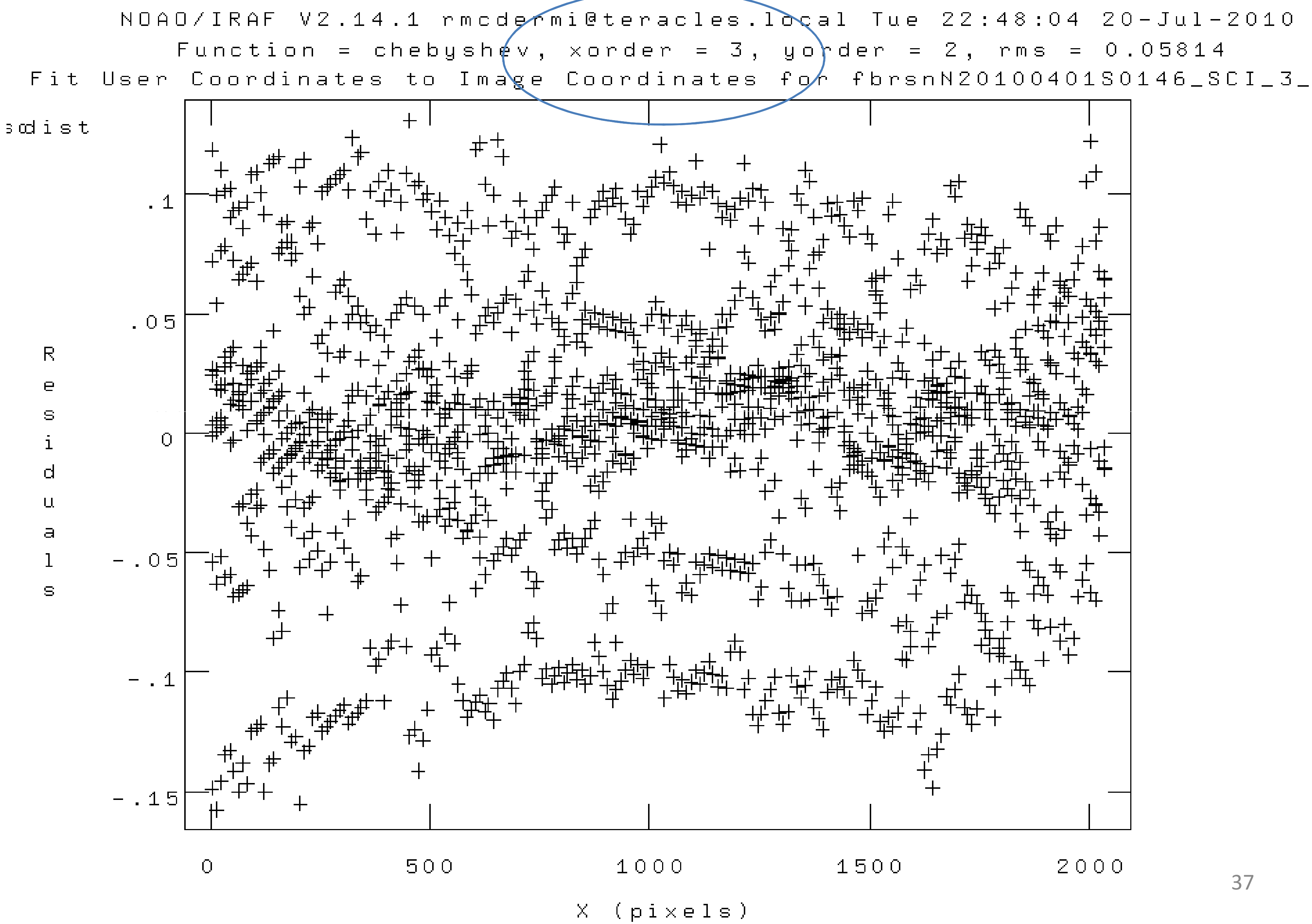
Nsfitcoords - spectral



Nsfitcoords - spatial



Nsfitcoords - spatial

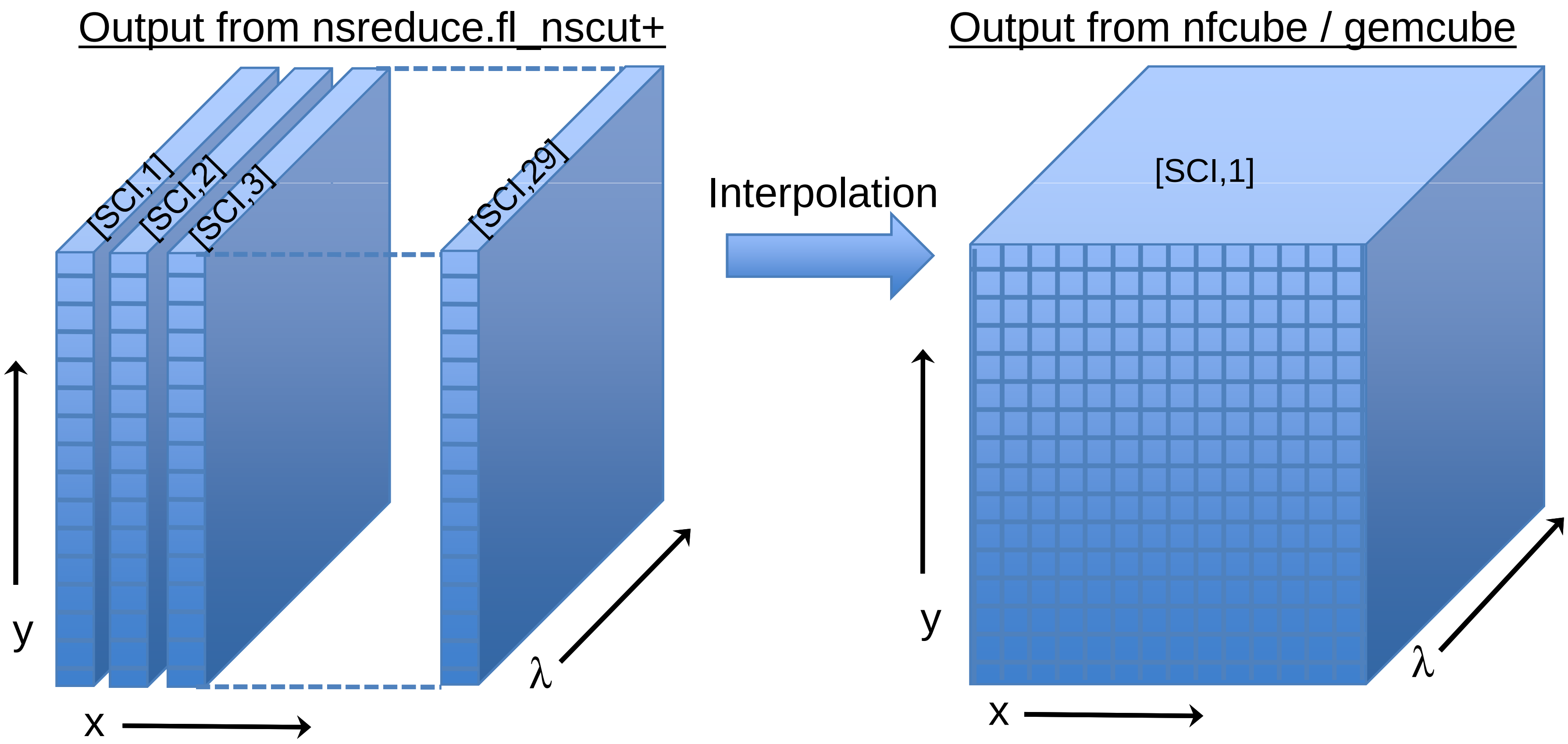


Telluric Star

- Step 3.2: Transform the slice images to the linear physical coordinates using nstransform
  – Uses transforms defined by nsfitcoords
  – Generates slices that are sampled in constant steps of wavelength and arcsec
- This is essentially a data-cube (even though its not a cube…)
  – Can run analysis directly from this point

