LONG-SLIT REDUCTION WITH PYRAF

Rubén García-Benito

IAA-CSIC

Part II

Optical Spectroscopy School
University of Rwanda
August 24th - September 2nd, 2018

Reduction steps

1) *Pre-processing (correct frames):*
   * *Master Bias*
   * *Master Flat*

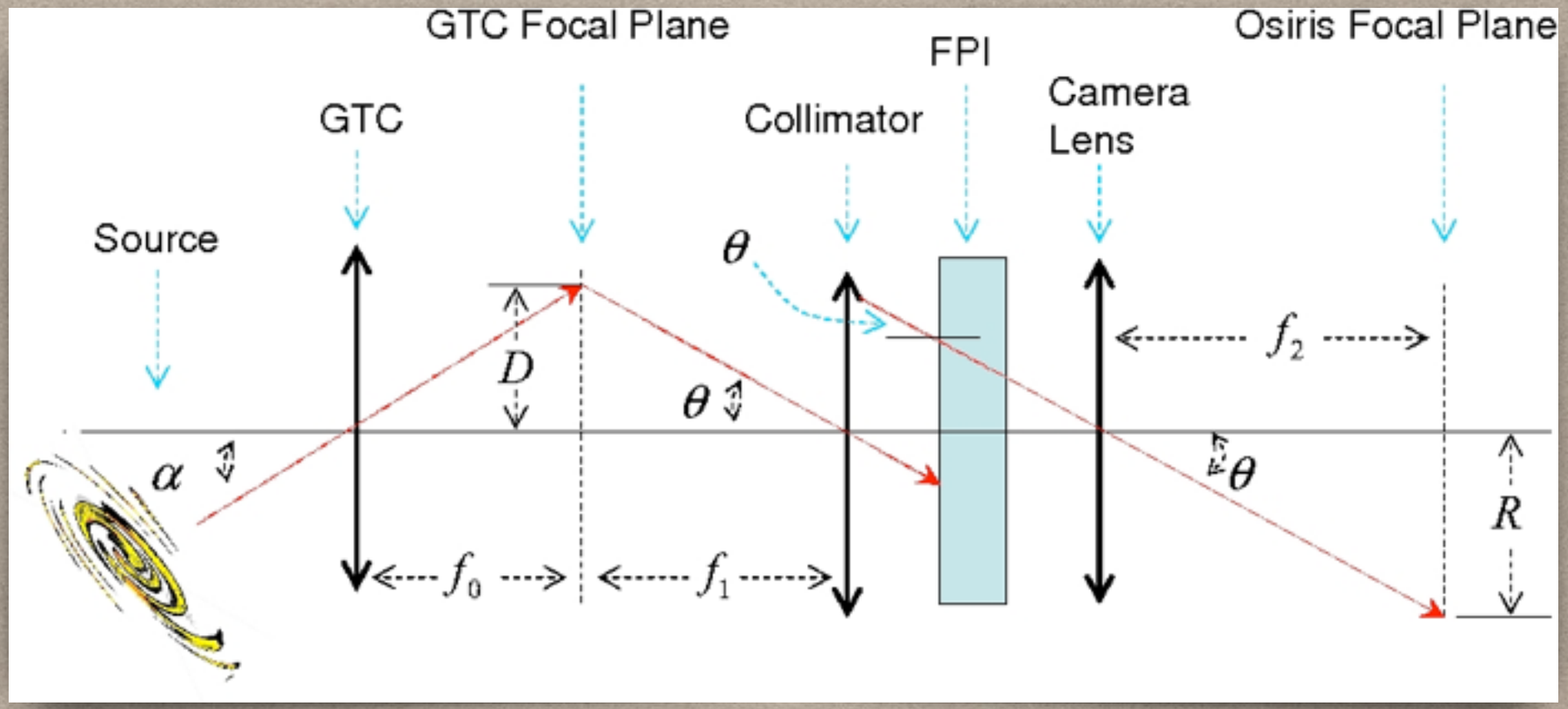
Frame - <bias>

<flat> - <bias>

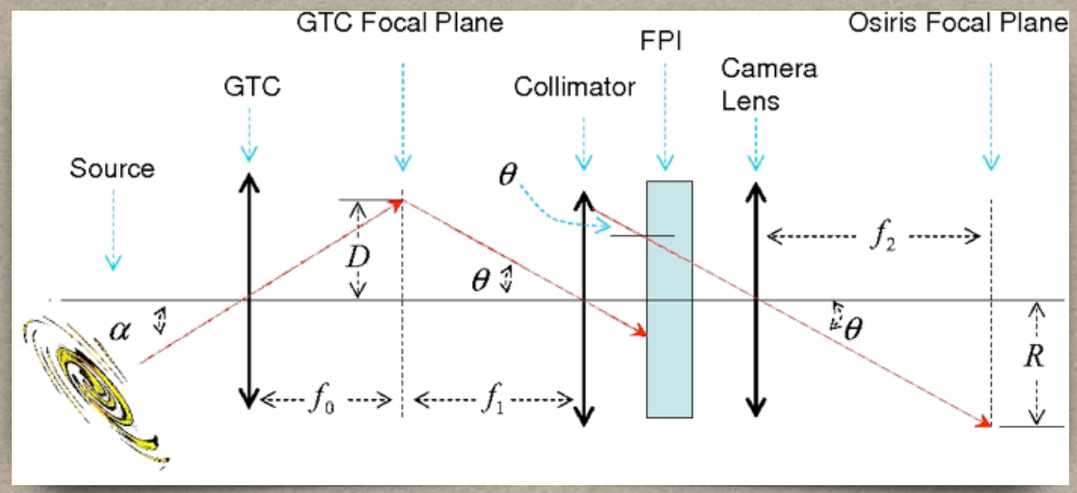
2) **Wavelength calibration:**
   * **identify**
   * **reidentify**
   * **fitcoords**

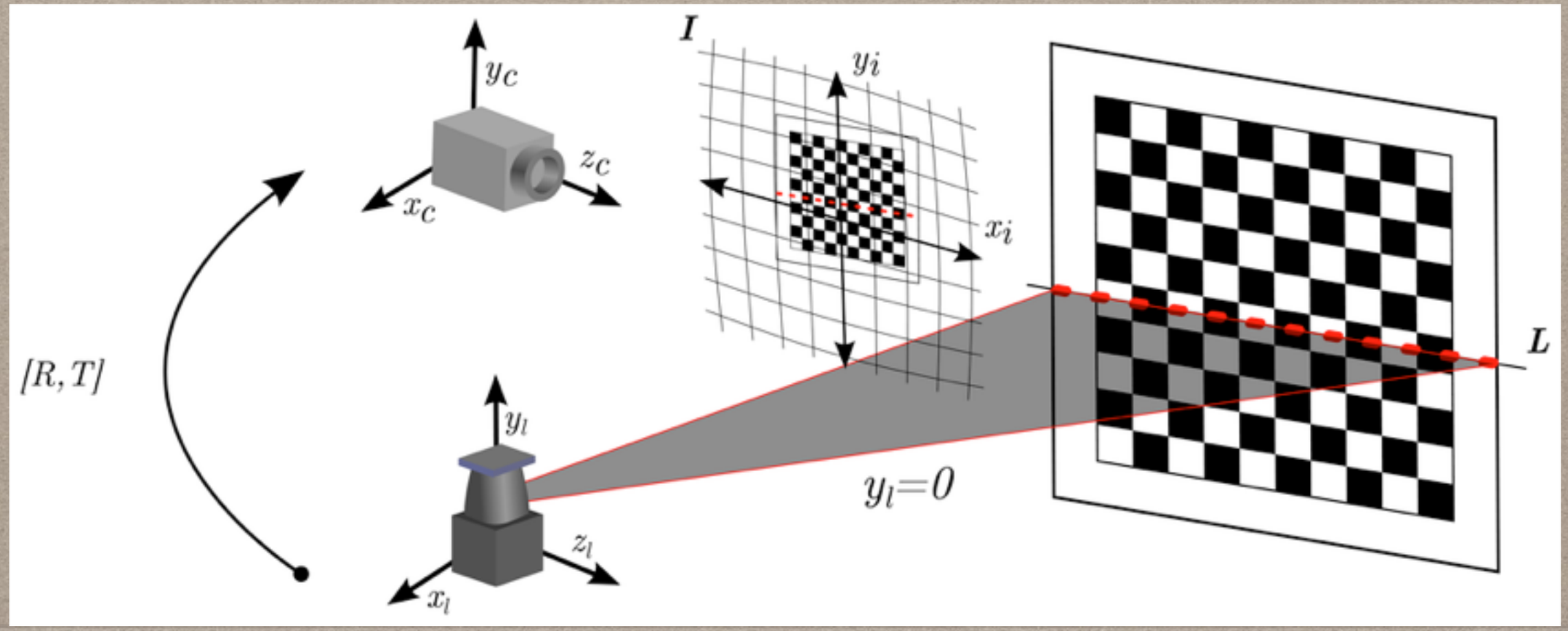
3) *Flux calibration & sky subtraction:*
   * standard
   * sensfunc
   * calibrate