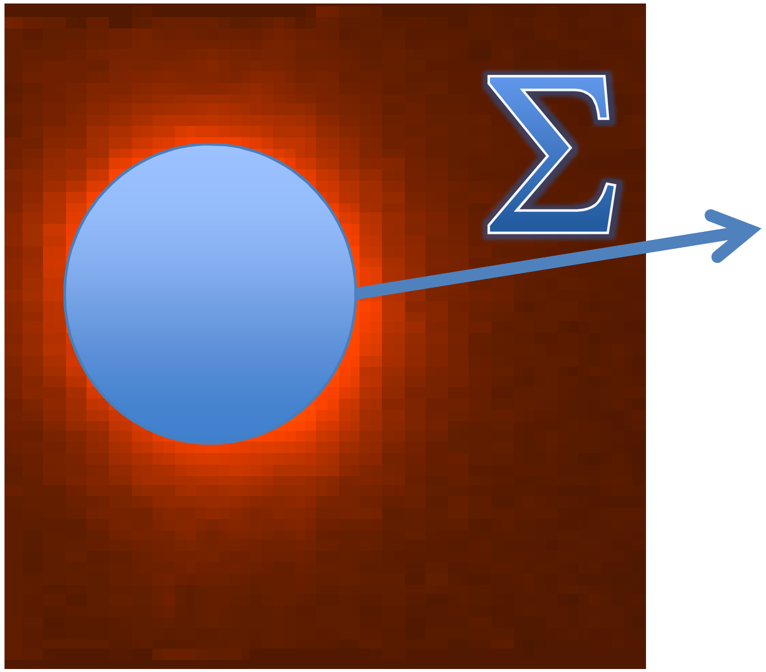
When is a cube not a cube?

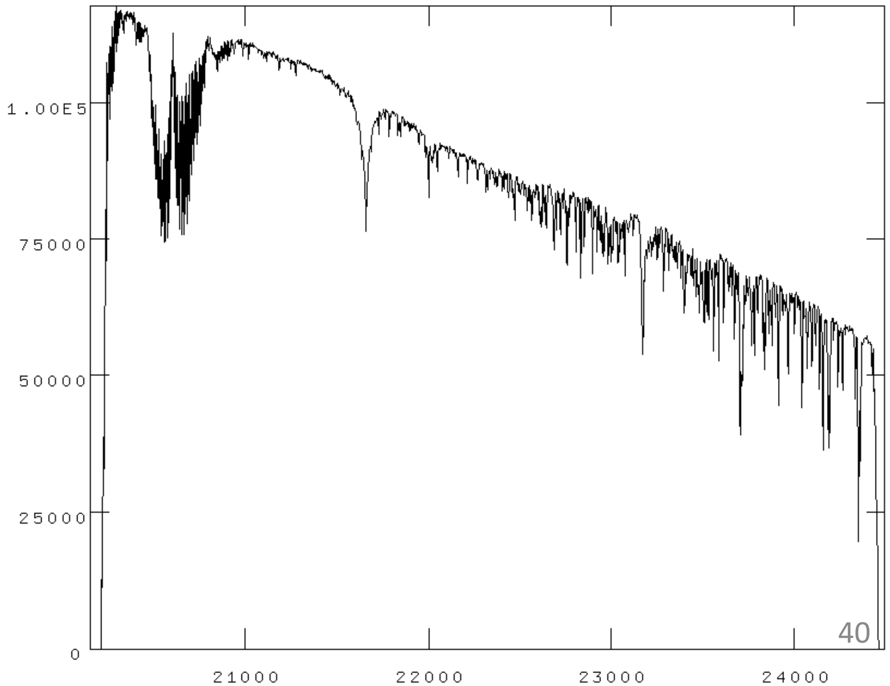
- 'Cube' = (x,y,λ)
- Does not have to be a rectilinear 3D array



Telluric Star

- Step 4.1: Extract 1D aperture spectra from the data cube
  - Use nfextract to define an aperture (radius and centre) and sum spectra within it
  - Outputs a 1D spectrum
- Step 4.2: Co-add the 1D spectra using gemcombine





Science Data

- Same preliminaries as telluric:
  - Copy database and arc+Ronchi files
  - Copy shift file, flat and BPM
  - Identify sky and object frames
- In addition, we make use of the 1D telluric
- Generally need to combine separate (and dithered) data-cubes

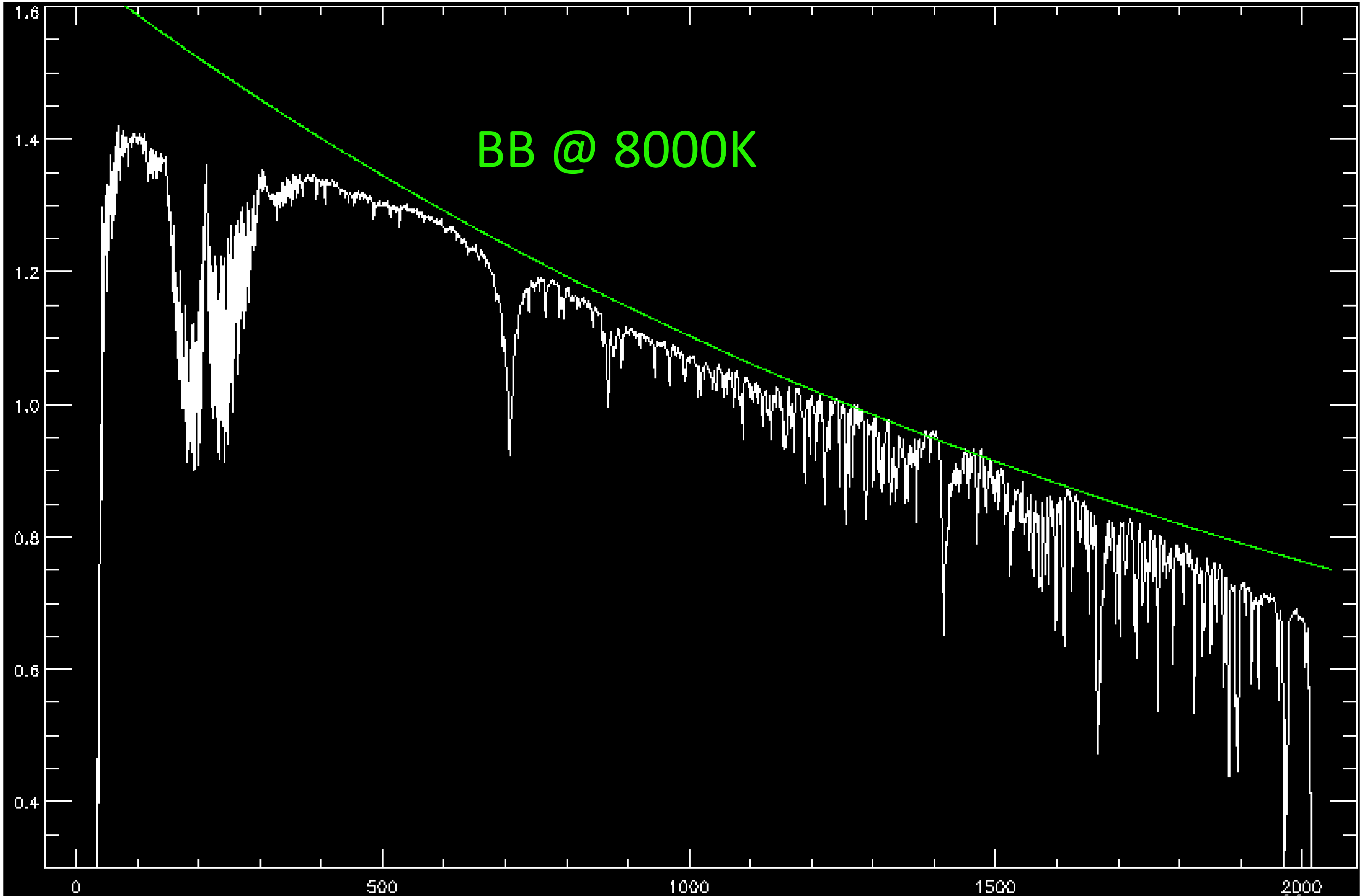
Science Data

- Initial steps:
  - Nfprepare as per telluric
  - Subtract sky using gemarith
    - Usually have one unique sky per object: ABAB
    - Can have ABA – two science share a sky
  - Nsreduce (inc. flat field)
  - Nffixbad, nsfitcoords, nstransform
- Now have data-cube with linear physical coordinates