Wavelength calibration

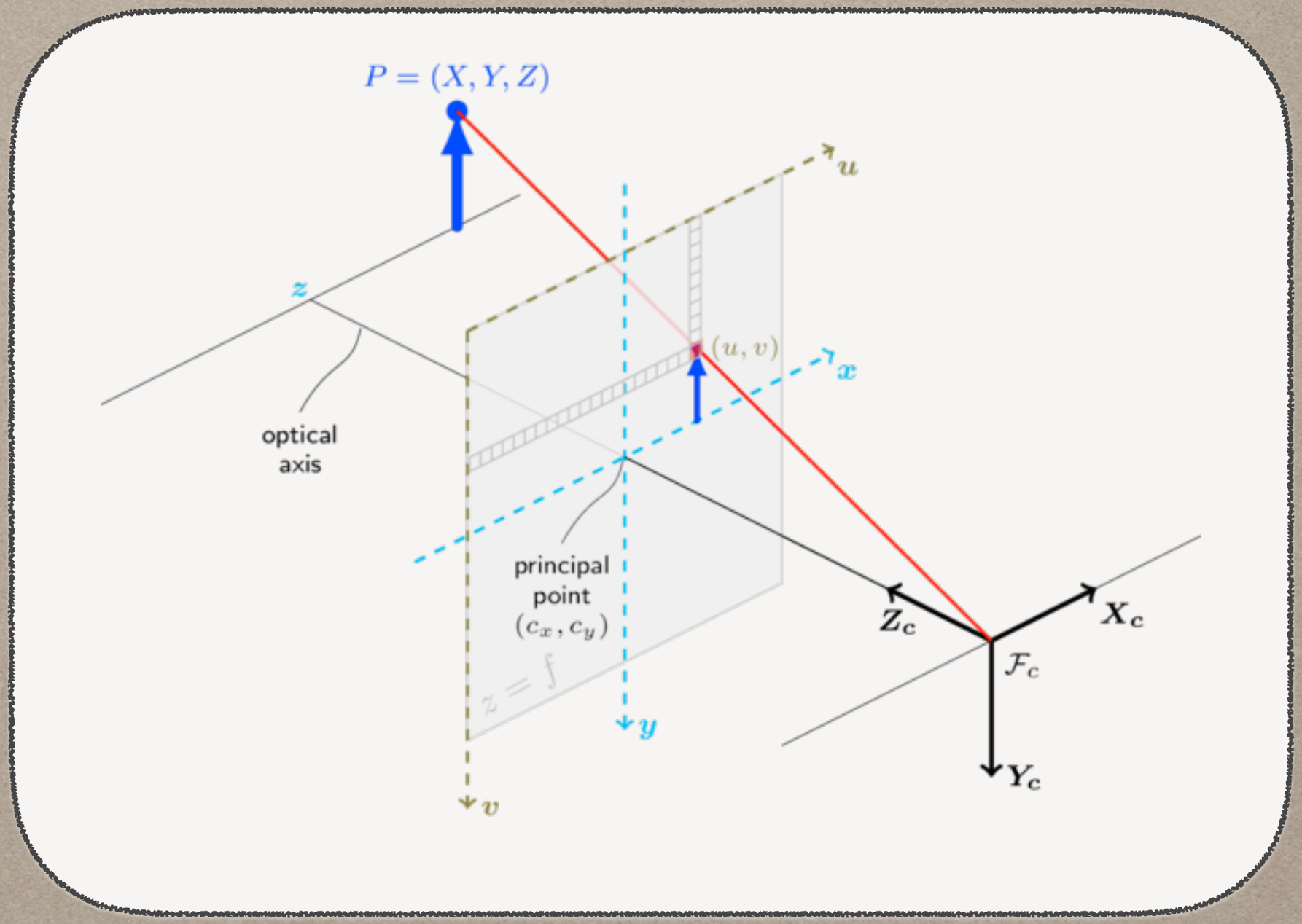


Wavelength calibration





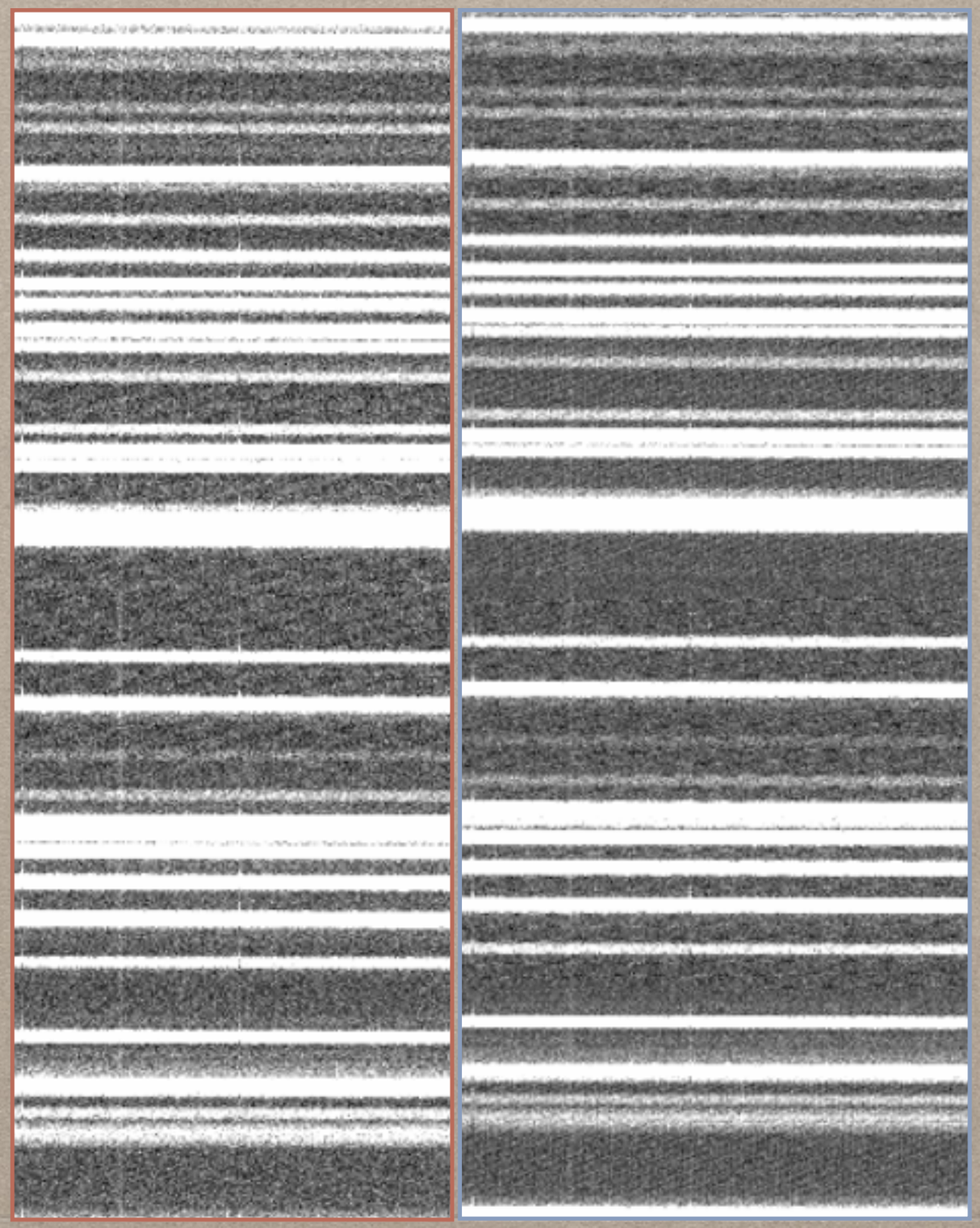
Wavelength calibration



Wavelength calibration

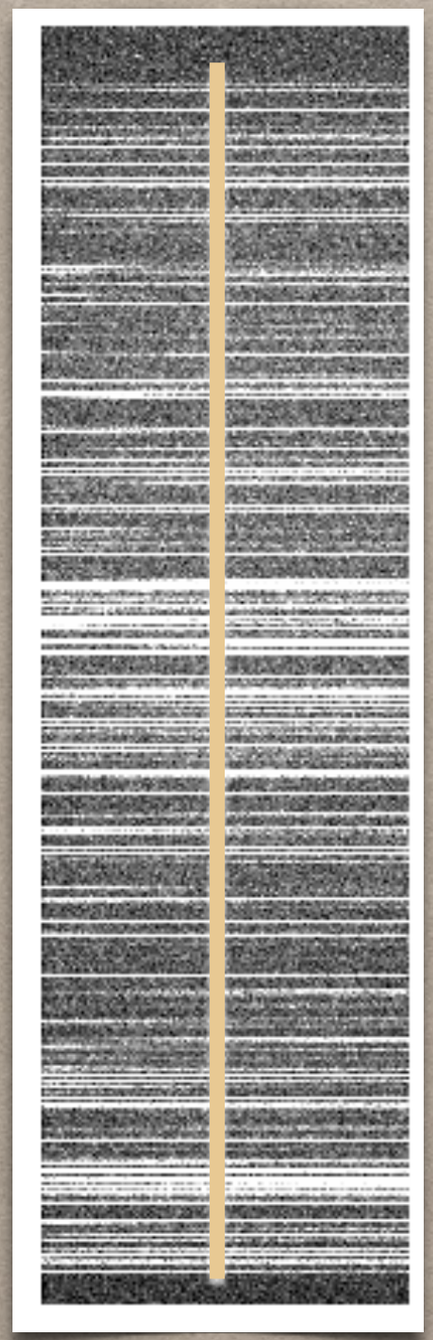
λ

Wavelength calibration



λ

Wavelength calibration



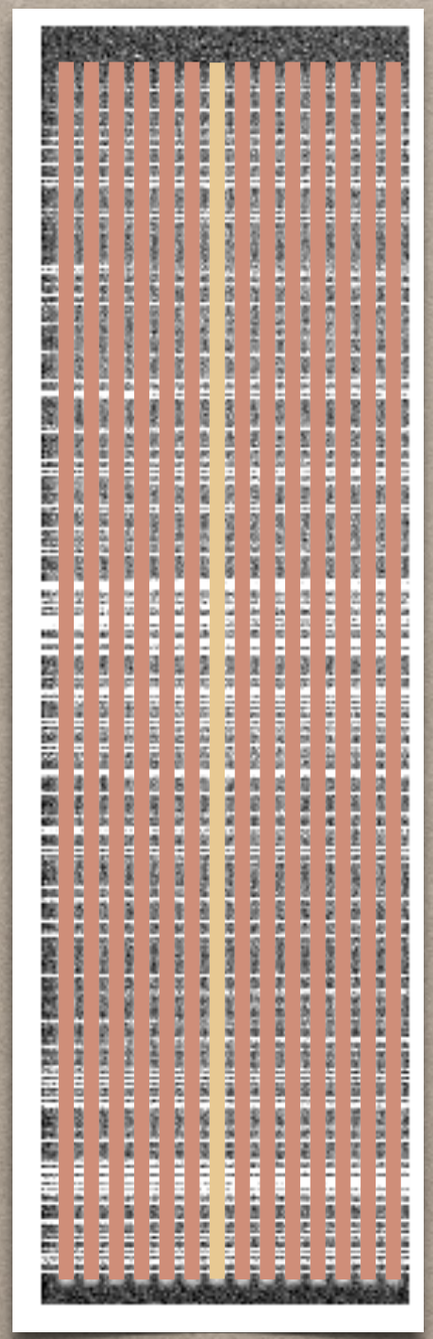
1) identify

1-D identification

2) reidentify

3) fitcoords

Wavelength calibration



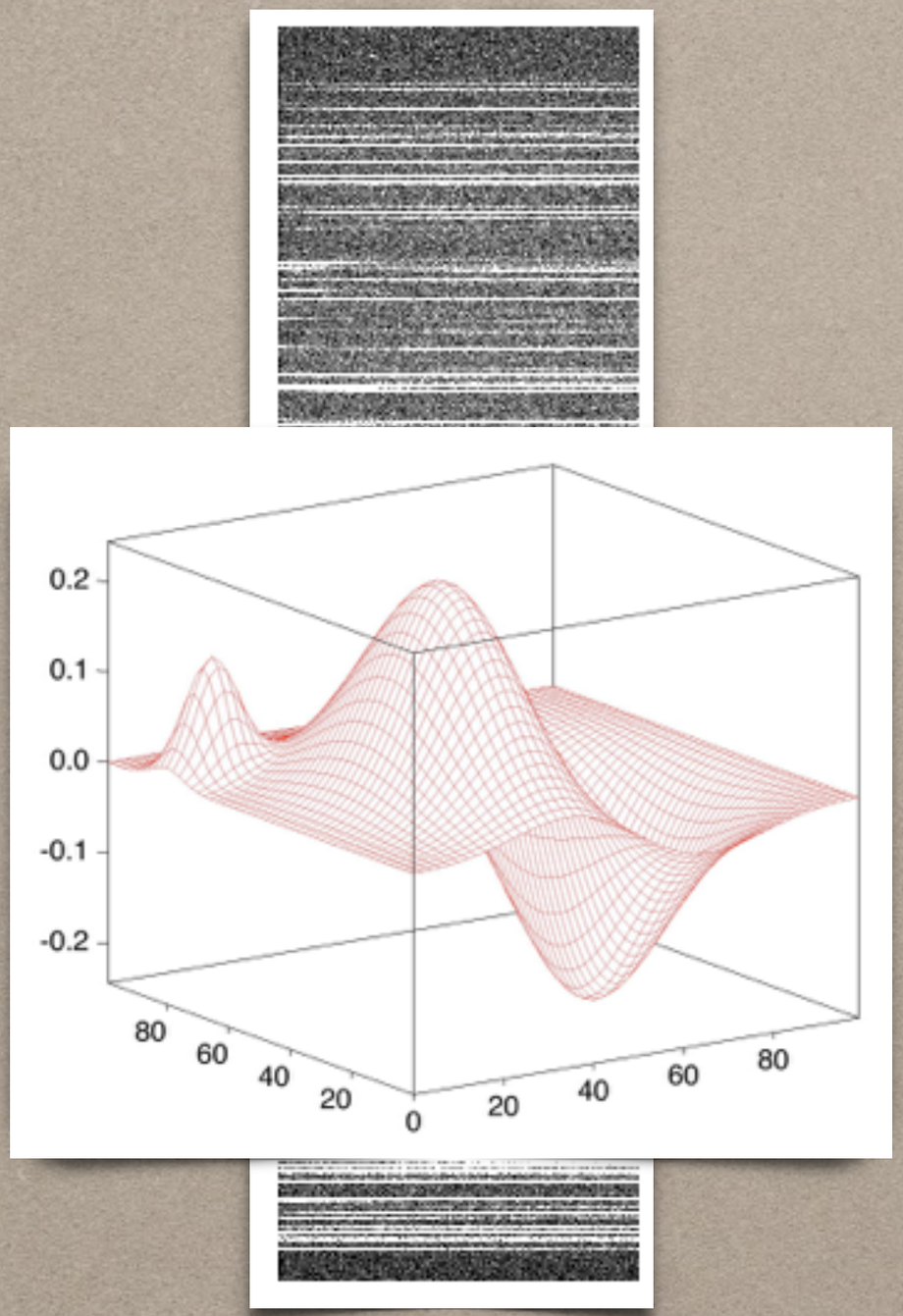
1) identify

2) reidentify

2-D identification

3) fitcoords

Wavelength calibration



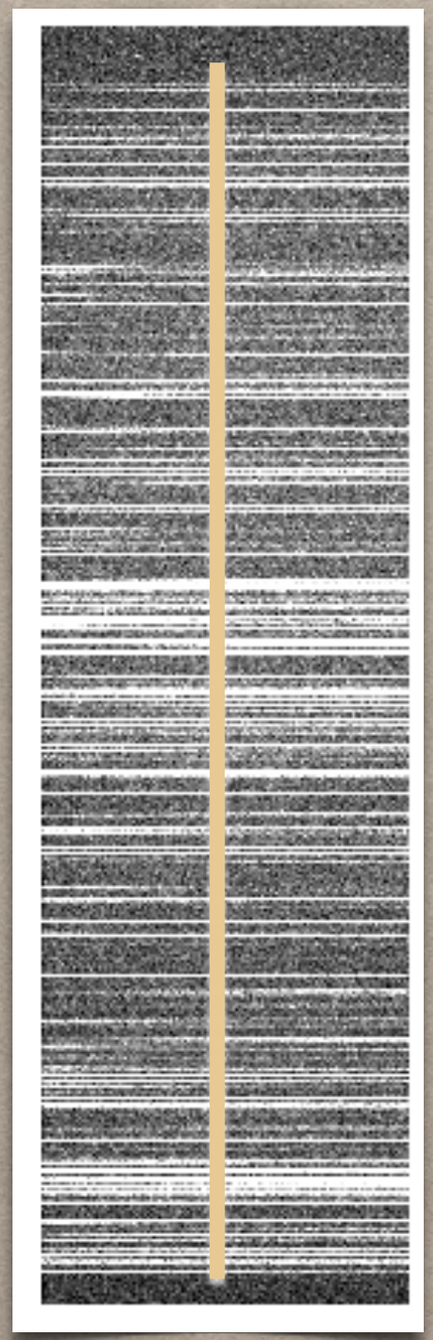
1) identify

2) reidentify

3) fitcoords

Fit a surface $\lambda = f(x, y)$

Wavelength calibration



1) identify

1-D identification

2) reidentify

3) fitcoords

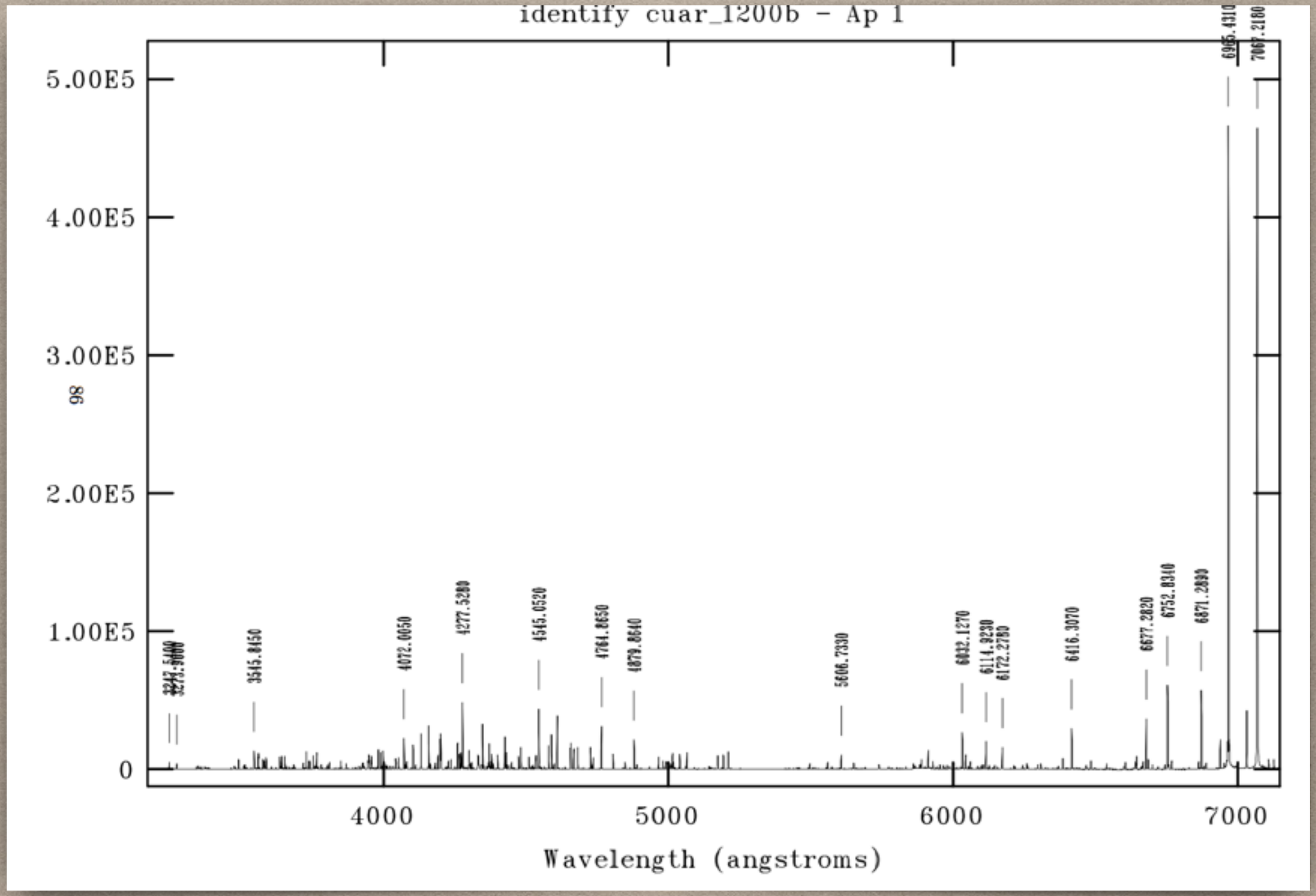
Wavelength calibration

1) identify

```
-->
--> lpar identify
       images =                  Images containing features to be identified
        crval =                  Approximate coordinate (at reference pixel)
        cdelt =                  Approximate dispersion
     (section = "middle line")   Section to apply to two dimensional images
    (database = "database")      Database in which to record feature data
   (coordlist = "linelists$idhenear.dat") User coordinate list
       (units = "")              Coordinate units
        (nsum = "10")            Number of lines/columns/bands to sum in 2D images
       (match = -3.0)            Coordinate list matching limit
 (maxfeatures = 50)              Maximum number of features for automatic identification
      (zwidth = 100.0)           Zoom graph width in user units
       (ftype = "emission")      Feature type
      (fwidth = 4.0)             Feature width in pixels
     (cradius = 5.0)             Centering radius in pixels
   (threshold = 0.0)             Feature threshold for centering
      (minsep = 2.0)             Minimum pixel separation
    (function = "spline3")       Coordinate function
       (order = 1)               Order of coordinate function
      (sample = "*")             Coordinate sample regions
    (niterate = 0)               Rejection iterations
  (low_reject = 3.0)             Lower rejection sigma
 (high_reject = 3.0)             Upper rejection sigma
        (grow = 0.0)             Rejection growing radius
   (autowrite = no)              Automatically write to database
    (graphics = "stdgraph")      Graphics output device
      (cursor = "")              Graphics cursor input
     (aidpars = "")              Automatic identification algorithm parameters
        (mode = "al")
```

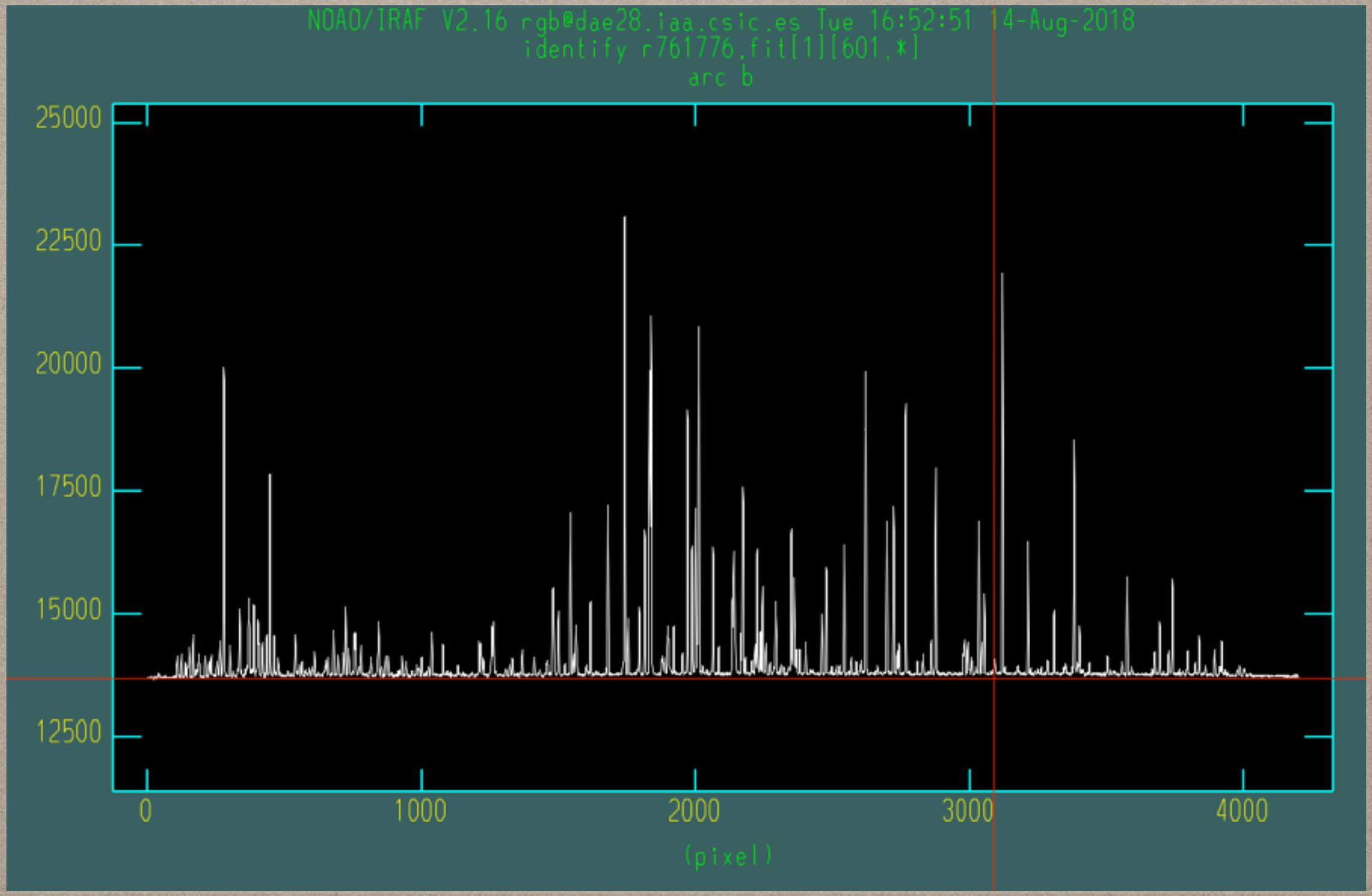
Wavelength calibration

1) identify



Wavelength calibration

1) identify



Wavelength calibration

1. vim

file	run	object	RA	Dec	equinox	UT	airm	instr	texp	PA	slit	grat	cenwA
r761744.fit	761744	bias	14:01:38.81	01:32:40.1	J2000.0	20:47	1.14	ISISB	0.0	180	1.03	R600B	4302
r761746.fit	761746	bias	14:01:38.74	01:32:39.3	J2000.0	20:48	1.14	ISISB	0.0	180	1.03	R600B	4302
r761748.fit	761748	bias	0:00:00.00	00:00:00.0	J2000.0	20:48	1.05	ISISB	0.0	180	1.03	R600B	4302
r761750.fit	761750	bias	13:47:15.55	17:27:24.7	J2000.0	20:49	1.05	ISISB	0.0	180	1.03	R600B	4302
r761752.fit	761752	bias	13:47:15.62	17:27:24.7	J2000.0	20:50	1.05	ISISB	0.0	180	1.03	R600B	4302
r761771.fit	761771	bd33	15:52:00.15	32:56:47.3	J2000.0	21:23	1.01	ISISB	240.0	76	1.02	R600B	4302
r761773.fit	761773	bd33	15:52:00.15	32:56:47.3	J2000.0	21:28	1.01	ISISB	240.0	76	1.02	R600B	4302

 # Dispersion [A/pix] = 0.45 Å / px (B)

 # Dispersion axis ~ 4000 pixels

 # cenwA = 4302 Å

 # ARC LAMPS in Blue arm: CuNe+CUAR

(copper-Argon +Copper-Neon)

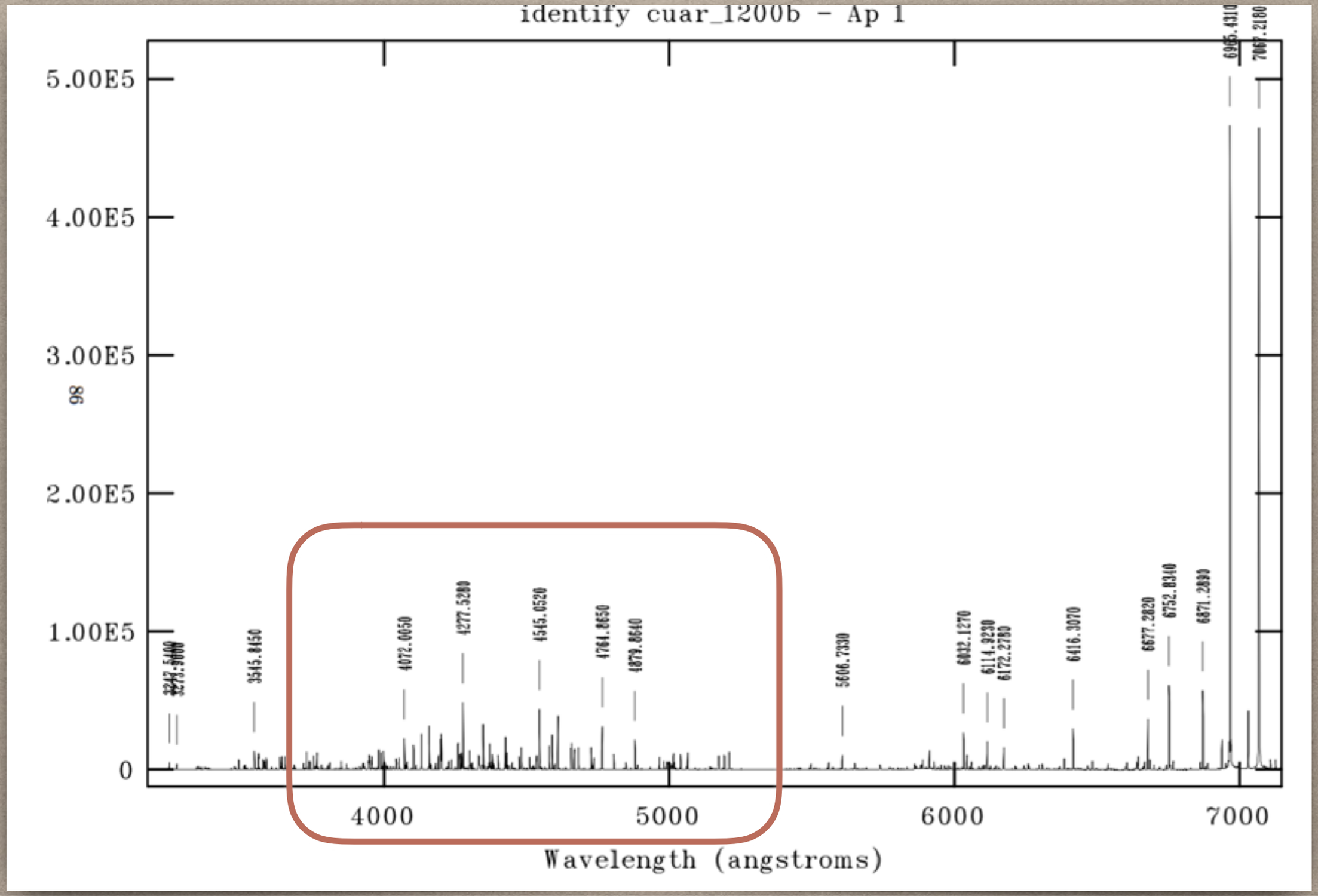
Wavelength range?