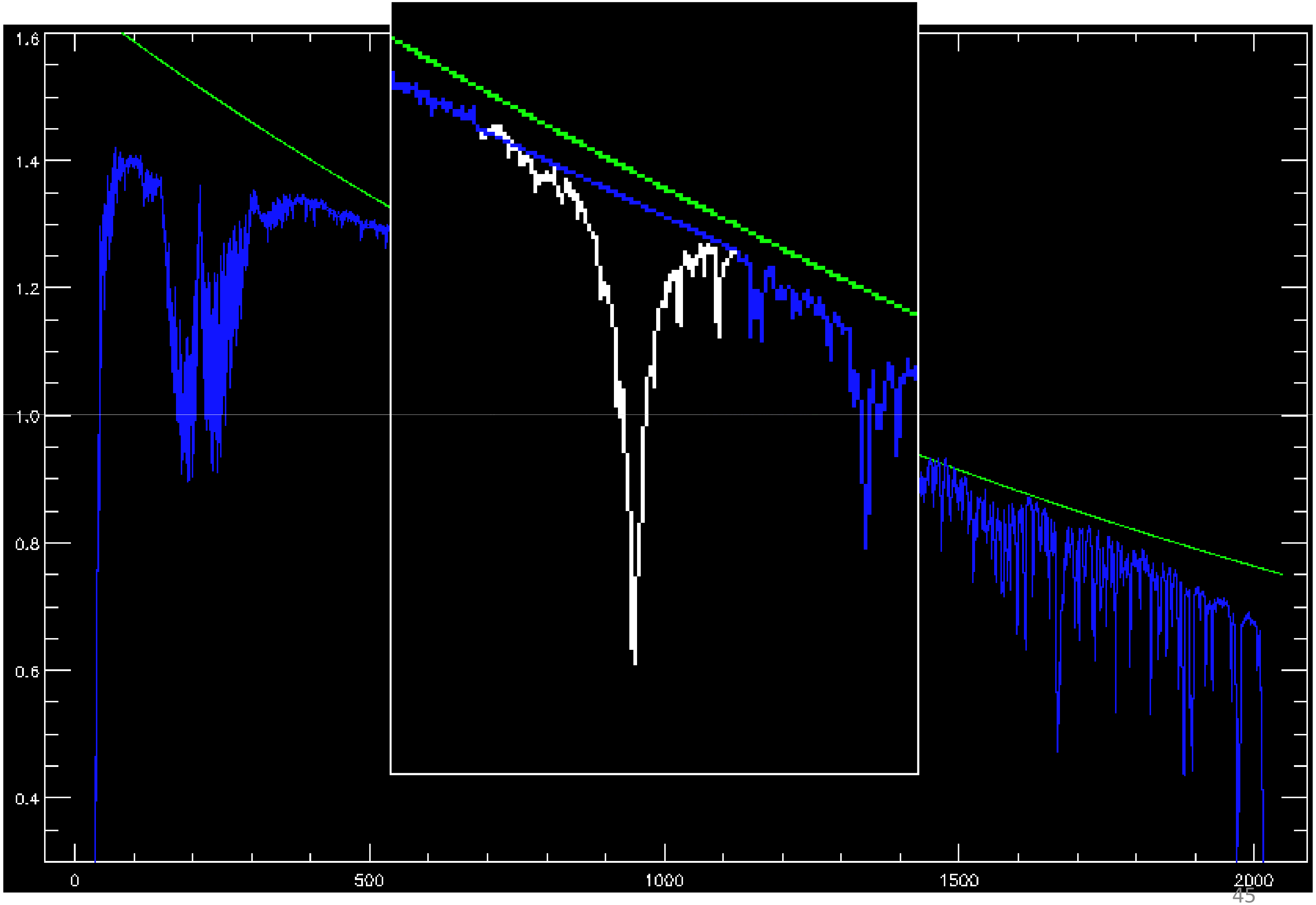
Science Data: Telluric correction

- Telluric spectrum is not only atmosphere, but also stellar spectrum:
  - Need to account for stellar absorption features
  - AND account for black-body continuum
- Needs some ‘by-hand’ steps to prepare the telluric star spectrum
  - Remove strong stellar features with splot
  - Remove BB shape with a BB spectrum