0.45 Å /px X 4000 px = 1800 Å

4300 - 900 < λ < 4300 + 900 => ***3400 Å < λ < 5200 Å***

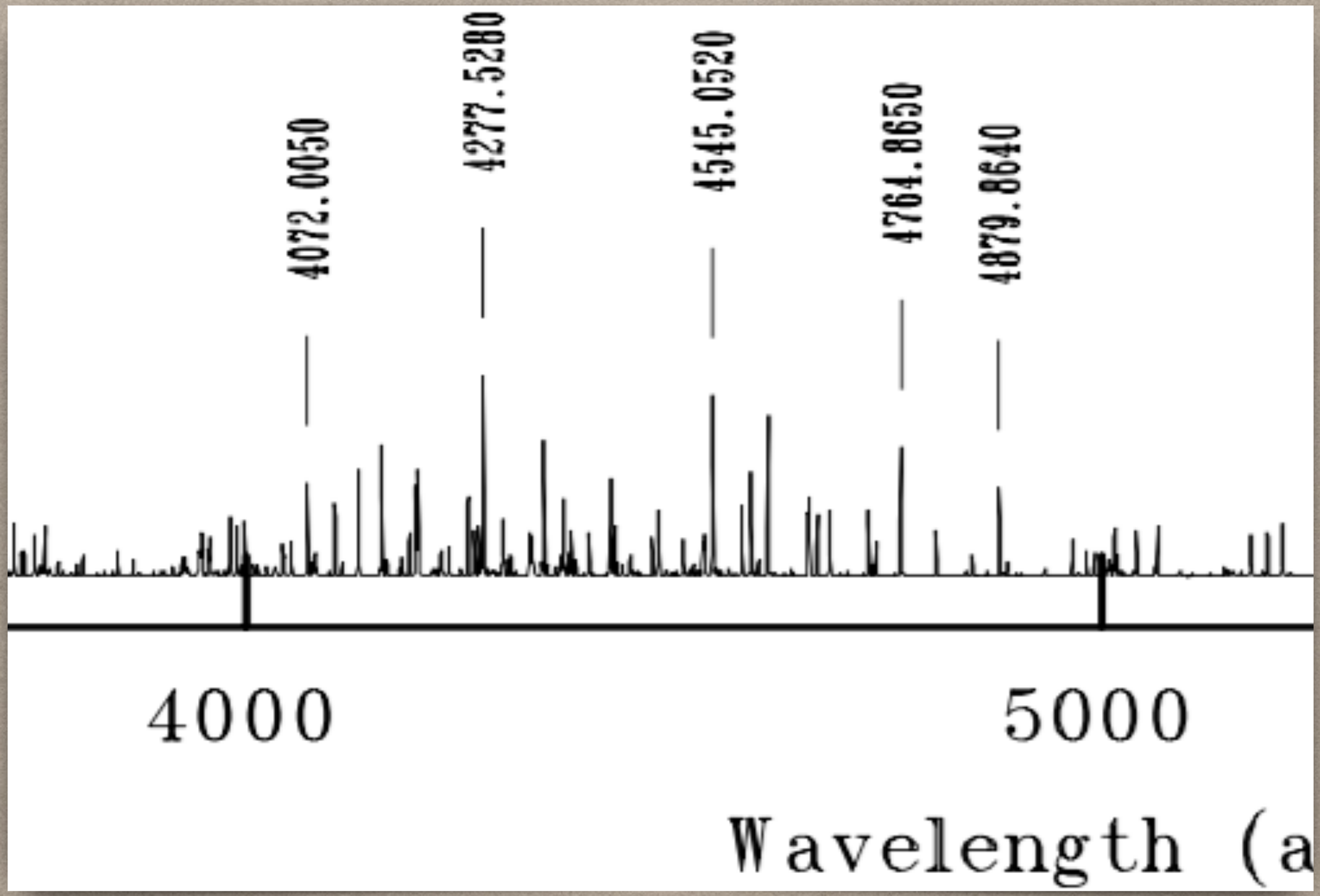
Wavelength calibration

1) identify



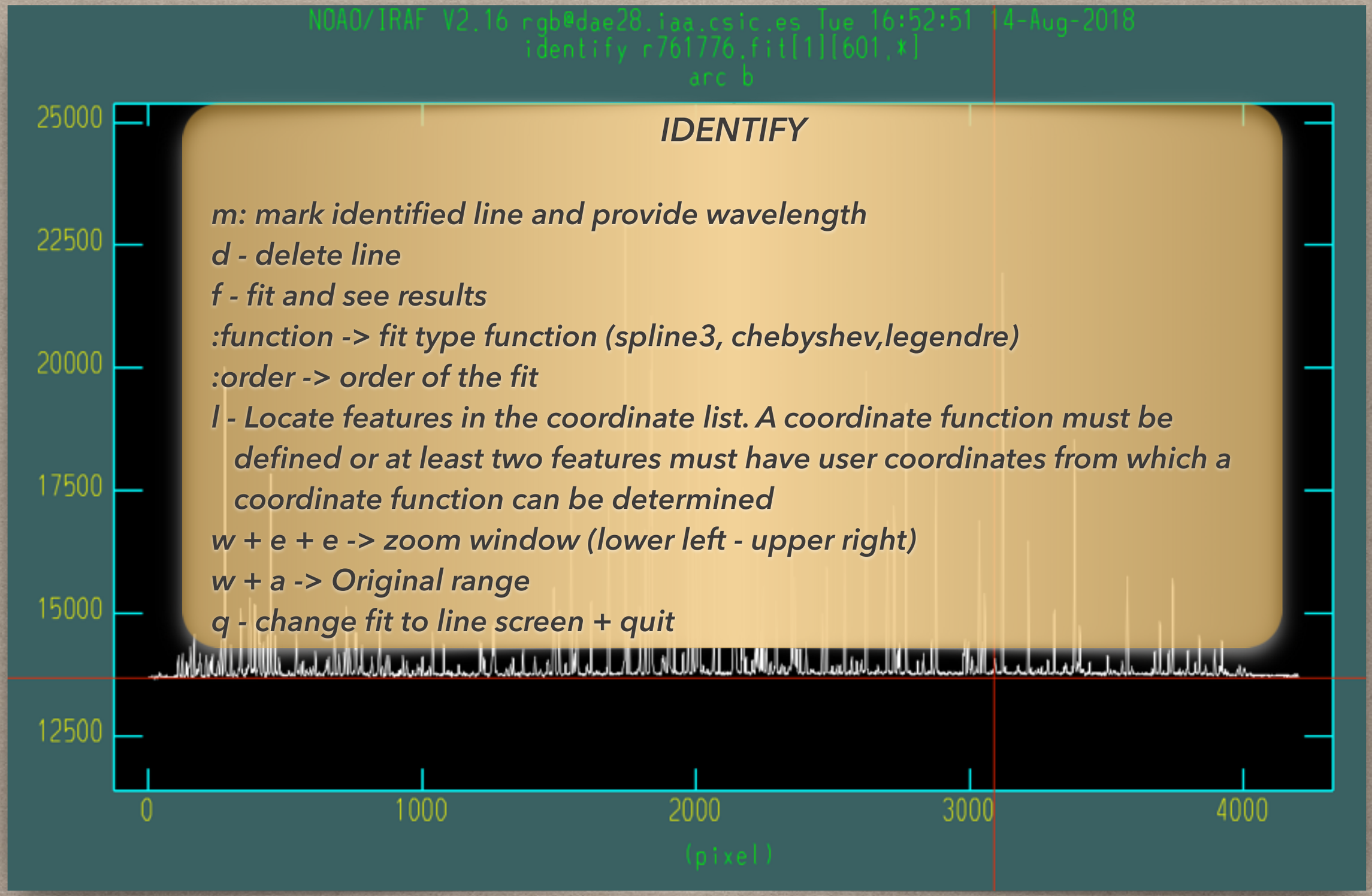
Wavelength calibration

1) identify



Wavelength calibration

1) identify

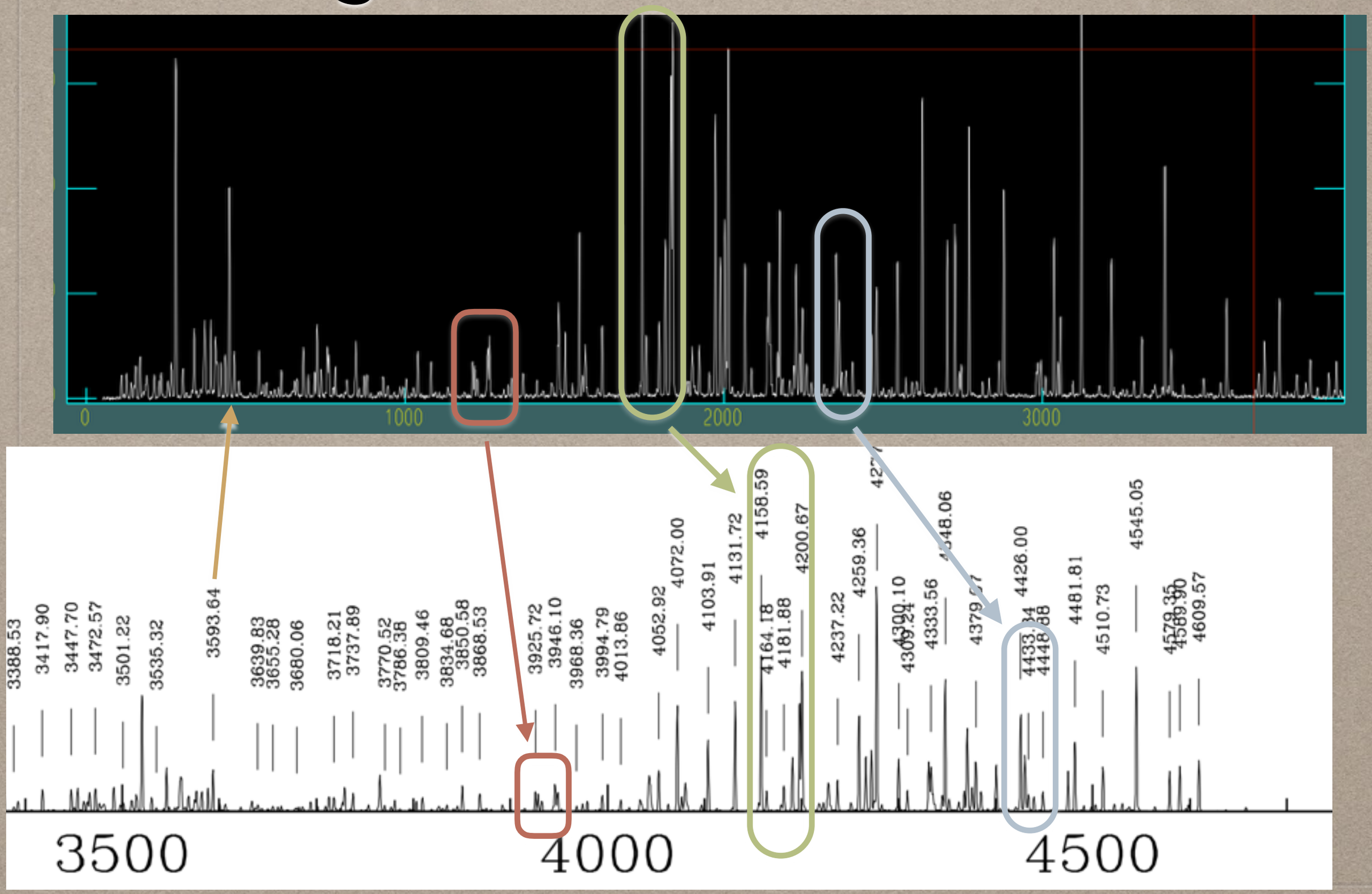


IDENTIFY

m: mark identified line and provide wavelength
d - delete line
f - fit and see results
:function -> fit type function (spline3, chebyshev,legendre)
:order -> order of the fit
l - Locate features in the coordinate list. A coordinate function must be defined or at least two features must have user coordinates from which a coordinate function can be determined
w + e + e -> zoom window (lower left - upper right)
w + a -> Original range
q - change fit to line screen + quit

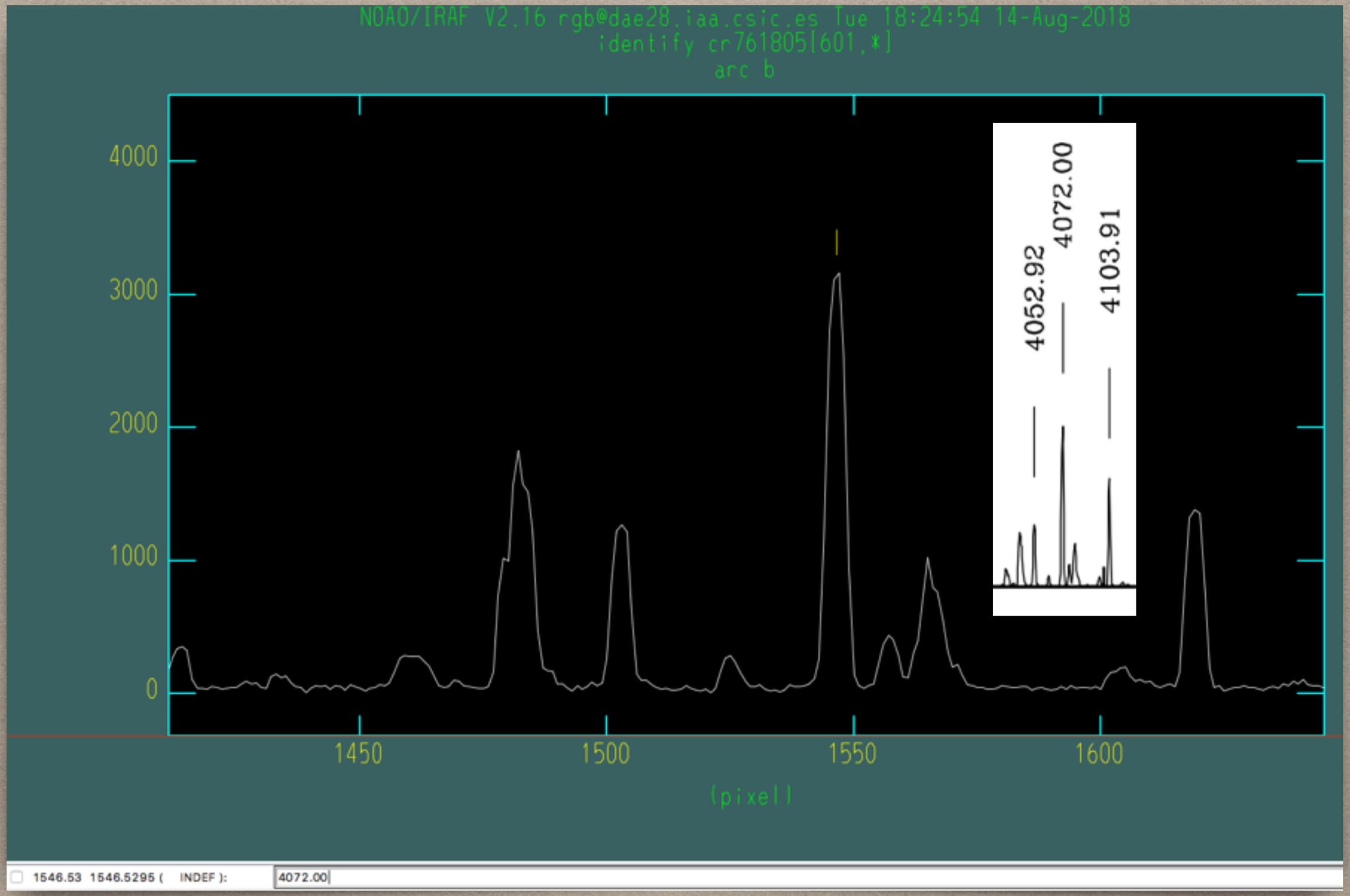
Wavelength calibration

1) identify



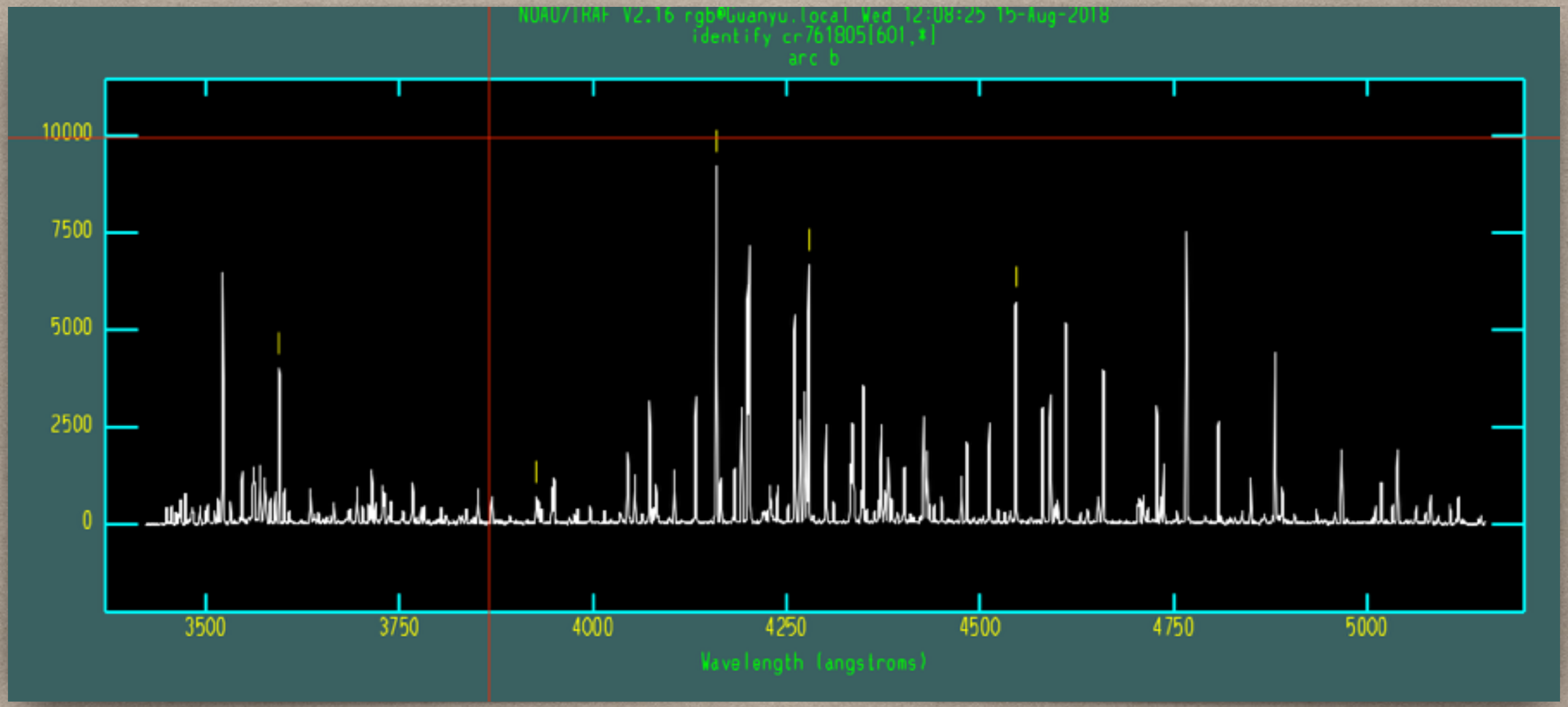
Wavelength calibration

1) identify



Wavelength calibration

1) identify



iraf/noao/lib/linelists/

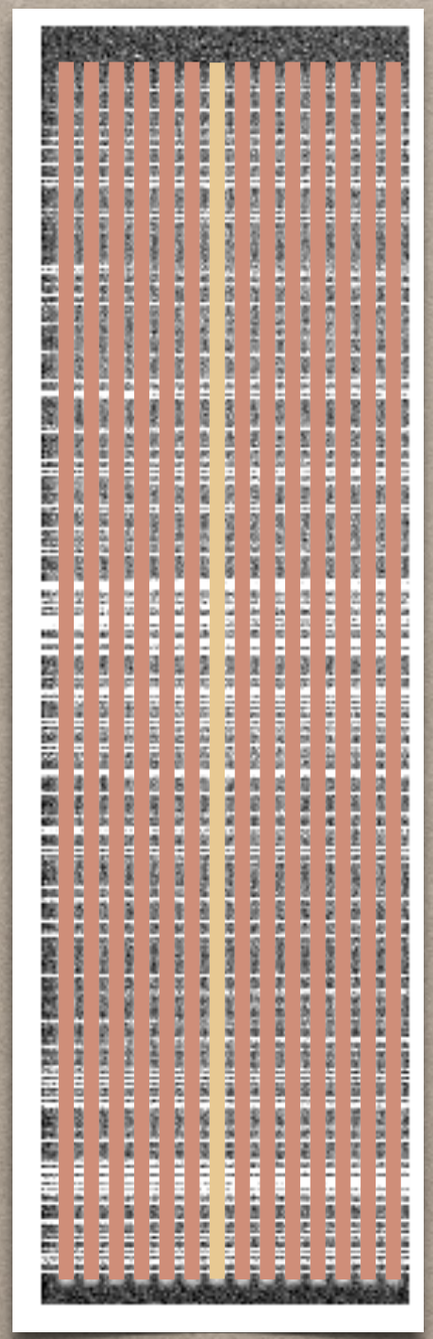
Wavelength calibration

1) identify

```
# Tue 18:45:48 14-Aug-2018
begin   identify cr761805[601,*]
        id      cr761805
        task    identify
        image   cr761805[601,*]
        units   Angstroms
        features        11
                464.53 3535.63177  3535.3196   3.4 1 1
               1212.13 3923.95669  3925.7188   3.4 1 1
               1259.47 3945.93392  3946.0971   3.4 1 1
               1743.99 4161.01217  4158.5905   3.4 1 1
               1839.95 4202.33038  4200.6745   3.4 1 1
               2014.81 4277.40403  4277.5282   3.4 1 1
               2174.88 4346.44498   4348.064   3.4 1 1
               2351.78 4423.84797  4426.0011   3.4 1 1
               2545.07 4510.66267  4510.7332   3.4 1 1
               2622.99 4546.55432  4545.0519   3.4 1 1
               2769.08  4615.5621  4609.5673   3.4 1 1
        function legendre
        order 4
        sample *
        naverage 1
        niterate 0
        low_reject 3.
        high_reject 3.
        grow 0.
        coefficients    8
                2.
                4.
                464.5339355468751
                2622.989501953125
                4062.928857977182
                493.2099601441272
                -21.83581532452953
                12.25131442954602
```

Directory: database/idcr761805

Wavelength calibration



1) identify

2) reidentify

2-D identification

3) fitcoords

Wavelength calibration

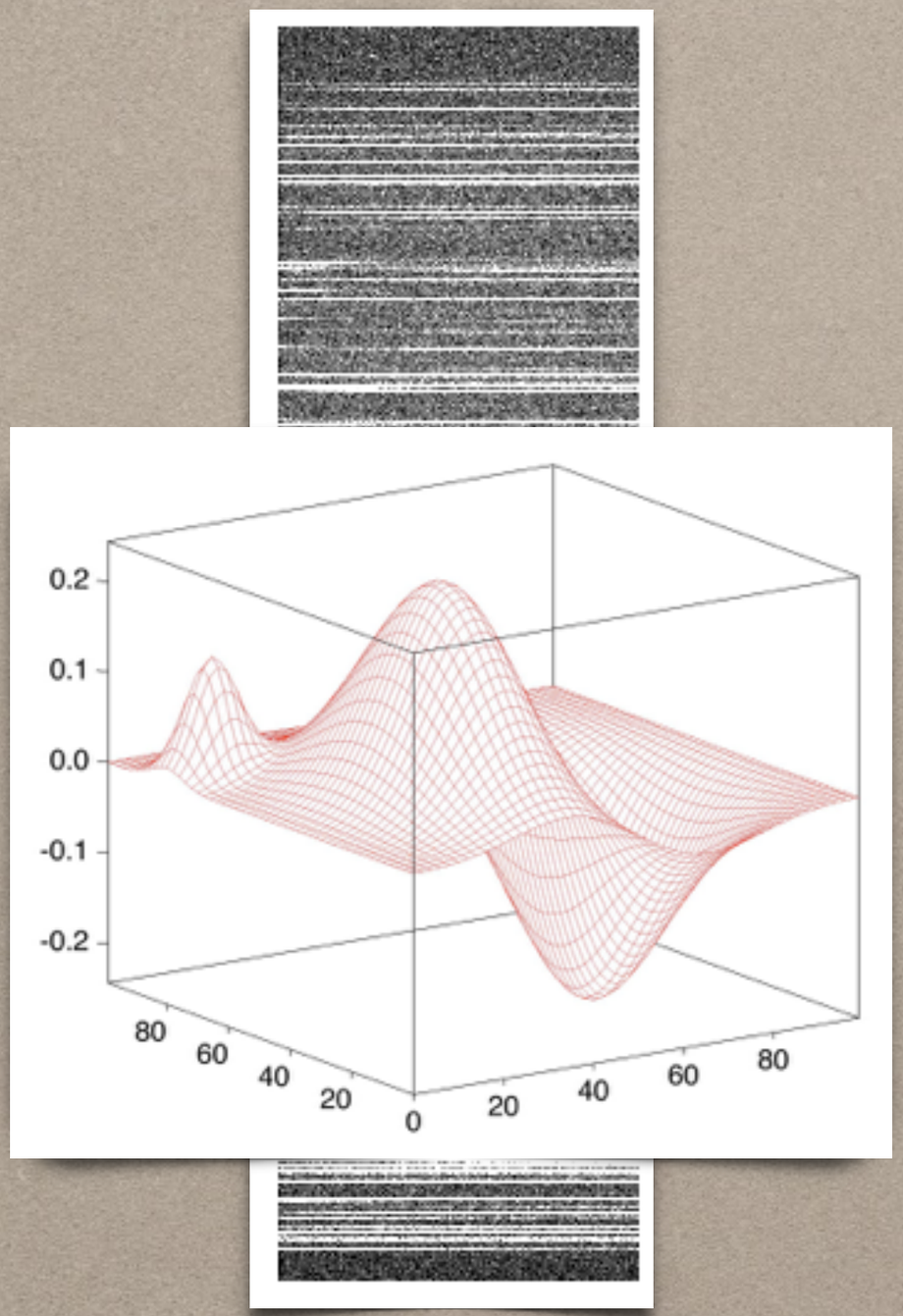
2) reidentify

2-D identification

```
--> lpar reidentify
    reference =                 Reference image
       images =                 Images to be reidentified
       answer = "yes"           Fit dispersion function interactively?
        crval =                 Approximate coordinate (at reference pixel)
        cdelt =                 Approximate dispersion
 (interactive = "no")           Interactive fitting?
     (section = "middle line")  Section to apply to two dimensional images
      (newaps = yes)            Reidentify apertures in images not in reference?
    (override = no)             Override previous solutions?
       (refit = yes)            Refit coordinate function?

       (trace = no)             Trace reference image?
        (step = "10")           Step in lines/columns/bands for tracing an image
        (nsum = "10")           Number of lines/columns/bands to sum
       (shift = "0.")           Shift to add to reference features (INDEF to search)
      (search = 0.0)            Search radius
       (nlost = 0)              Maximum number of features which may be lost

     (cradius = 5.0)            Centering radius
   (threshold = 0.0)            Feature threshold for centering
 (addfeatures = no)             Add features from a line list?
   (coordlist = "linelists$idhenear.dat") User coordinate list
       (match = -3.0)           Coordinate list matching limit
 (maxfeatures = 50)             Maximum number of features for automatic identification
      (minsep = 2.0)            Minimum pixel separation
```

images = cr761805.fit[1] *reference = cr761805.fit*

Wavelength calibration



1) identify

2) reidentify

3) fitcoords

Fit a surface $\lambda = f(x, y)$

Wavelength calibration

3) fitcoords

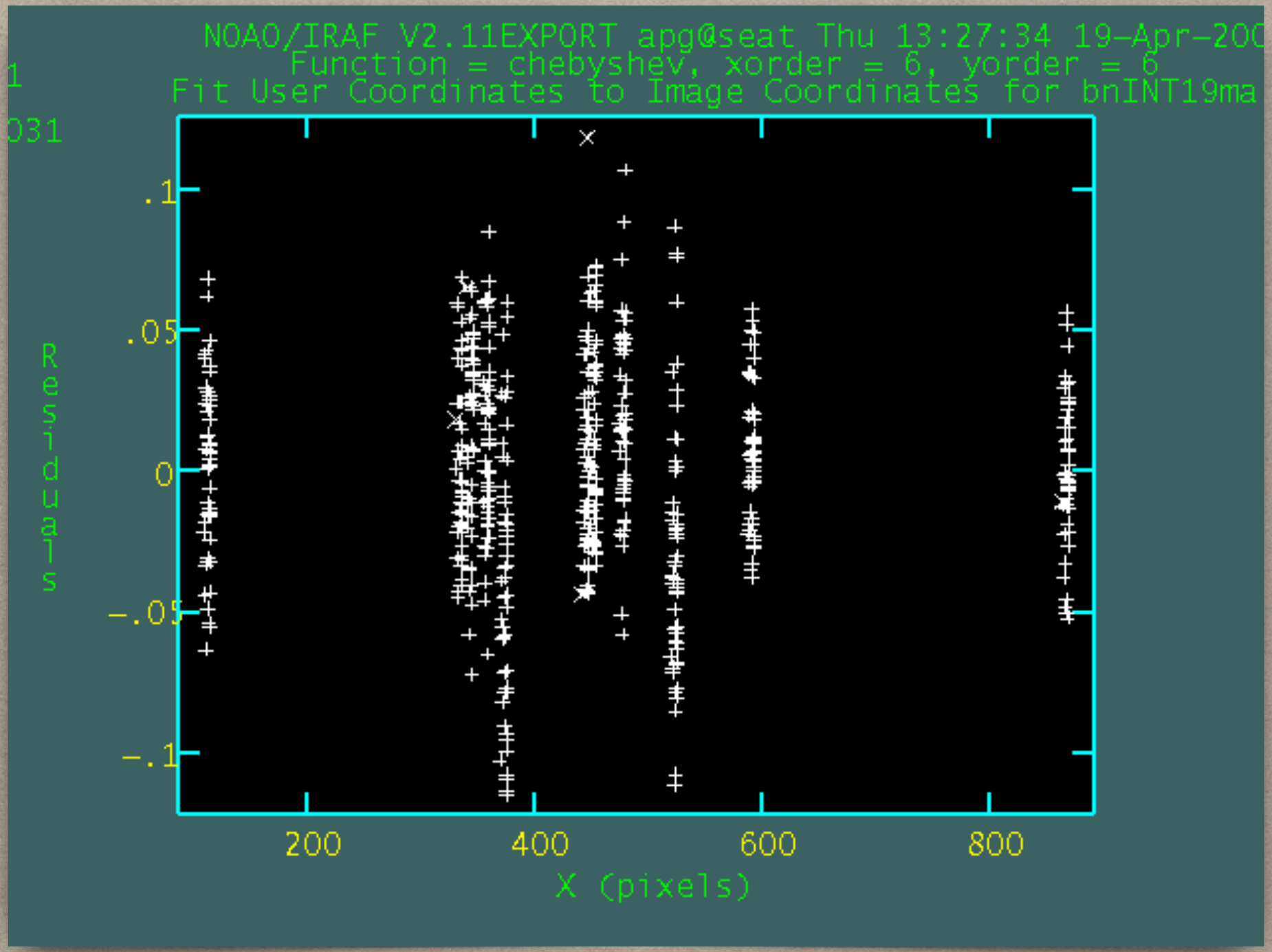
Fit a surface $\lambda = f(x, y)$

images = cr761805

Wavelength calibration

3) fitcoords

Fit a surface λ = f(x, y)



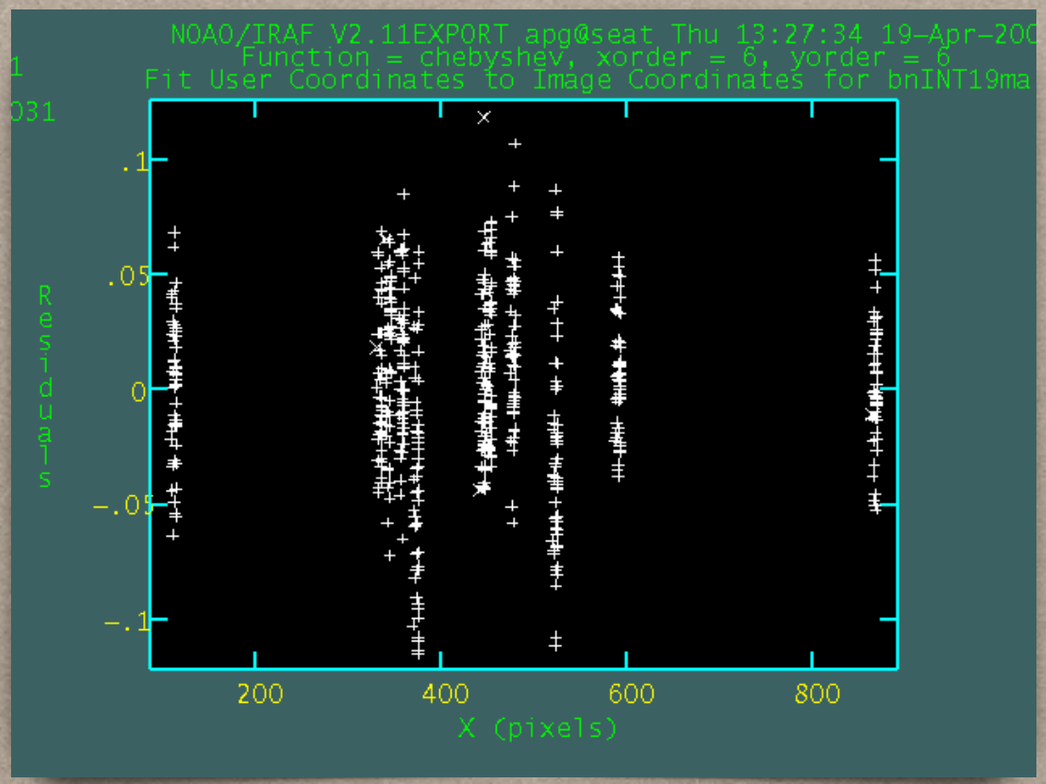
Wavelength calibration

3) fitcoords

Fit a surface λ = f(x, y)

FITCOORDS

d + p - delete a point
:function -> fit type function (spline3, chebyshev,legendre)
:xorder -> order of the fit in x
:yorder -> order of the fit in y
f - fit
q - change fit to line screen + quit



order not very high (4,5,6)

Residuals without structure (order +1)

Wavelength calibration

3) fitcoords

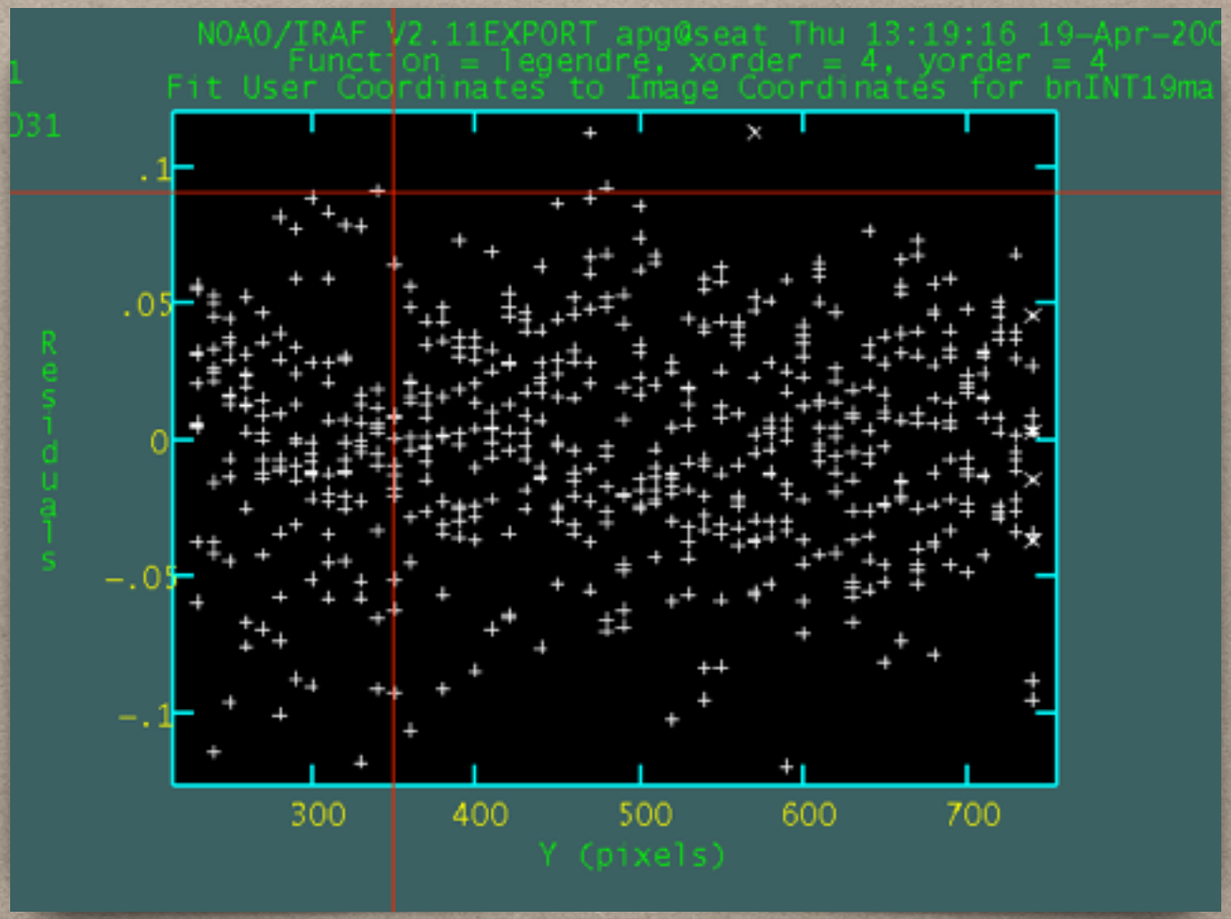
Fit a surface λ = f(x, y)

FITCOORDS (residuals)

x - define plot variable of axis x of the plot
y - define plot variable of axis y of the plot
r - plot

(x, y, z, r, s)

x + y
y + r
r



Wavelength calibration

identify
reidentify
fitcoords

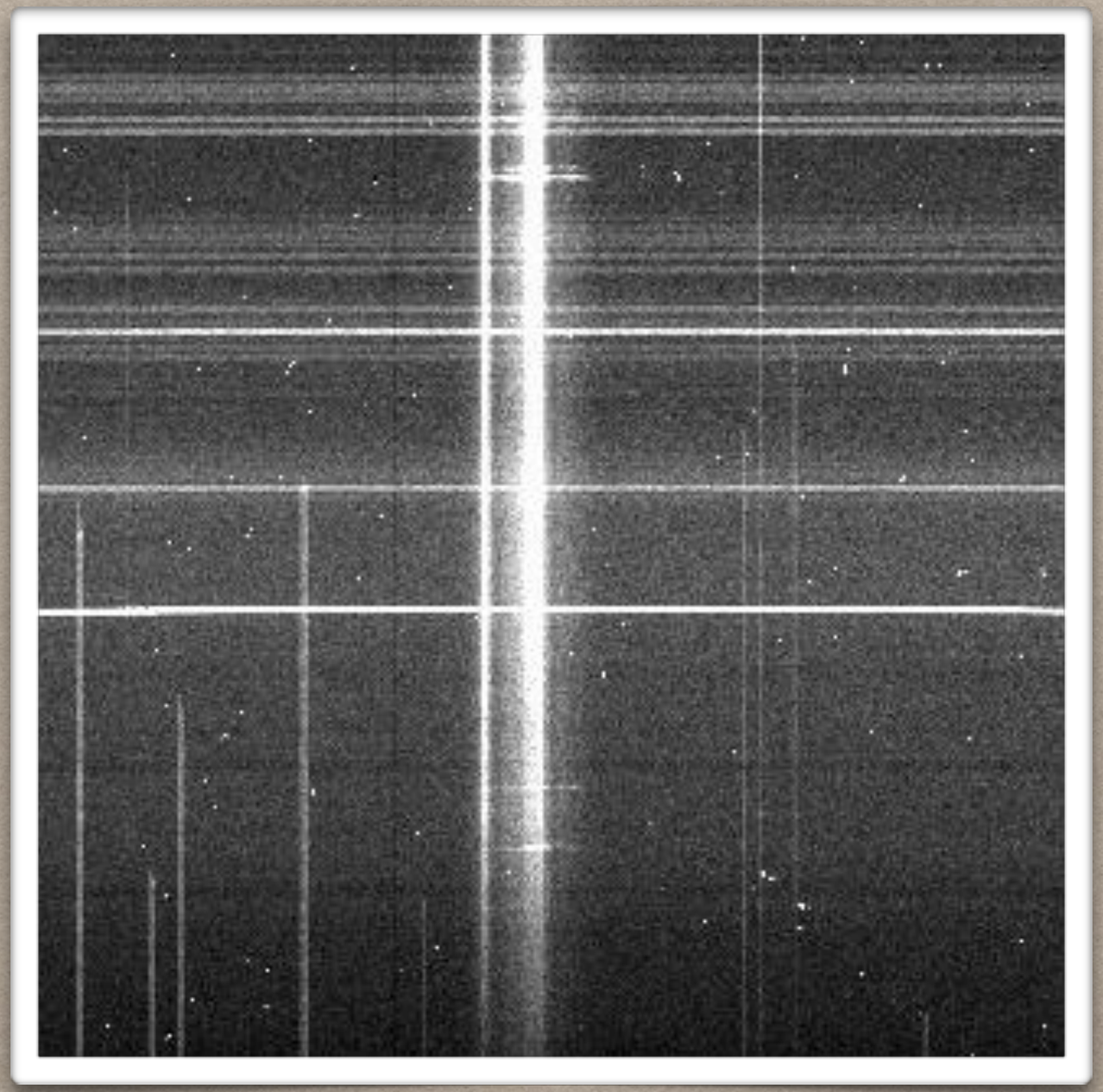
```
def wlsolution(arc, arc_ref=None, iden=True, reiden=True, section="middle column", coor='linelists$cuar.dat',
        database='database', fwidth=3.4, function='legendre', order=4, nsum=10, overrid='yes', refit='no',
        trace='yes', nlost=20, step=10, xorder=3, yorder=4, rinterac='no', finterac='yes',
        fitc=True, suffix=None):
    # twodspec.longslit
    # Measure the FWHM (fwidth) in a reduced arc with splot 'k' (file.fits[1][1000,*])

    lw_tasks = ['identify', 'reidentify', 'fitcoords']
    unlearn_list(lw_tasks)

    if suffix is not None:
        arc = '%s%s' % (arc, suffix)

    if arc_ref is None:
        arc_ref = arc

    arc_ref = arc_ref.split('[')[0]

    if iden:
        # 'm': select lines | 'l': searching in a list | 'f': final fit
        identify(arc, section=section, database=database, fwidth=fwidth, coordli=coor,
            function=function, order=order, nsum=nsum)

    if reiden:
        reidentify(images=arc, reference=arc_ref, interac=rinterac, section=section, overrid=overrid,
            refit=refit, trace=trace, nlost=nlost, step=step, nsum=nsum, coordlist=coor)

    if fitc:
        arc = '.'.join(arc.split('.')[0:-1])
        fitcoords(arc, fitname="", interac=finterac, xorder=xorder, yorder=yorder, function=function)

    unlearn_list(lw_tasks)
```

Wavelength calibration

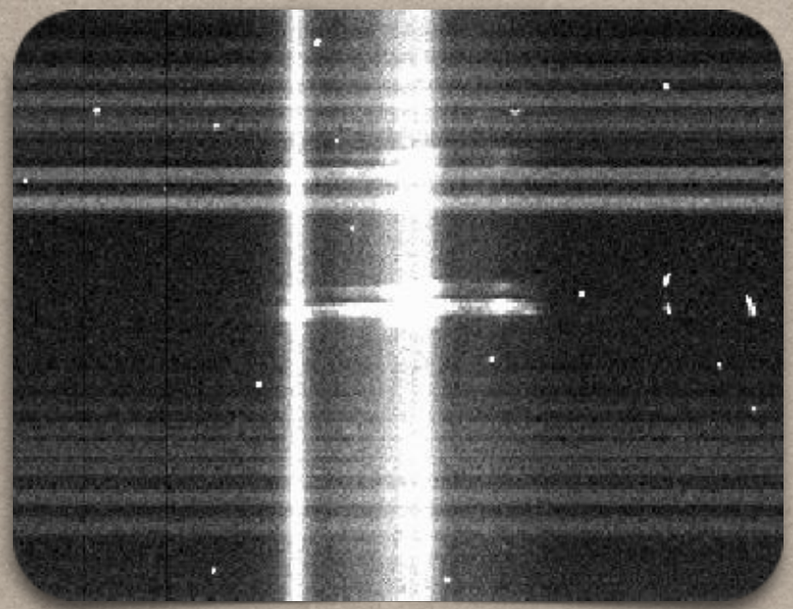
identify
reidentify
fitcoords

```
idarc  = (tlog['object'] == 'arc')
arcref = 'c%s' % tlog['file'][idarc][0]

fwidth = 3.4

# Find solution in one arc file
wlsolution(arcref, fwidth=fwidth)
```

Cosmic rays

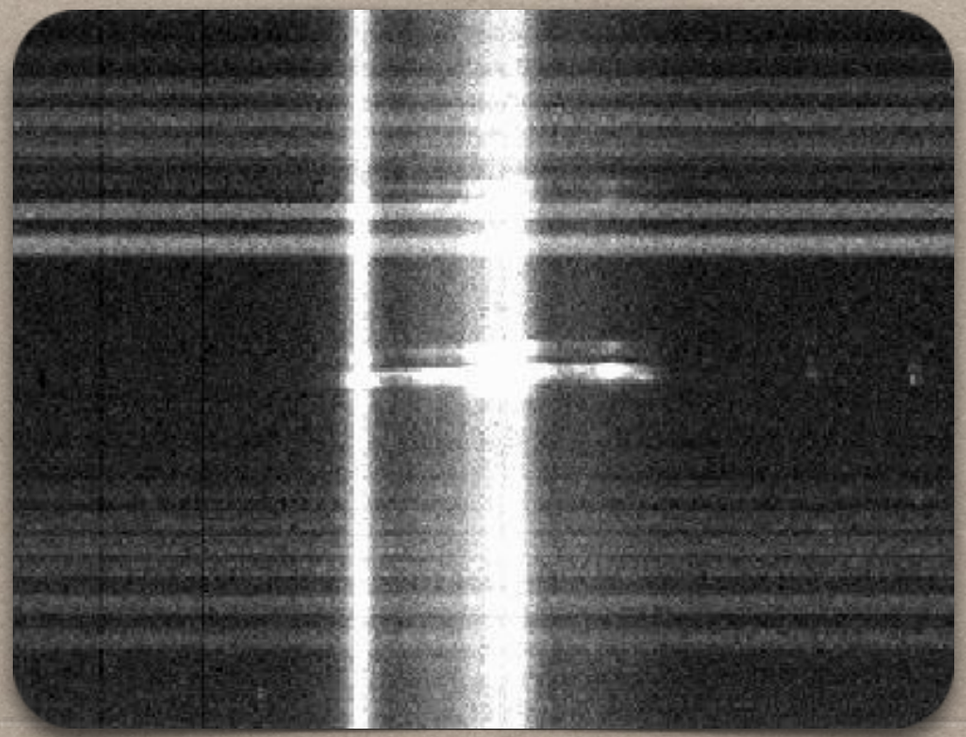


Cosmic rays



Spectrum with cosmic rays

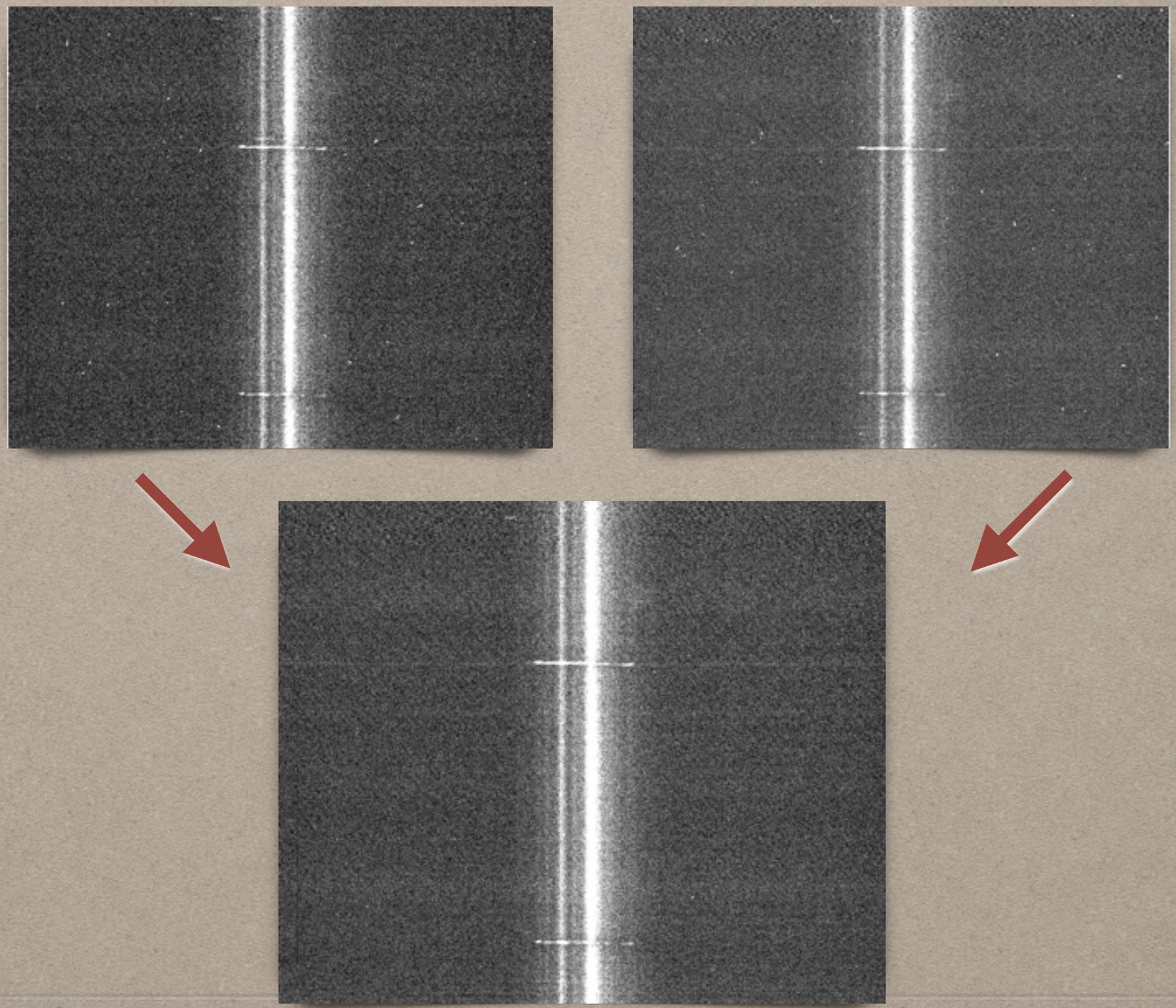
Cosmic rays mask



Cleaned image

Cosmic rays

imcombine



Cosmic rays

imcombine

```
def combine(lfits, outfits, rej='crreject', combine='average', scale='exposure', prefix=None,
                weight='exposure', nlow=1, expname='EXPTIME', rdnoise=0.0,
                gain=1.0, suffix=None, input_files='tmp_input.list', delete_list=True):

    wfile(input_files, lfits, suffix=suffix, prefix=prefix)
    delfiles(outfits)

    unlearn('imcombine')
    imcombine('@'+input_files, output=outfits, combine=combine, reject=rej, scale=scale,
            weight=weight, expname=expname, rdnoise=rdnoise, gain=gain, nlow=nlow)
    unlearn('imcombine')

    if delete_list:
        delfiles(input_files)
```

Cosmic rays

imcombine

```
# ------------ Combine science and standard star images to remove cosmic rays --------------

idobj  = (tlog['object'] == 'GALAXY')
imgal  = 'galaxy.fits'

idstd1 = (tlog['object'] == 'bd33')
std1   = 'bd33.fits'

idstd2 = (tlog['object'] == 'bd25')
std2   = 'bd25.fits'

for selection, name in zip([idobj, idstd1, idstd2], [imgal, std1, std2]):
    combine(tlog['file'][selection], name, prefix='c')
```

Cosmic rays

lacos_spec

```
# Remove cosmic rays of single images -------------------------
def rmcosmic(lfits, xorder=5, yorder=5, sigfrac=2.0, niter=5, objlim=2, sigclip=3.0,
                    fmt_in=None, fmt_out='cr%s'):

    lfits = cklist(lfits)
    fmt_in = '%s' if fmt_in is None else fmt_in

    for fits in lfits:
        fits   = fmt_in  % fits
        output = fmt_out % fits
        mask   = 'mask%s' % fits
        delfiles([mask, output])

        unlearn('lacos_spec')
        lacos_spec(fits, output, mask, gain=gain, readn=rdnoise, xorder=xorder, yorder=yorder,
                sigclip=sigclip, sigfrac=sigfrac, objlim=objlim, niter=niter)
        unlearn('lacos_spec')

        delfiles(mask)
```