Science Data: Telluric correction

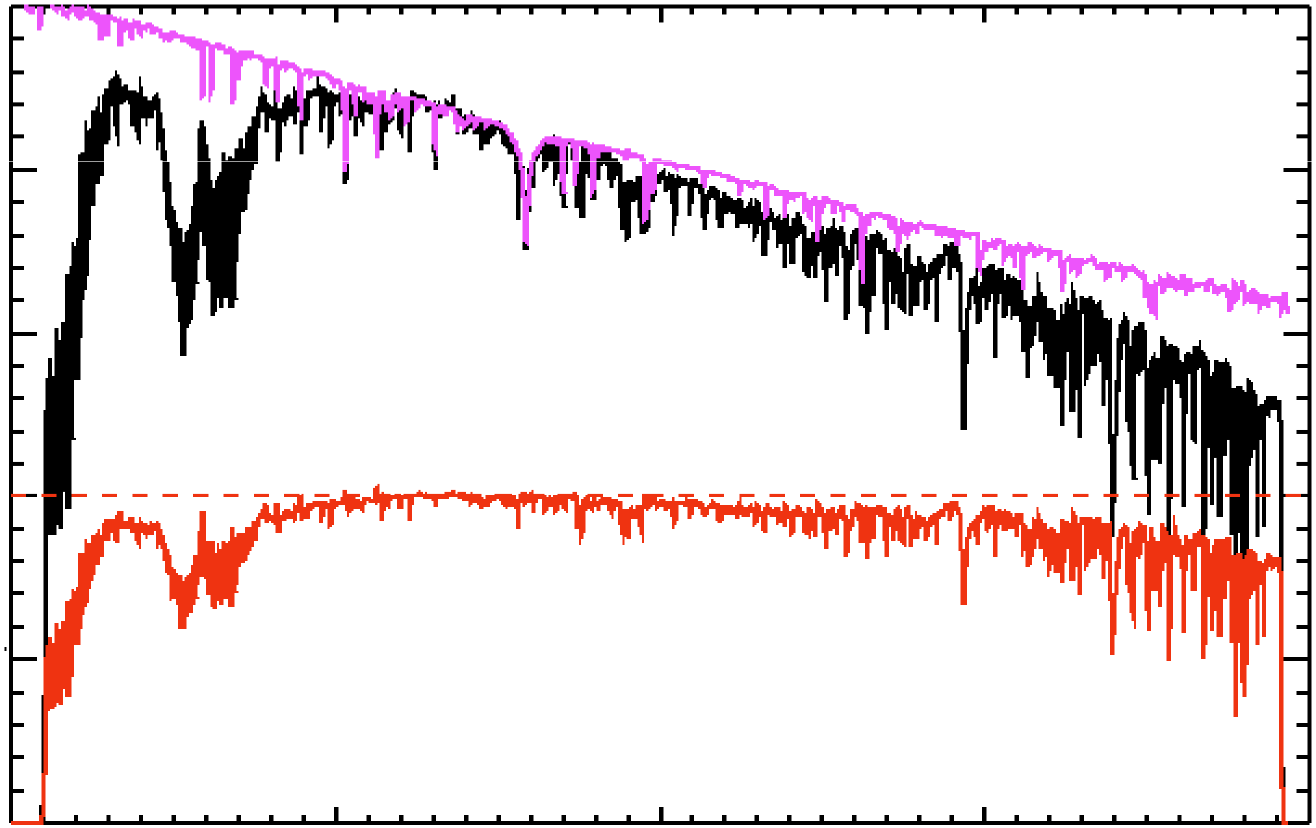


Science Data: Telluric correction



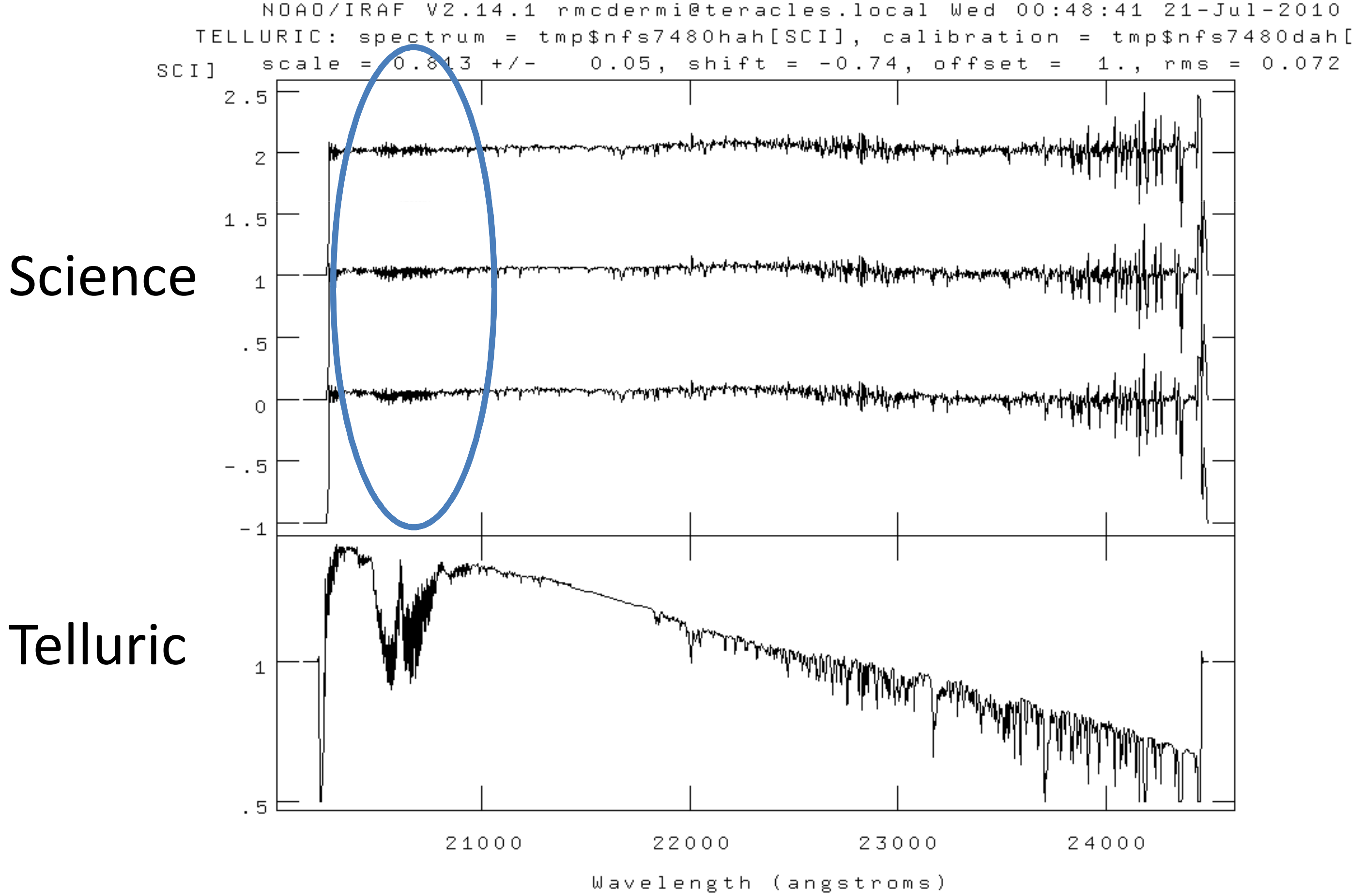
Telluric Absorption

- Alternative approach is to fit a stellar template (Vacca et al. 2003)
- Need good template
- Can use solar-type stars, but needs careful treatment…



Science Data: Telluric correction

- Finally, run nftelluric
  - Computes the normalized correction spectrum
  - Allows for shifts and amplitude scaling
  - Divides the correction spectrum through the data



Science Data: Merging