Apply wavelength solution

transfrom

Science objets, standard stars, …

```
-->
--> lpar transform
        input =                 Input images
       output =                 Output images
     fitnames =                 Names of coordinate fits in the database
      (minput = "")             Input masks
     (moutput = "")             Output masks
    (database = "database")     Identify database
  (interptype = "spline3")      Interpolation type
           (x1 = INDEF)         Output starting x coordinate
           (x2 = INDEF)         Output ending x coordinate
           (dx = INDEF)         Output X pixel interval
           (nx = INDEF)         Number of output x pixels
        (xlog = no)             Logarithmic x coordinate?
           (y1 = INDEF)         Output starting y coordinate
           (y2 = INDEF)         Output ending y coordinate
           (dy = INDEF)         Output Y pixel interval
           (ny = INDEF)         Number of output y pixels
        (ylog = no)             Logarithmic y coordinate?
        (flux = yes)            Conserve flux per pixel?
       (blank = INDEF)          Value for out of range pixels
    (logfiles = "STDOUT,logfile") List of log files
        (mode = "al")
-->
```

Apply wavelength solution

transfrom

```
def transf(lfits, fitnames, interp='linear', database='database', flux='yes', y1='INDEF', suffix=None,
        input_files='tmp_input.list', output_files='tmp_output.list', dispaxis=None, **kwargs):

    fitnames = '.'.join(fitnames.split('.')[0:-1])

    if isinstance(lfits, str) and input_files is None:
        input_files = lfits
        lfits = np.loadtxt(input_files, dtype=np.str, unpack=True)
    else:
        lfits = cklist(lfits)
        wfile(input_files, lfits, suffix=suffix)

    wfits = ['l%s' % fits.split('[')[0] for fits in lfits]
    wfile(output_files, wfits)
    delfiles(wfits)

    if dispaxis is not None:
        for fits in lfits:
            hedit(fits, 'DISPAXIS', dispaxis, add='yes', verify='no')

    unlearn('transform')
    transform(input='@'+input_files, output='@'+output_files, fitnames=fitnames, interp=interp,
            database=database, flux=yes, y1=y1, **kwargs)
    unlearn('transform')

    delfiles([input_files, output_files])
```

Apply wavelength solution

transfrom

```
idobj  = (tlog['object'] == 'GALAXY')
imgal  = 'galaxy.fits'

idstd1 = (tlog['object'] == 'bd33')
std1   = 'bd33.fits'

idstd2 = (tlog['object'] == 'bd25')
std2   = 'bd25.fits'

for selection, name in zip([idobj, idstd1, idstd2], [imgal, std1, std2]):
    combine(tlog['file'][selection], name, prefix='c')

idarc  = (tlog['object'] == 'arc')
arcref = 'c%s[1]' % tlog['file'][idarc][0]

# Find solution in one arc file
wlsolution(arcref)

# Apply solution to science images
lobj     = [imgal, std1, std2]
fitnames = arcref

transf(lobj, fitnames)
```

Reduction steps

1) *Pre-processing (correct frames):*
   * *Master Bias*
   * *Master Flat*

Frame - <bias>

<flat> - <bias>

2) *Wavelength calibration:*
   * identify
   * reidentify
   * fitcoords

3) **Flux calibration & sky subtraction:**
   * **standard**
   * **sensfunc**
   * **calibrate**

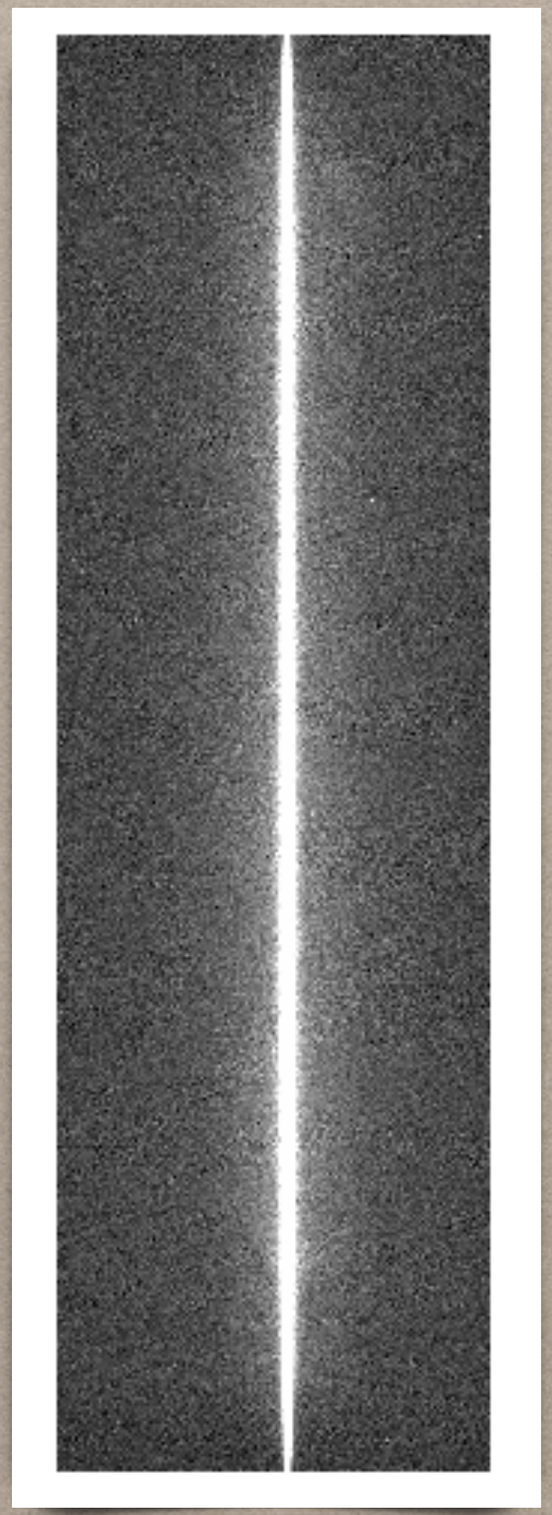
Prepare for flux calibration

setairmass

```
def setair(files, observa='lapalma', equinox='EPOCH', ut='UTSTART',
           input_files='tmp_input.list'):

    files = cklist(files)
    wfile(input_files, files)

    unlearn('setairmass')
    setairmass(images='@'+input_files, observa=observa, equinox=equinox, ut=ut)
    unlearn('setairmass')

    delfiles(input_files)

lobj    = [imgal, std1, std2]

setair(lobj)
```

Extract standard start spectrum

apall

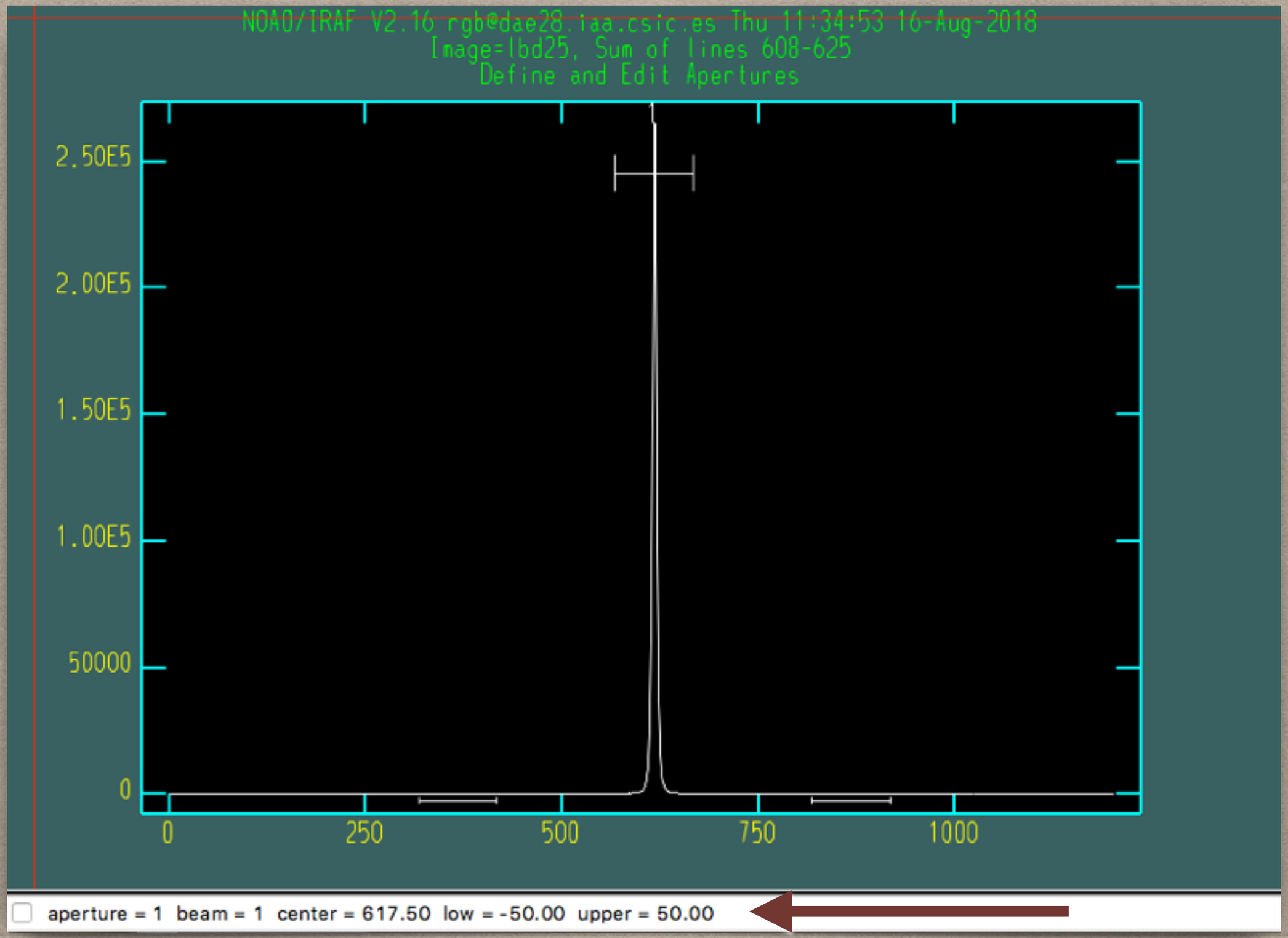


Extract standard start spectrum

apall

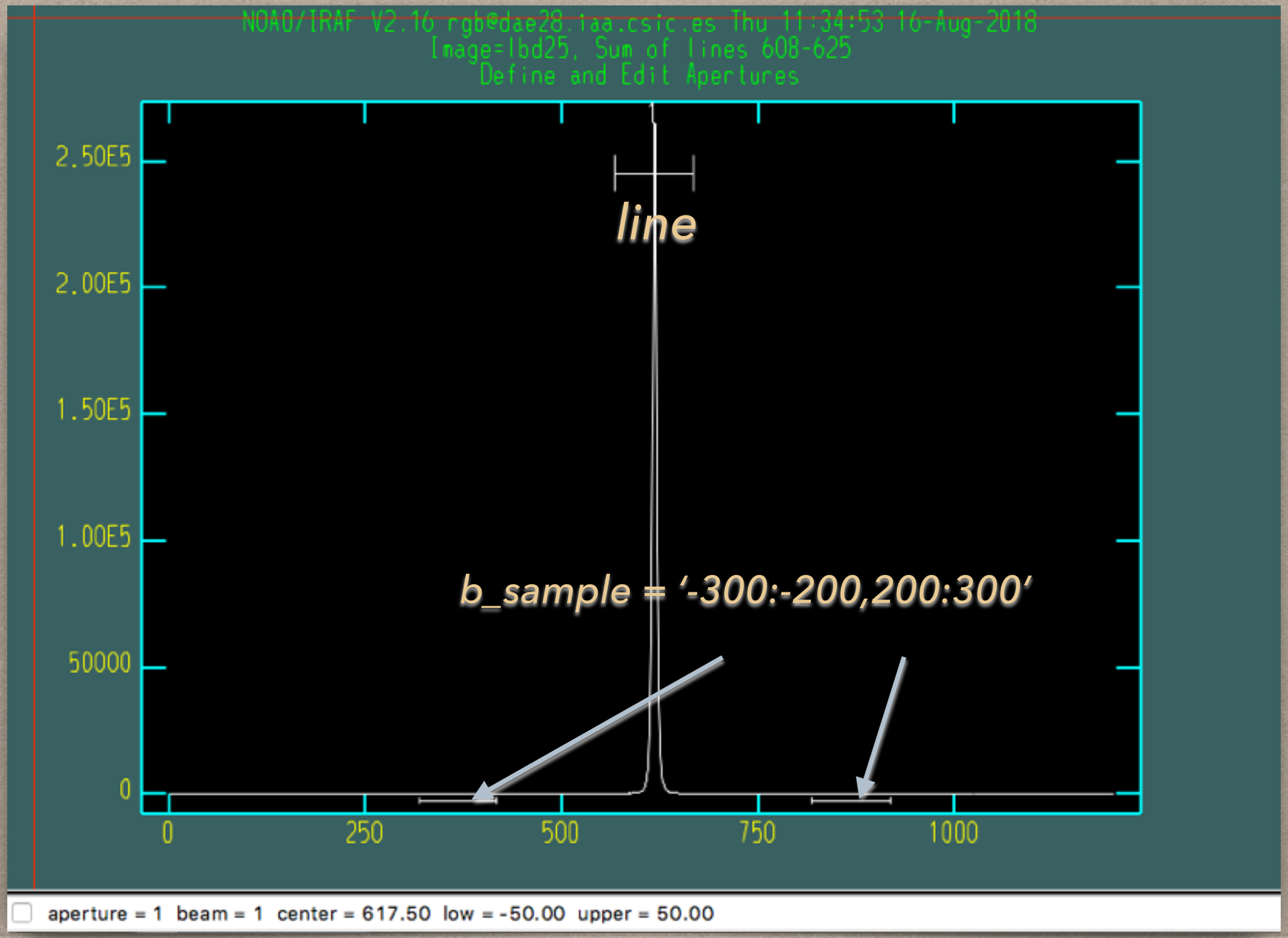
Extract standard start spectrum

apall



Extract standard start spectrum

apall



Extract standard start spectrum

apall

```
def extract_aperture(fits, line, lower=-5, upper=5, b_sample='-10:-6,6:10', outfits=None, fmt_out='ap%s',
                     fmt_ap='%s.000?.fits', database='database/', rdnoise=0.0, gain=1.0, std=False, **kwargs):

    # Default parameter dictionary
    dapall = dict(format='onedspec', nsum=18, t_function='spline3', t_order=5,   recenter='no'', b_naverage=-50,
                  resize='no', gain=gain,  readnoise=rdnoise, interac='yes', extras='yes', trace='no',
                  references="", apertures="1", b_function='legendre')

    # Default values for standard or object
    if std:
        dextra = dict(background='fit', weights='variance', clean='yes')
    else:
        dextra = dict(background='none', weights='none', clean='no')
    dapall.update(dextra)

    # Overwrite default values
    dapall.update(kwargs)

    # outfits name
    if outfits is None:
        outfits = fmt_file(fmt_out, fits)

    # Delete output file
    delfiles(outfits)

    # Delete aperture files created from previous extractions
    ap_file = fmt_ap % '.'.join(outfits.split('.')[0:-1])
    list_ap_files = glob(ap_file)
    delfiles(list_ap_files)

    # Delete aperture database for this file
    root_name_ap = '.'.join(outfits.split('.fit')[0:-1])
    db_files = glob(os.path.join(database, root_name_ap))
    delfiles(db_files)

unlearn('apall')
apall(input=fits, output=outfits, line=line, lower=lower, upper=upper, b_sample=b_sample, **dapall)
unlearn('apall')
```

Extract standard start spectrum

apall

```
lstd = OrderedDict()

lstd['lbd25.fits'] = dict(line=617, lower=-50, upper=50, b_sample='-300:-200,200:300')
lstd['lbd33.fits'] = dict(line=642, lower=-50, upper=50, b_sample='-300:-200,200:300')

for std in lstd:
    extract_aperture(std, std=True, **lstd[std])
```

Extract standard start spectrum

apall

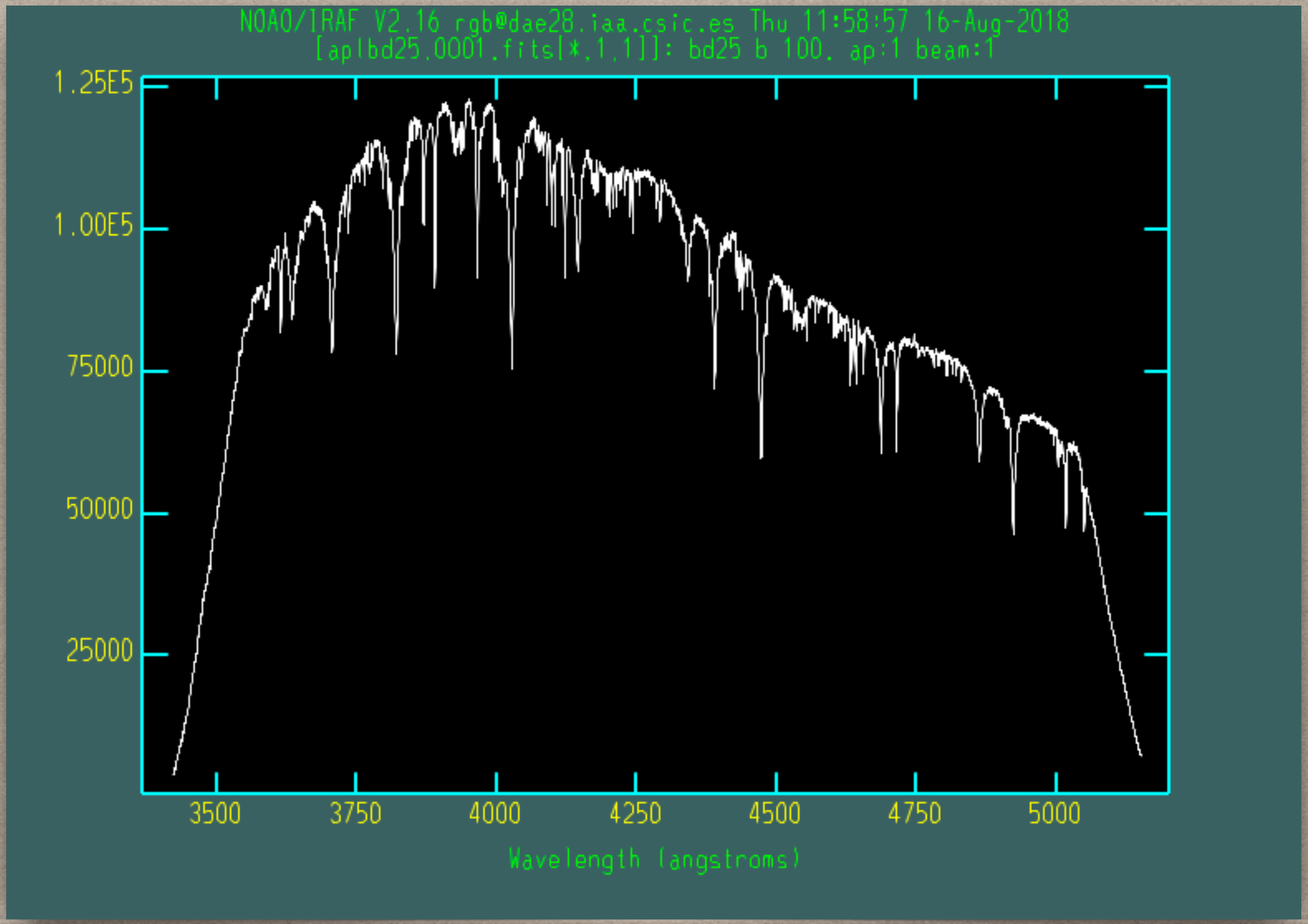
```
# Thu 11:42:18 16-Aug-2018
begin   aperture lbd25 1 617.5008 617.
        image   lbd25
        aperture        1
        beam    1
        center  617.5008 617.
        low     -50. -616.
        high    50. 3334.
        background
                xmin -300.
                xmax 300.
                function legendre
                order 1
                sample -300:-200,200:300
                naverage -50
                niterate 0
                low_reject 3.
                high_reject 3.
                grow 0.
      axis    1
      curve   5
                2.
                1.
                1.
                3951.
                0.
```

database/aplbd25

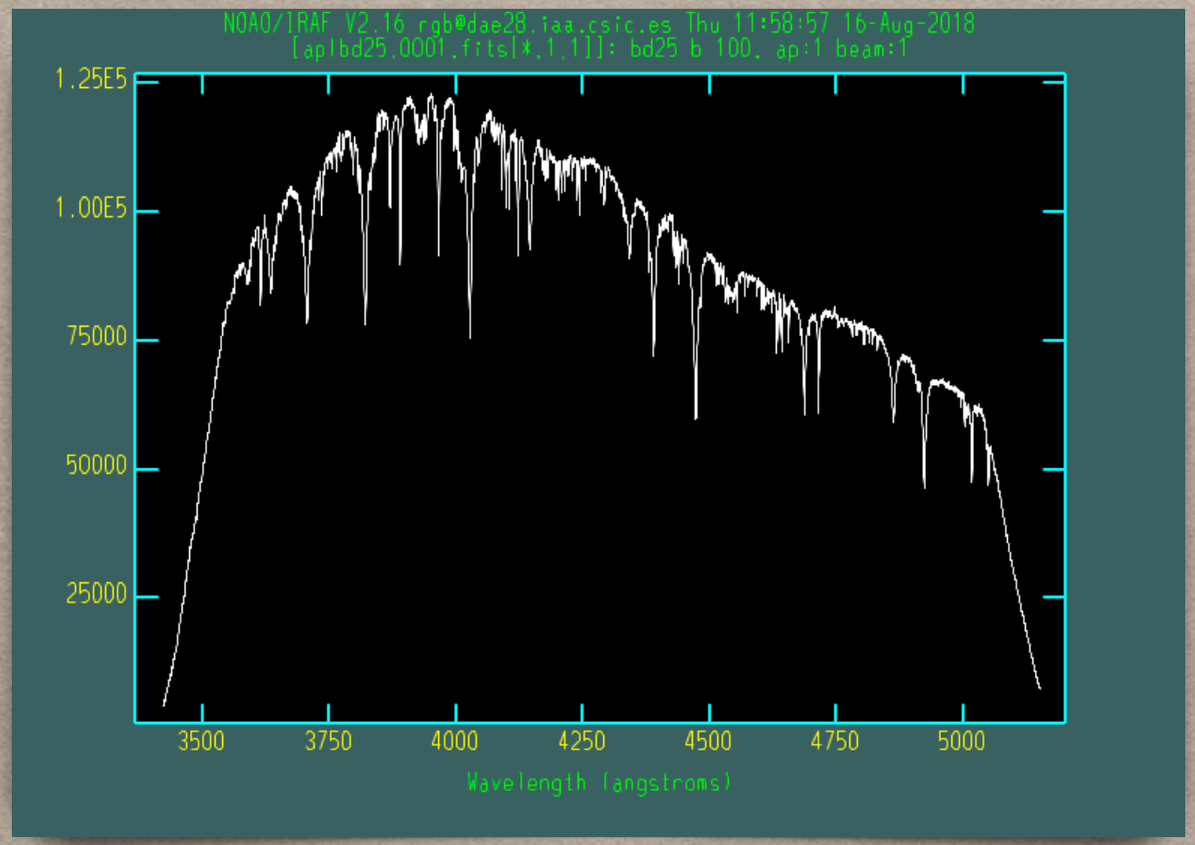
Extract standard start spectrum

apall

apldb25.0001.fits



Flux calibration



1) standard

2) sensfunc

3) calibrate

Flux calibration

1) standard
2) sensfunc
3) calibrate

The reference will be spectra of some standard stars observed during the same night. The aim is to calibrate the CCD chip response, spectrograph +telescope throughput and allow for atmospheric extinction.

a) take from a tabular compilation the energy distribution of the standard star you observed

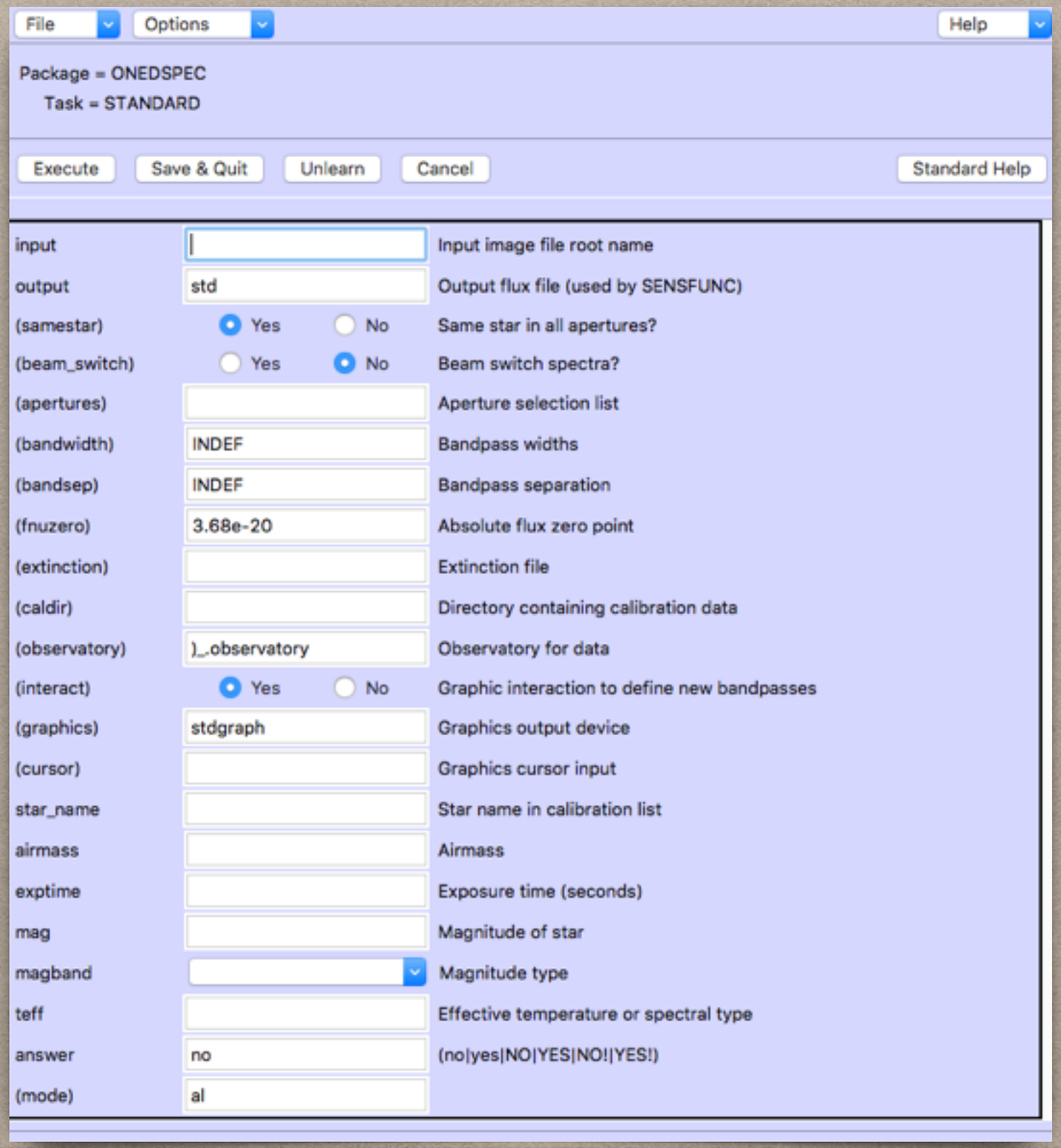
b) correct this energy distribution for wavelength-dependent atmospheric extinction

c) compare it to the energy distribution of the observed spectrum

d) derive from such a comparison the function that gives the response of your system for every wavelength.

Flux calibration

1) standard



Flux calibration

1) standard

** La Palma extinction:*

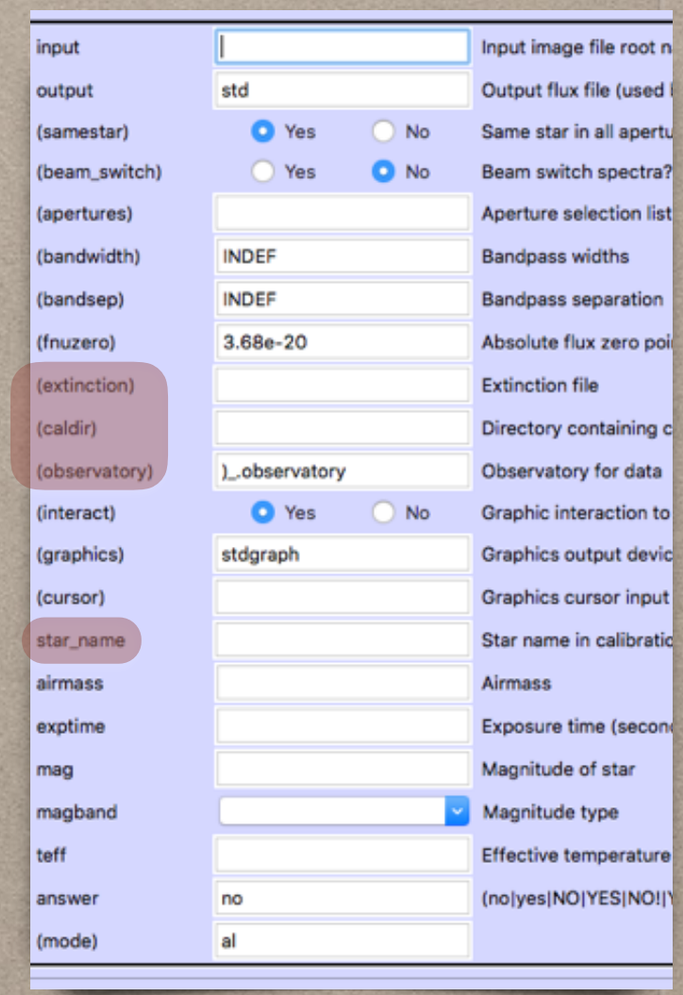
http://www.ing.iac.es/Astronomy/observing/conditions/wlext.html

extinct = “lapalma_extinction.dat”
observatory = 'lapalma'

** Directory of the standard star file:*

iraf/noao/lib/onedstds/iidscal/bd253941.dat
iraf/noao/lib/onedstds/iidscal/bd332642.dat

caldir = “onedstds$iidscal/"
star_name = ‘bd253941' (‘bd332642')



Flux calibration

1) standard

extinct = “lapalma_extinction.dat”

```
# ORM Observatory extinction with wavelength

 3000   3.7150
 3010   3.3105
 3020   2.9638
 3030   2.6782
 3040   2.4211
 3050   2.2140
 3060   2.0423
 3070   1.9001
 3080   1.7822
 3090   1.6826
 3100   1.5961
 3110   1.5169
 3120   1.4393
 3130   1.3579
 3140   1.2670
 3150   1.1610
....
```

Flux calibration

1) standard

```
(astroconda) $ ls anaconda/envs/astroconda/iraf/noao/lib/onedstds/iidscal/

40erib.dat        feige56.dat       he3.dat           kopff27.dat       names.men
bd253941.dat      feige92.dat       hiltner102.dat    l13633.dat        ross627.dat
bd284211.dat      feige98.dat       hiltner600.dat    l1151234b.dat     ross640.dat
bd332642.dat      gl91b2b.dat       hz14.dat          l74546a.dat       sa29130.dat
bd404032.dat      g4718.dat         hz15.dat          l8702.dat         standards.men
bd82015.dat       g9937.dat         hz2.dat           l93080.dat        ton573.dat
feige110.dat      gd140.dat         hz29.dat          l97030.dat        wolf1346.dat
feige15.dat       gd190.dat         hz4.dat           lb1240.dat        wolf485a.dat
feige24.dat       grw705824.dat     hz43.dat          lb227.dat
feige25.dat       grw708247.dat     hz44.dat          lds235b.dat
feige34.dat       grw738031.dat     hz7.dat           lds749b.dat
```

Flux calibration

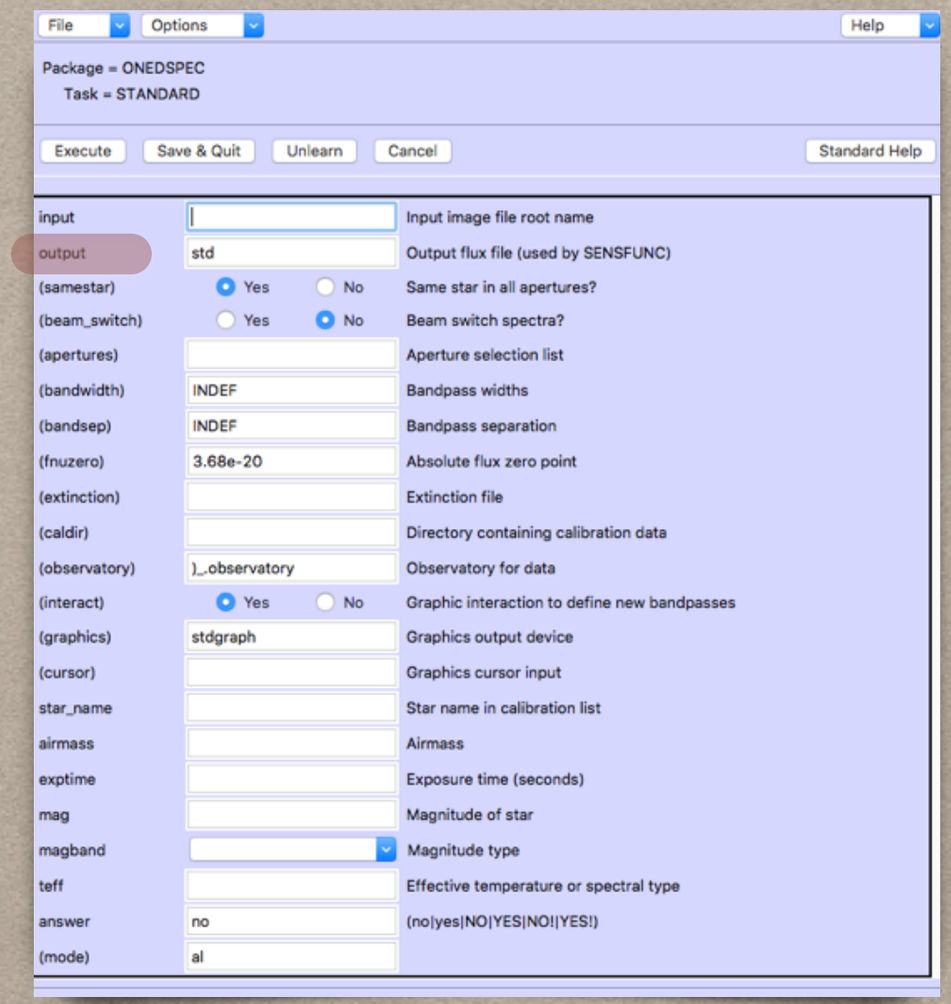
1) standard

bd253941.dat

```
# BD+25 3941
3200. 12.11 49.
3250. 12.05 49.
3300. 11.98 49.
3350. 11.91 49.
3400. 11.85 49.
3450. 11.82 49.
3500. 11.77 49.
3571. 11.73 49.
3636. 11.7   49.
3704. 11.63 49.
3862. 11.26 49.
4036. 11.16 49.
4167. 11.09  49.
4255. 11.04 49.
4464. 10.96 49.
4566. 10.87 49.
4785. 10.79 49.
5000. 10.69 49.
5263. 10.49 49.
......
```

Flux calibration

1) standard

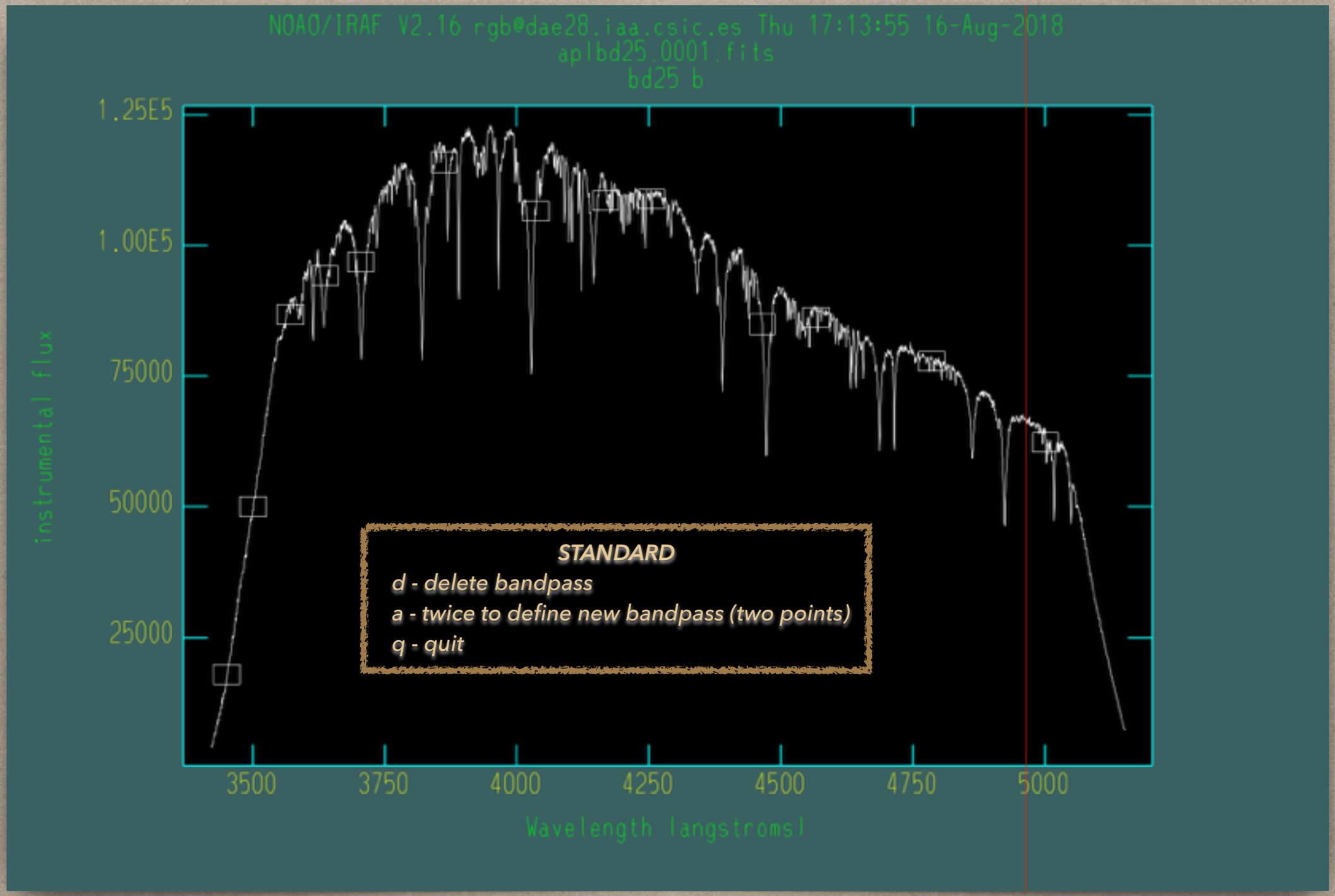


Run standard once for each exposure of a standard star

The results will be appended to the std file

Flux calibration

1) standard



Flux calibration

1) standard

output file (standards.dat)

```
[aplbd25.0001.fits] 1 3951 100.01 1.250  3420.520  5151.062 bd25 b
 3450.00   1.7339E-13   49.000     2018819.
 3500.00   1.7641E-13   49.000     5606776.
 3571.00   1.7583E-13   49.000     9719213.
 3636.00   1.7435E-13   49.000     1.0545E7
 3704.00   1.7920E-13   49.000     1.0843E7
 3862.00   2.3176E-13   49.000     1.2979E7
 4036.00   2.3268E-13   49.000     1.1930E7
 4167.00   2.3282E-13   49.000     1.2148E7
 4255.00   2.3381E-13   49.000     1.2182E7
 4464.00   2.2868E-13   49.000     9510847.
 4566.00   2.3746E-13   49.000     9659592.
 4785.00   2.3276E-13   49.000     8715135.
 5000.00   2.3374E-13   49.000     6978830.
[aplbd33.0001.fits] 1 3951 240.01 1.004  3420.520  5151.062 bd33 b
 3450.00   5.0007E-13   49.000      934099.
 3500.00   4.9492E-13   49.000     2618254.
 3571.00   4.5824E-13   49.000     4890074.
 3636.00   4.3795E-13   49.000     5456799.
 3704.00   4.3384E-13   49.000     5970267.
 3862.00   4.9779E-13   49.000     8458620.
 4036.00   4.3930E-13   49.000     8664230.
 4167.00   3.9357E-13   49.000     8336657.
 4255.00   3.7057E-13   49.000     8023688.
 4464.00   3.2153E-13   49.000     7031432.
 4566.00   2.9895E-13   49.000     6584059.
 4785.00   2.5996E-13   49.000     5967113.
 5000.00   2.1713E-13   49.000     4949685.
~
```

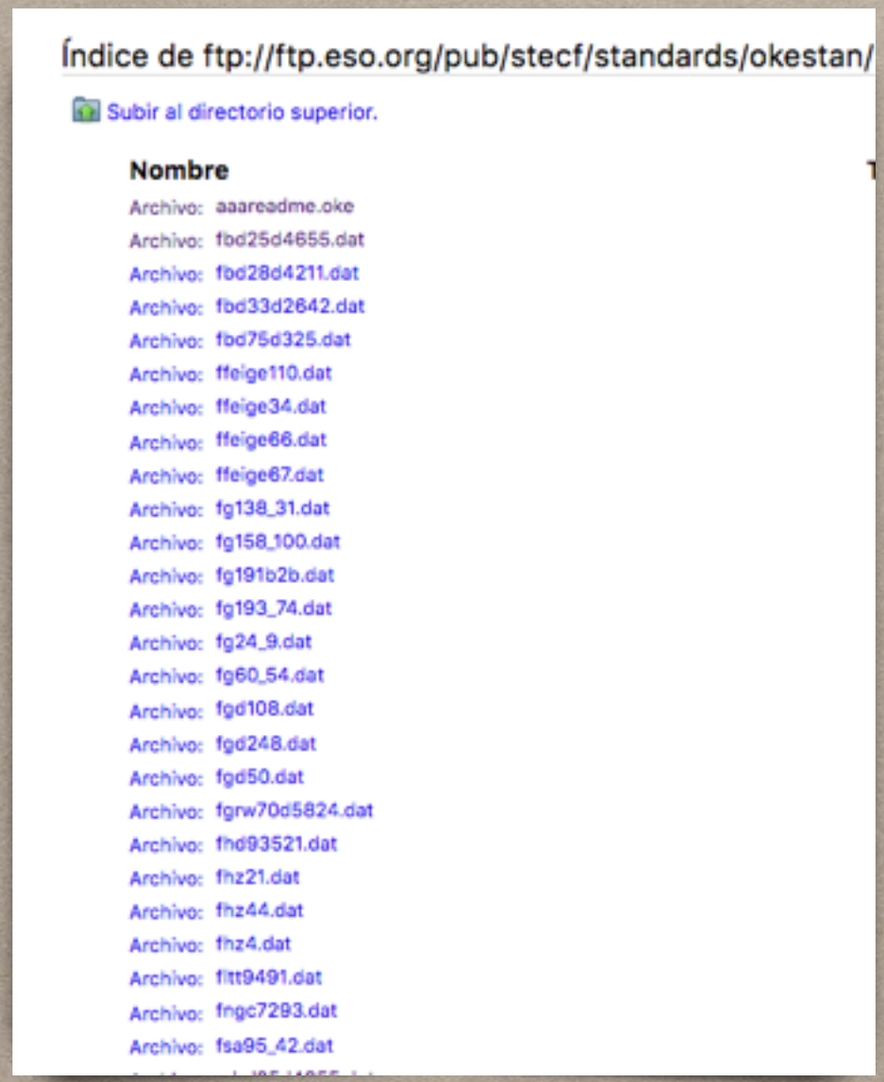
Extract standard start spectrum

apall

ftp://ftp.eso.org/pub/stecf/standards/okestan/

http://www.eso.org/sci/observing/tools/standards/spectra/okestandards.html

http://www.stsci.edu/hst/observatory/cdbs/astronomical_catalogs.html



Índice de ftp://ftp.eso.org/pub/stecf/standards/okestan/

Subir al directorio superior.