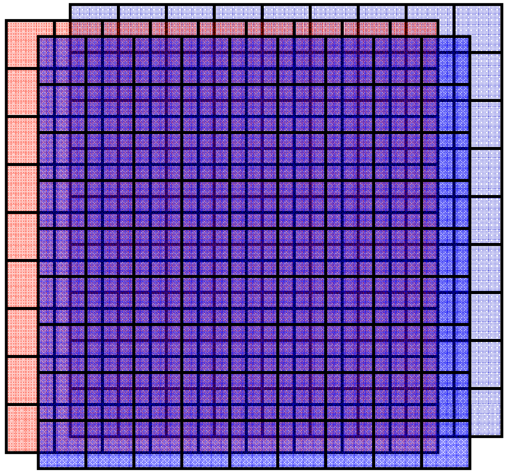
- Now have series of data-cubes:
  - No dark current or sky (sky-subtracted)
  - Spatially and spectrally linearized
  - Bad pixels interpolated over
  - No instrumental transmission (flat-fielded)
  - No atmospheric transmission (telluric-corrected)
- Need to **combine** the data-cubes

Science Data: Merging

Two approaches:

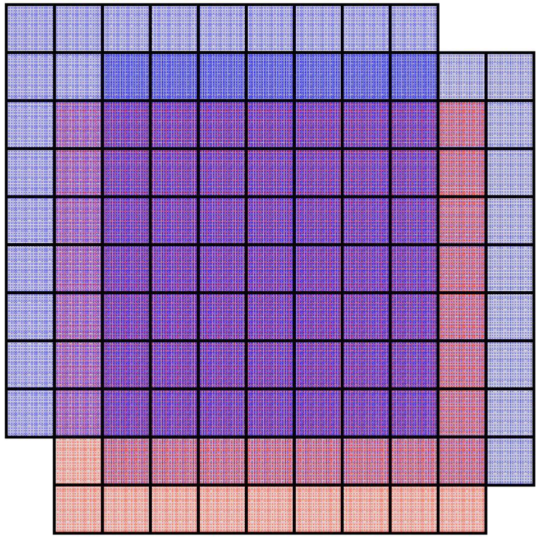
1. Dithering by non-integer number of spaxels:
   - Allows over-sampling, via 'drizzling'
   - Resampling introduces correlated noise
   - Good for fairly bright sources