Nombre
Archivo: aaareadme.oke
Archivo: fbd25d4655.dat
Archivo: fbd28d4211.dat
Archivo: fbd33d2642.dat
Archivo: fbd75d325.dat
Archivo: ffeige110.dat
Archivo: ffeige34.dat
Archivo: ffeige66.dat
Archivo: ffeige67.dat
Archivo: fg138_31.dat
Archivo: fg158_100.dat
Archivo: fg191b2b.dat
Archivo: fg193_74.dat
Archivo: fg24_9.dat
Archivo: fg60_54.dat
Archivo: fgd108.dat
Archivo: fgd248.dat
Archivo: fgd50.dat
Archivo: fgrw70d5824.dat
Archivo: fhd93521.dat
Archivo: fhz21.dat
Archivo: fhz44.dat
Archivo: fhz4.dat
Archivo: fitt9491.dat
Archivo: fngc7293.dat
Archivo: fsa95_42.dat

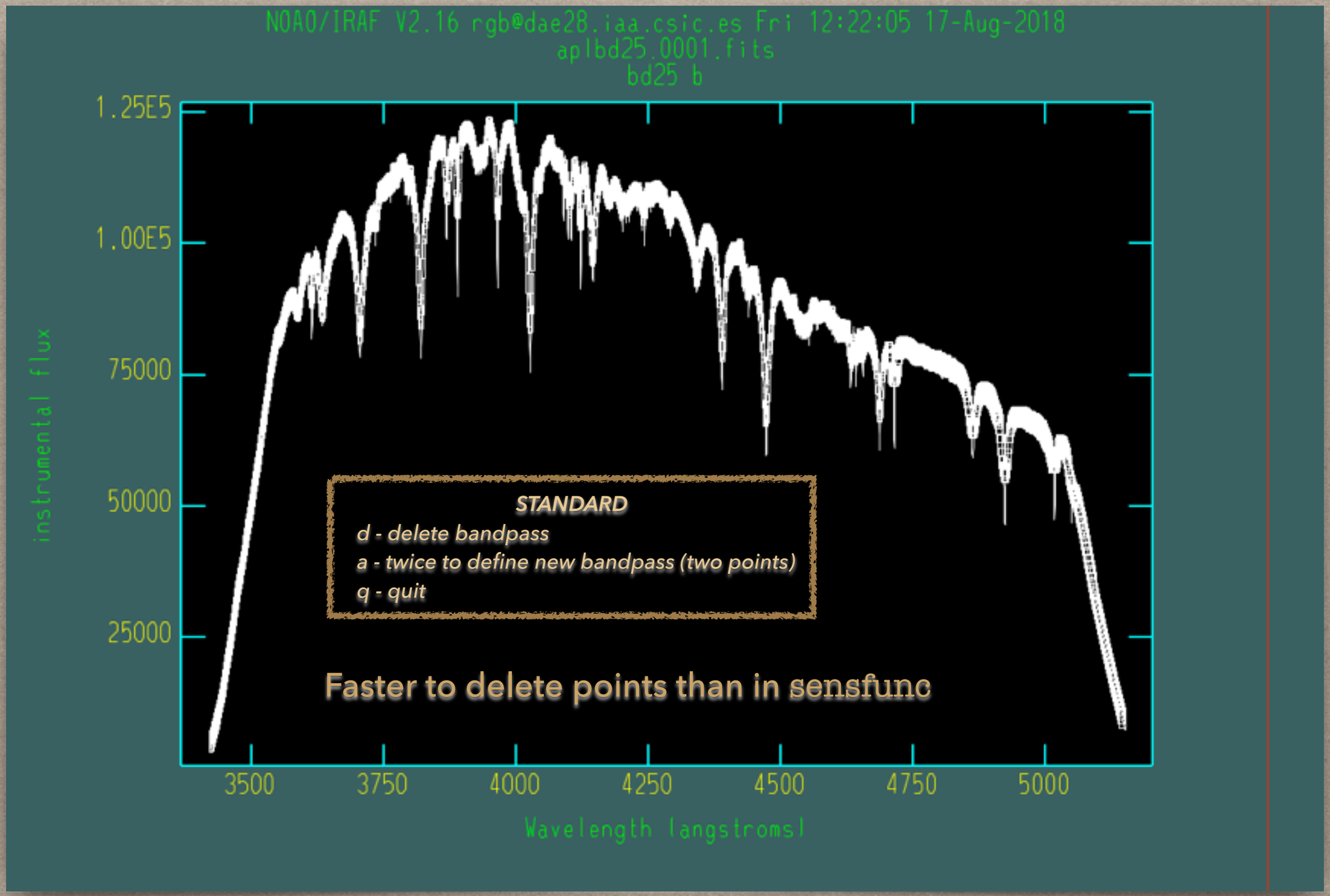
This directory contains data files for all the spectrophotometric standard stars in the Oke 1990 (AJ, 99, 1621) reference, taken from the CALOBS directory at STScI.

*The 'f' files list wavelength (A), flux (ergs/cm/cm/s/A * 10**16) and flux (milli-Jy) and bin (A). The file name consists of a prefix 'f' and the star name.*

The 'm' files list wavelength (A), AB magnitude and bin (A). The file name consists of a prefix 'm' and the star name.

Flux calibration

1) standard



Flux calibration

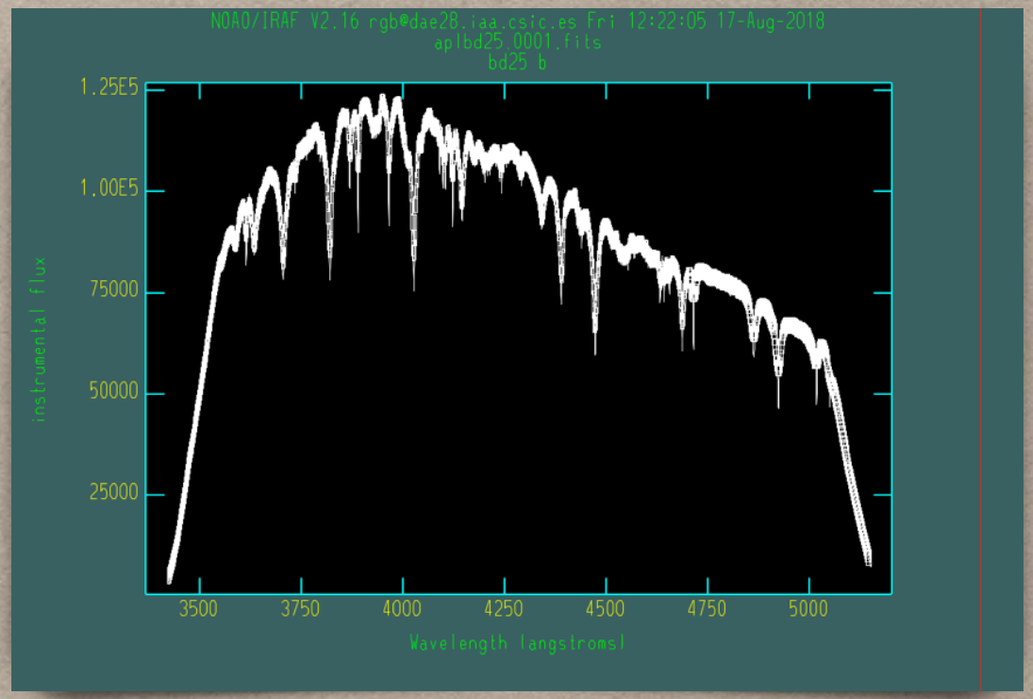
1) standard

External file: mbd25d4655.dat

1) Create file called "standards.men":

Standard stars in .:

mbd25d4655



2) Standard *function write parameter:*

caldir="""

Flux calibration

2) sensfunc

Package = ONEDSPEC
Task = SENSFUNC

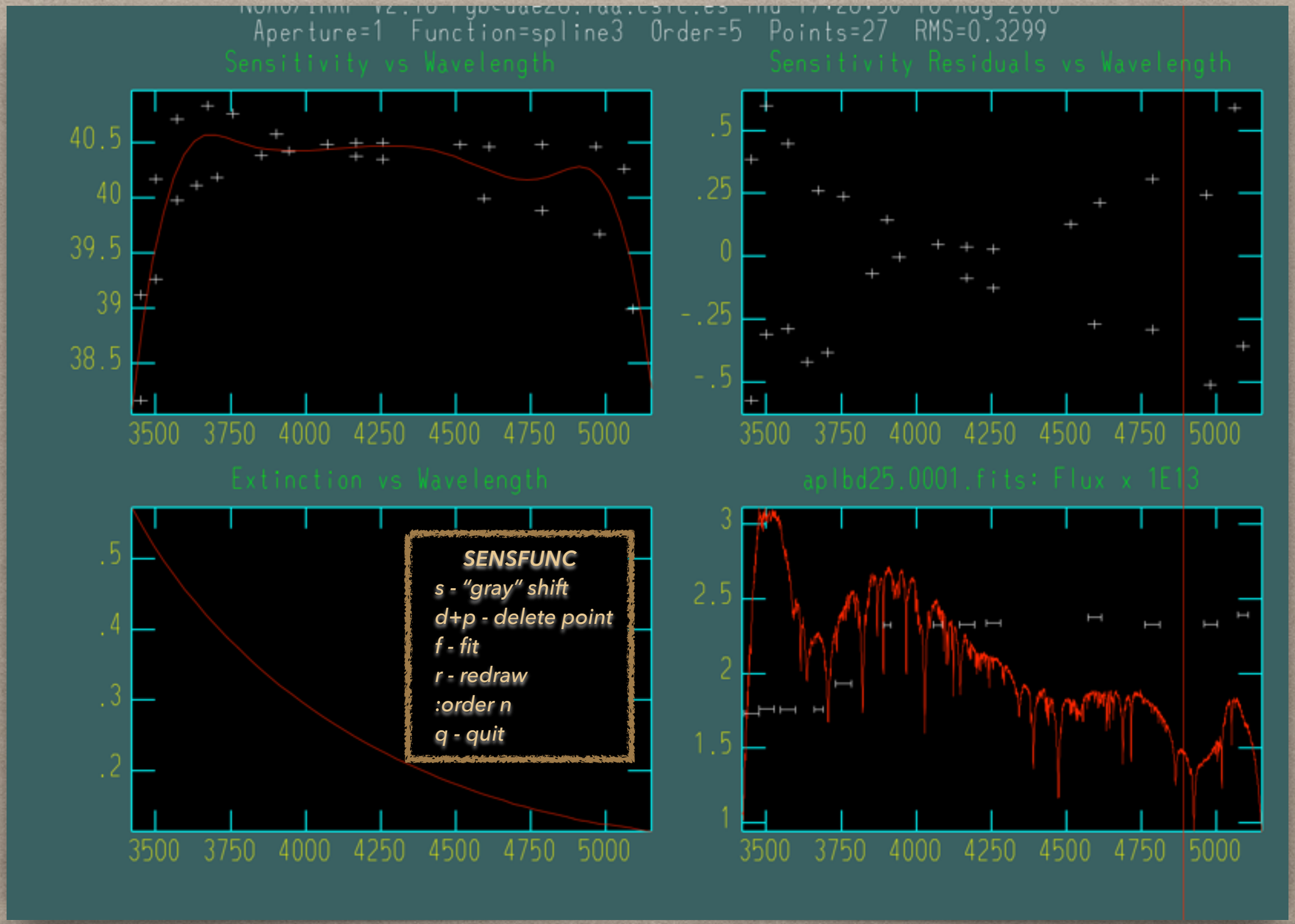
Execute | Save & Quit | Unlearn | Cancel | Sensfunc Help

standards	std	Input standard star data file (from STANDARD)
sensitivity	sens	Output root sensitivity function imagename
(apertures)		Aperture selection list
(ignoreaps)	Yes / No	Ignore apertures and make one sensitivity function?
(logfile)	logfile	Output log for statistics information
(extinction)		Extinction file
(newextinction)	extinct.dat	Output revised extinction file
(observatory)	)_.observatory	Observatory of data
(function)	spline3	Fitting function
(order)	6	Order of fit
(interactive)	Yes / No	Determine sensitivity function interactively?
(graphs)	sr	Graphs per frame
(marks)	plus cross box	Data mark types (marks deleted added)
(colors)	2 1 3 4	Colors (lines marks deleted added)
(cursor)		Graphics cursor input
(device)	stdgraph	Graphics output device
answer	yes	(no|yes|NO|YES)
(mode)	al	

graphs = 'srei'

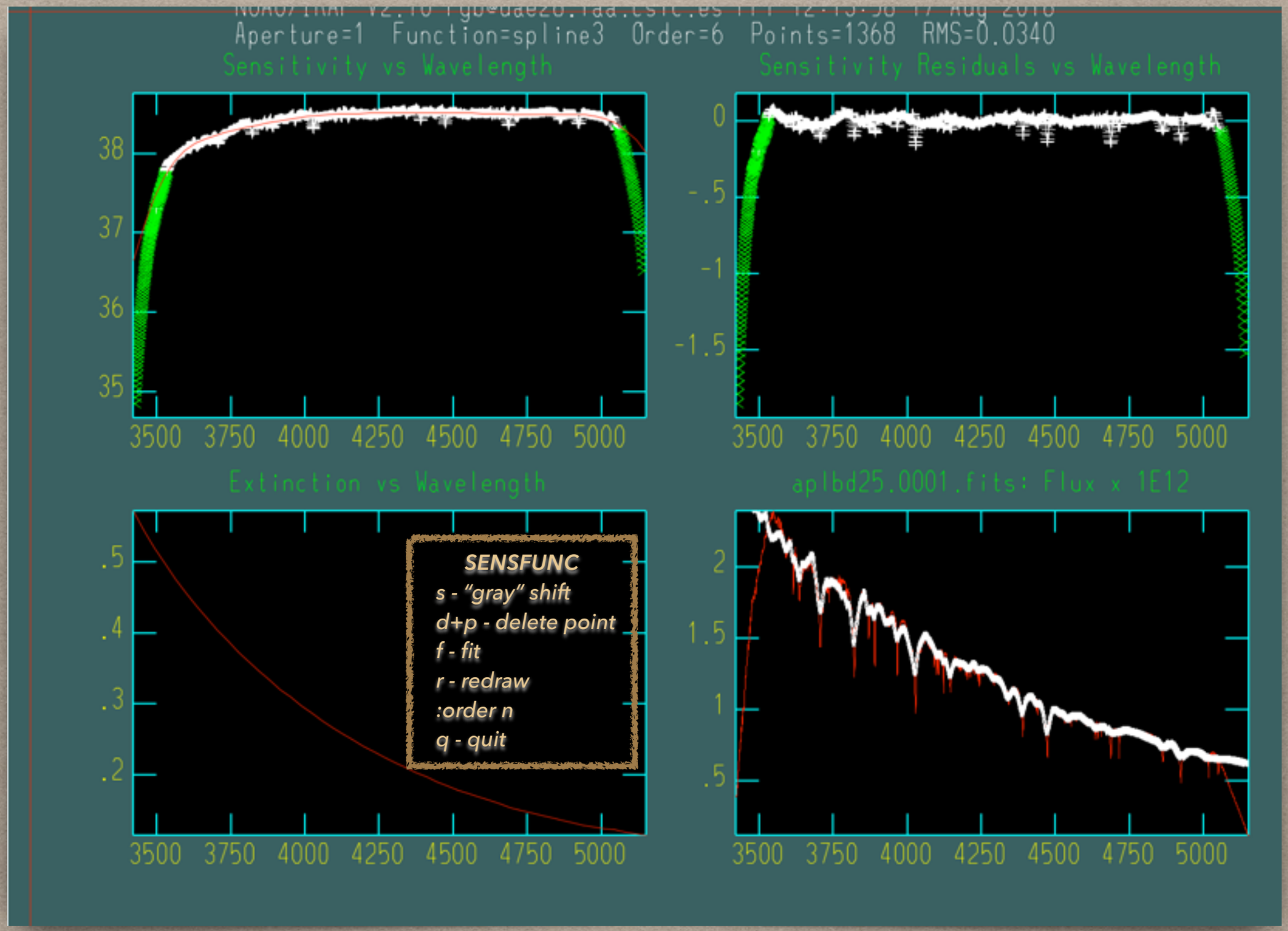
Flux calibration

2) sensfunc



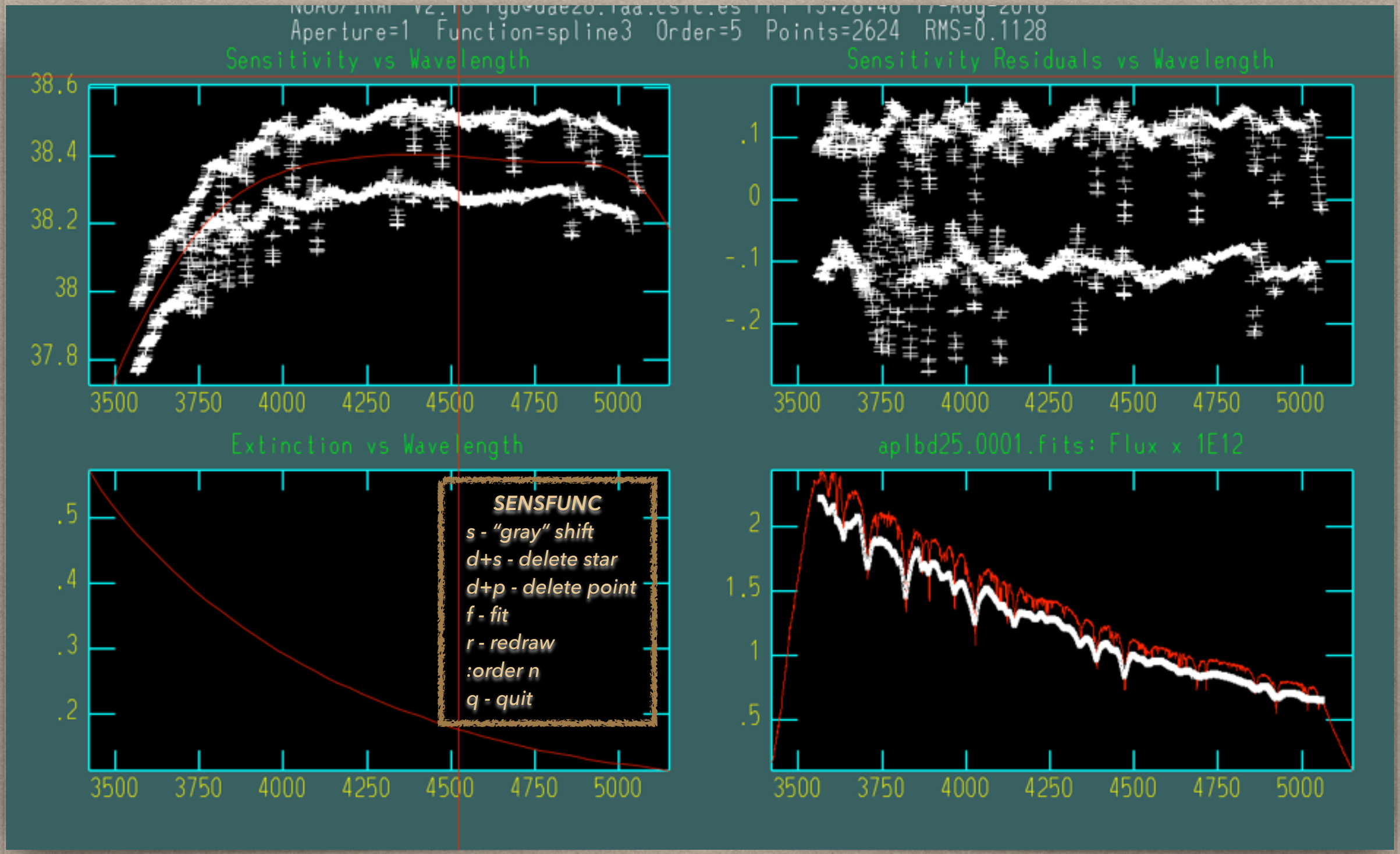
Flux calibration

2) sensfunc



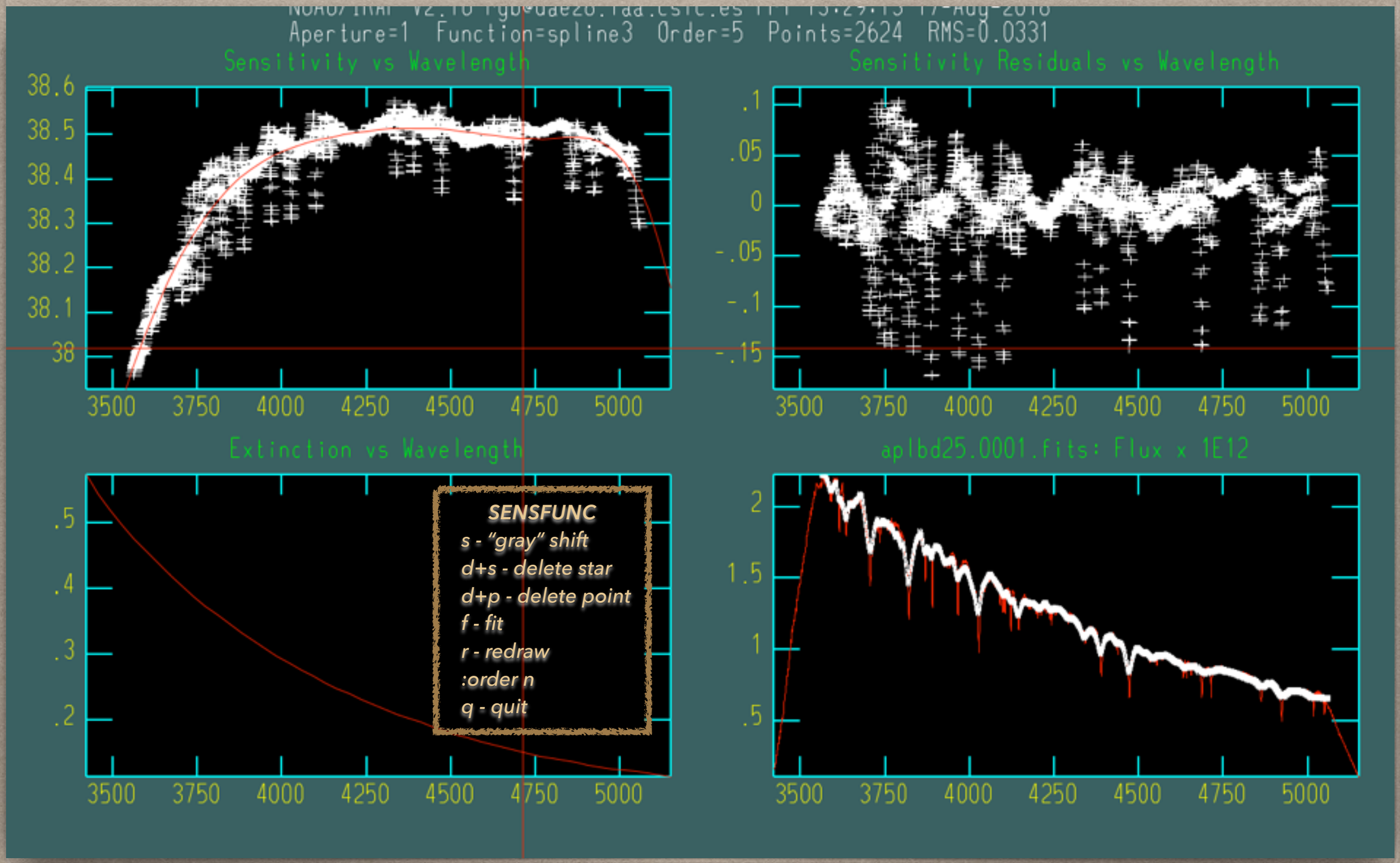
Flux calibration

2) sensfunc



Flux calibration

2) sensfunc



Flux calibration

standard
sensfunc

```
def sensf(lstd, sensitivity='sensfunc', standards='standards.dat', observatory='lapalma', graphs='srei',
        caldir="onedstds$iidscal/", extinct="lapalma_extinction.dat", istand='yes', isens='yes',
        order=5, logfile="", star_names=None, **kwargs):

    sensfile = '%s.fits' % sensitivity
    delfiles([sensfile, standards])

    unlearn('standard')
    unlearn('sensfunc')

    lstd = cklist(lstd)
    star_names = cklist(star_names)

    if star_names is None:
        star_names = ['.'.join(os.path.basename(std).split('.')[0:-1]) for std in lstd]

    for std, star_name in zip(lstd, star_names):
        standard(input=std, output=standards, observatory=observatory, caldir=caldir,
            extinct=extinct, star_name=star_name, interact=istand)

    # ignorepars='yes' --> All the observations are combined into a single sensitivity function
    sensfunc(standards=standards, sensitivity=sensitivity, extinct=extinct, observatory=observatory,
        logfile=logfile, graphs=graphs, interactive=isens, order=order, ignoreaps='yes', **kwargs)

    unlearn('standard')
    unlearn('sensfunc')
```

Flux calibration

standard sensfunc

```
lstd = [aplbd25.0001.fits', 'aplbd33.0001.fits']

lstar_names = ['mbd25d4655', 'mbd33d2642']

sensf(lstd, star_names=lstar_names, caldir="", observatory=observatory,
        extinction=extinction)
```

Flux calibration

calibrate

```
def calflux(lfits, sensitivity='sensfunc.fits', extinction="lapalma_extinction.dat",
        observatory='lapalma', extinct='yes', outfits=None, fmt_out='cf%s',
        in_files='tmp_in.list', out_files='tmp_out.list'):

    lfits = cklist(lfits)

    if outfits is None:
        outfits = [fmt_file(fmt_out, fits) for fits in lfits]

    wfile(in_files, lfits)
    wfile(out_files, outfits)

    unlearn('calibrate')
    # output=pref+'//@'+tlist
    calibrate(input='@'+in_files, output='@'+out_files, sensiti=sensitivity,
        extinct=extinct, extinction=extinction, observatory=observatory,
        ignoreaps='yes')
    unlearn('calibrate')

    delfiles([in_files, out_files])
```