Nifcube / gemcube method



2. Dither by integer number of spaxels
   - Allows direct 'shift and add' approach
   - No resampling:- better error characterisation
   - Assumes accurate (sub-pixel) offsetting
   - No over-sampling gain
   - Suitable for 'deep-field' applications

Imcombine method



Science Data: Merging with nf/gemcube

1. Use nfcube to create the 3D arrays
   – 2D slices to 3D cube - interpolation
   – Default pixel scale is 0.05”x0.05” (arrays need square pixels..)
2. Find a reference pixel coordinate
   – These cubes are easily displayed using ds9
   – Load as an array, scroll through the slices
   – Reference point should be an easily recognizable feature, common to all the cubes
   – Can blindly use offsets, but not recommended
3. Adapt the headers to reflect the common spatial axes origin
4. Run gemcube (interpolates again)

What headers to change?

- Gemcube uses the file WCS to merge overlapping cubes
- Need to ‘trick’ the WCS to include your offsets
- Following list seemed to work:
  - Position of reference source measured on cube plane – different in each cube
    - CRPIX1 - Reference source (x position in pixels), e.g ‘29’
    - CRPIX2 - Reference source (y position in pixels), e.g. ‘27.5’
  - Approx decimal RA & Dec at reference point – same in all cubes
    - CRVAL1 - approx. decimal RA of target (same for all cubes)
    - CRVAL2 - approx. decimal declination of target
  - Pixel scale parameters - these are set here to be 0.05 arcsec, but in degrees
    - CD1_1 = 0.0
    - CD1_2 = 1.9E-05
    - CD2_1 = -4.7e-5
    - CD2_2 = 0.0
  - Coordinate system info
    - WAT1_001 = 'wtype=linear axtype=xi’
    - WAT2_001 = 'wtype=linear axtype=eta’
    - CTYPE1 = 'RA---TAN’
    - CTYPE2 = ‘DEC--TAN'

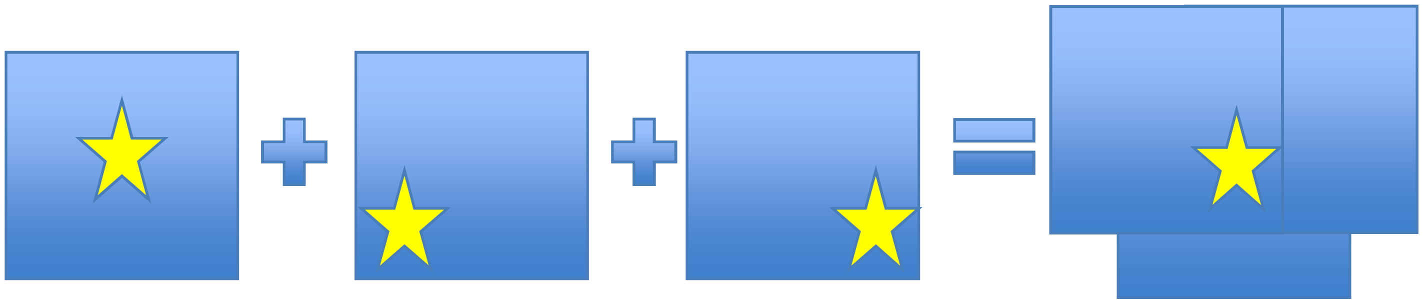
Hack alert! This is not the best way to proceed! But lets you progress within iraf. Other ideas are welcome!

Science Data: Merging with nf/gemcube

- This approach involves (at least) one superfluous interpolation: nifcube + gemcube both interpolate
- Might be possible to use gemcube directly from pre-transformed data, but wrapper not written (TBD: works on single slices, so can be adapted)
- Nifcube step is convenient for determining reference coordinate, and allows gemcube to combine the frames
- Not ideal, but gives a way to combine your data

Science Data: Merging with imcombine

1. Convert files to 3D cubes with nfcube
2. Provide the list of input files, and an ‘offsets’ file of integer shifts to imcombine



offsets.dat

x	y	λ
0	0	0
10	10	0
-10	10	0

```
imcombine(”cat*.fits", output=“Merged”,offset=offsets.dat,
logfile=log_file)
```

Science Data: Merging

Other details for merging not covered:

- For smoother images / less border effects, individual cubes should be renormalized to have common flux in overlap regions
- For optimal S/N, spectra should be weighted by the variance when combining
- Even without dithers, long observations will have sub-pixel shifts due to flexure and atmospheric refraction
- To maintain high image quality, keep a reference source common to all fields in the mosaic if possible, and use sub-pixel dithers

The End

(….or rather, just the beginning….)