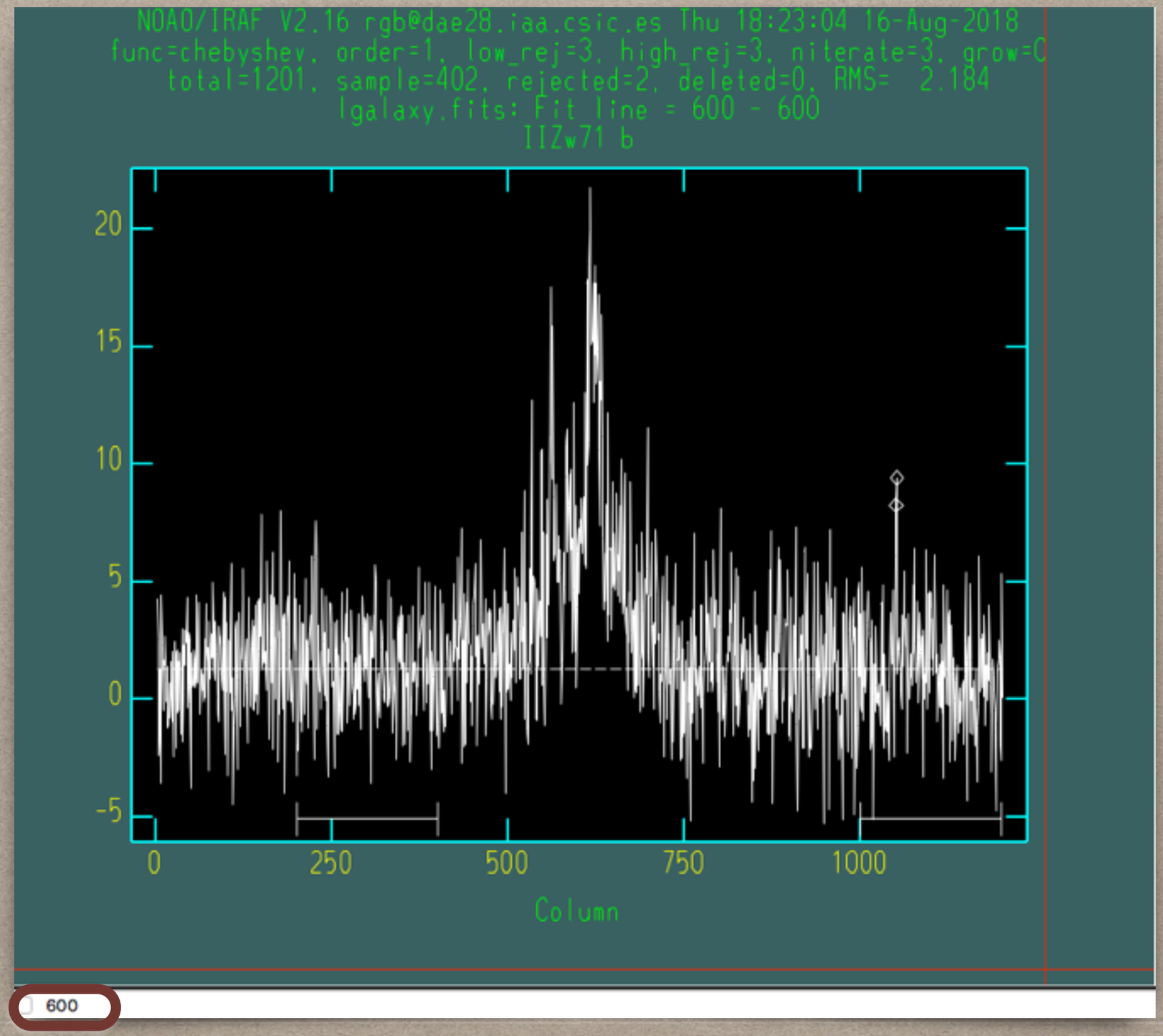
Sky subtraction

background

```
def back(fits, outfits=None, fmt='b%s', **kwargs):
  dback = dict(sample=sample, axis=1, low_reject=3., high_reject=3., inter='yes', order=1, niterat=3)
   dback.update(kwargs)

   if outfits is None:
      outfits = fmt_file(fmt, fits)
   delfiles(outfits)

   background(fits, outfits, **dback)
```

Sky subtraction

background



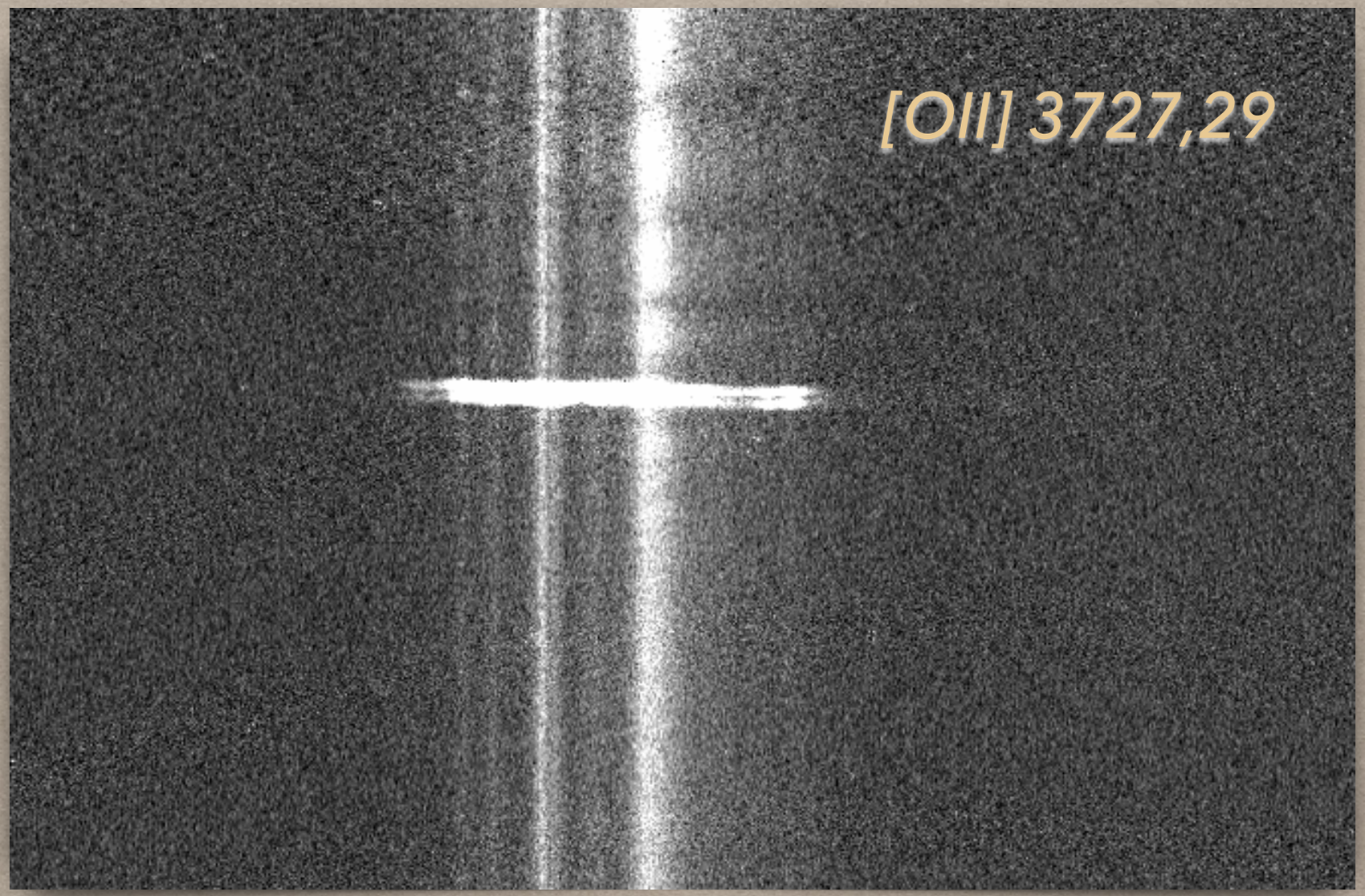
Profiles

splot

λ

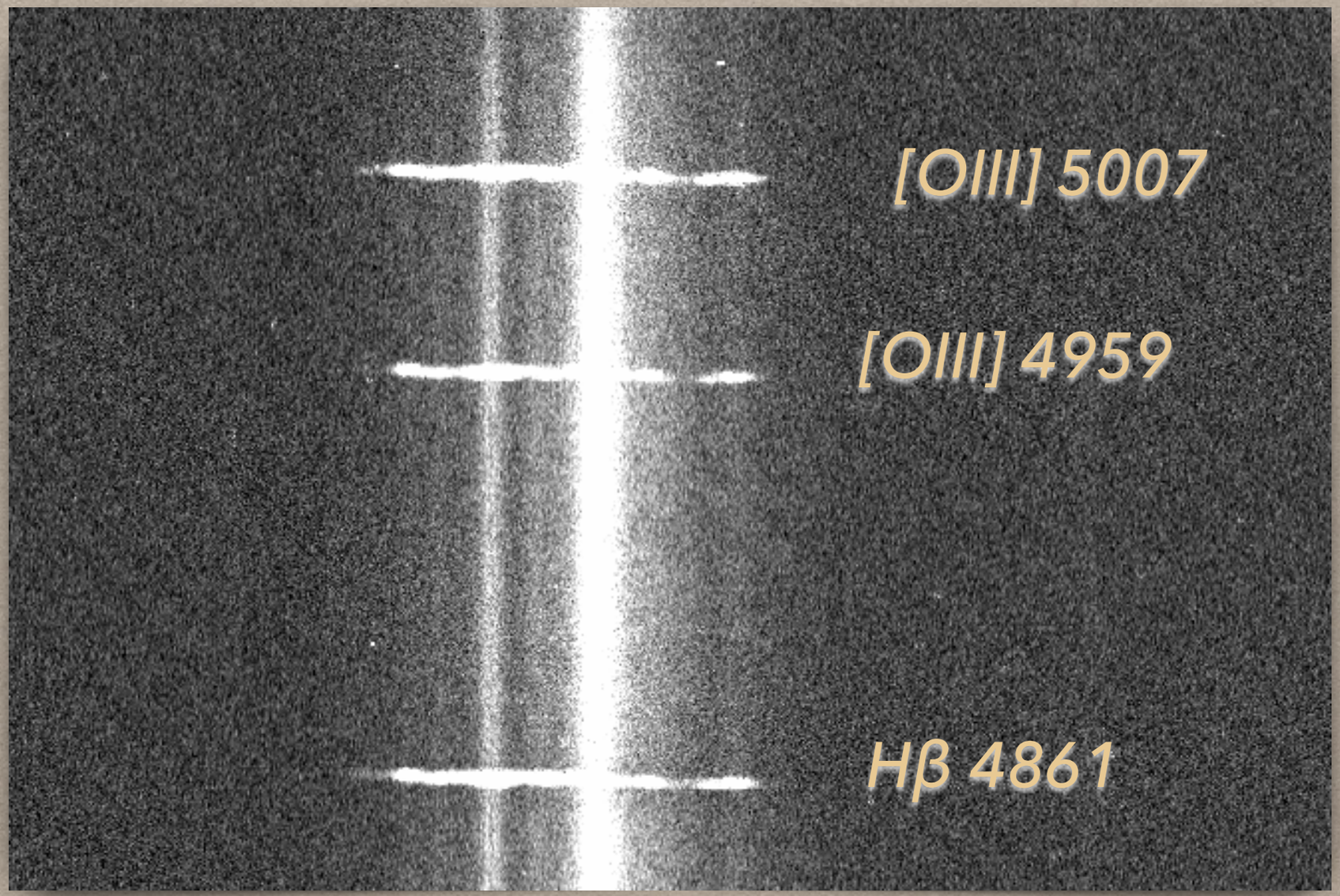
Profiles

splot



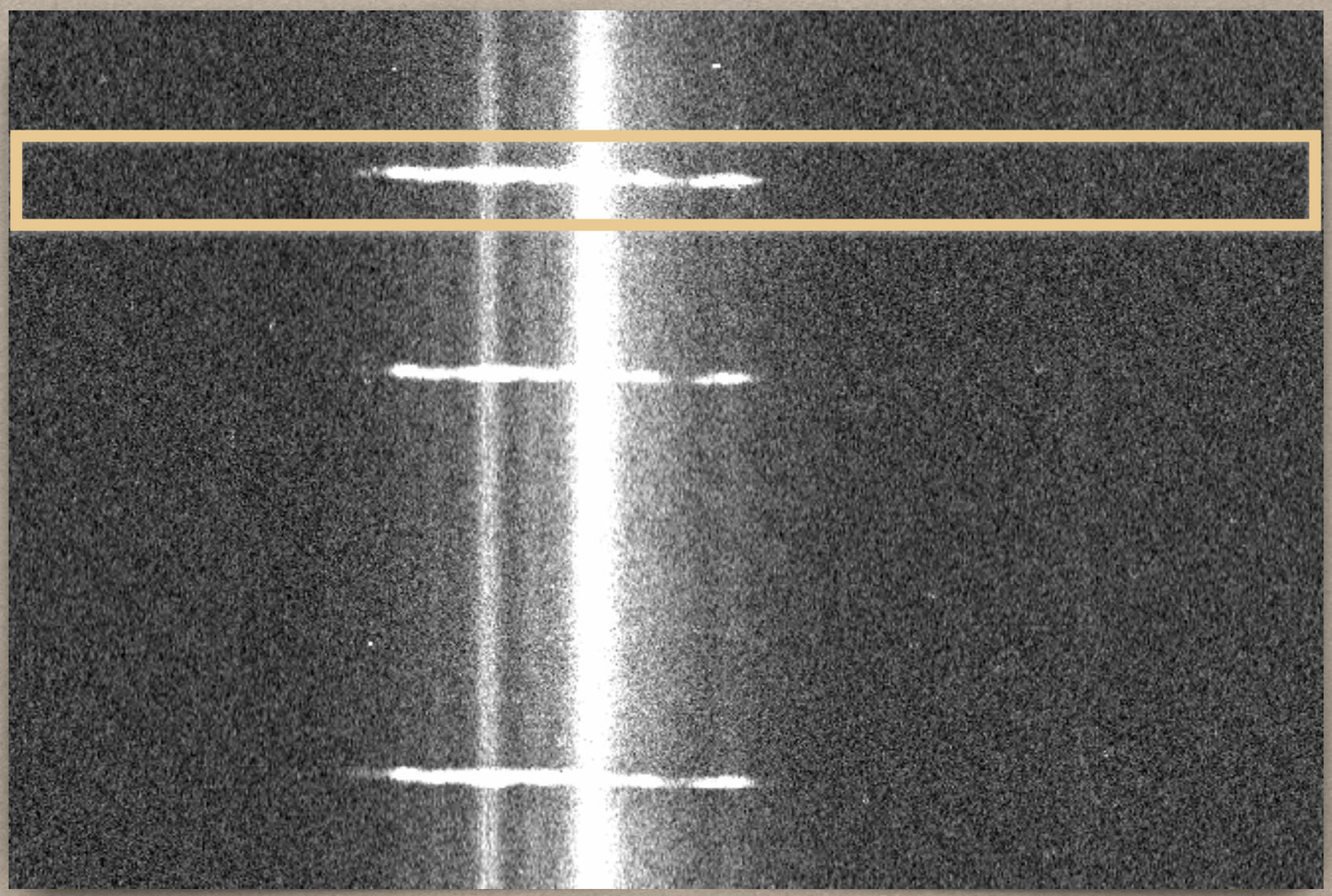
Profiles

splot



Profiles

splot

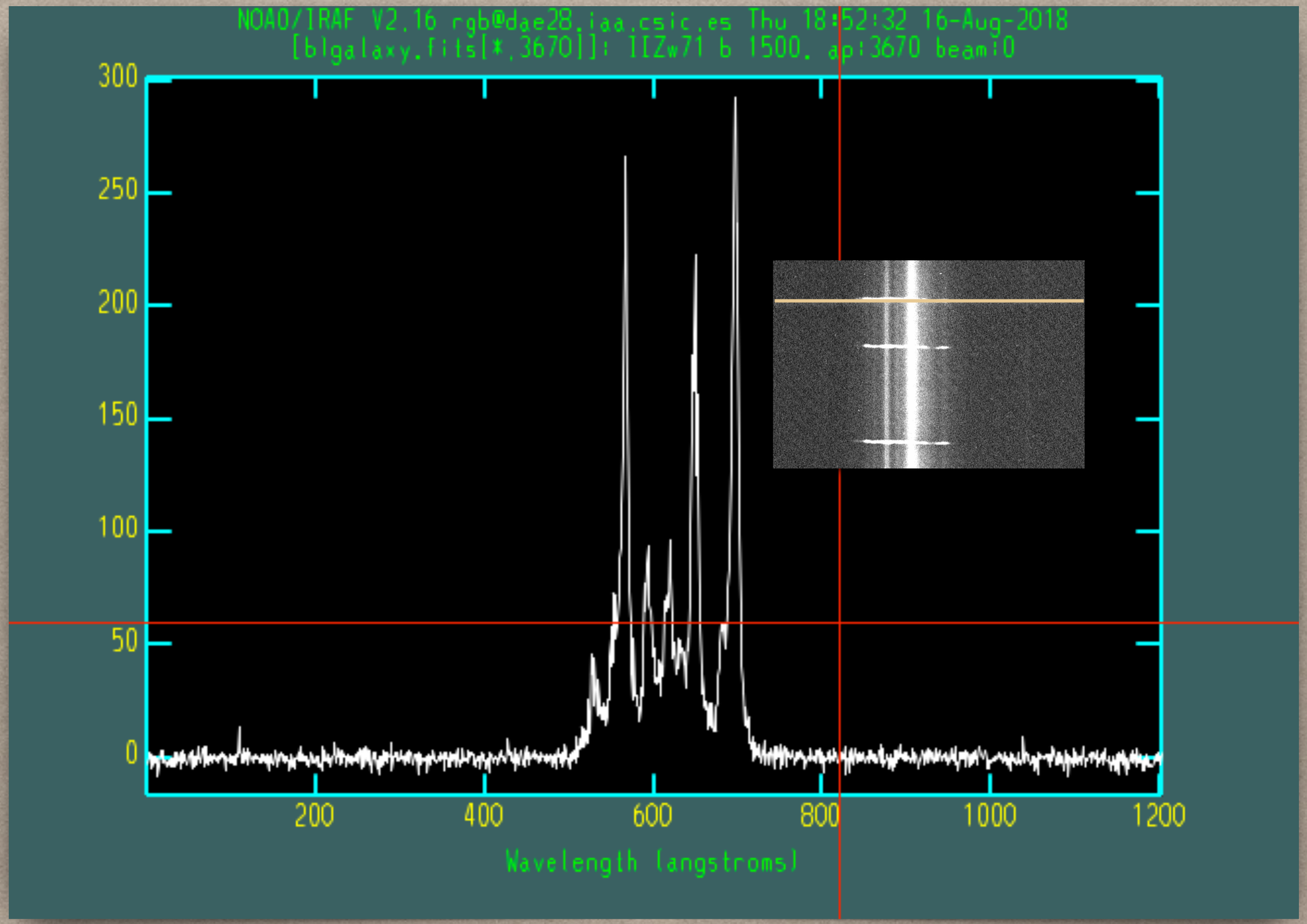


λ

Profiles

splot

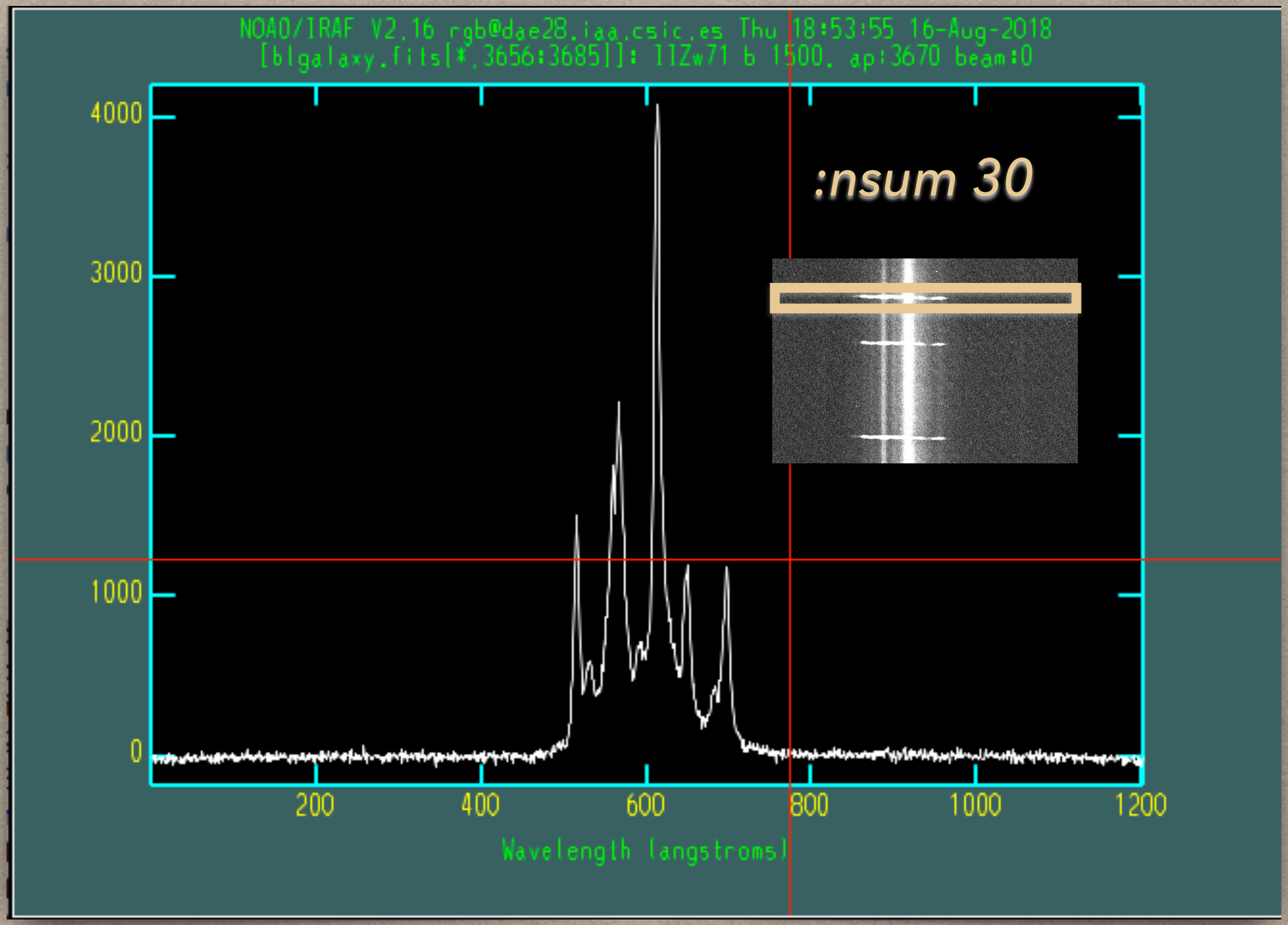
splot blgalaxy.fits[*, 3670]



Profiles

splot

splot blgalaxy.fits[*, 3670]



LONG-SLIT REDUCTION WITH PYRAF

Rubén García-Benito

IAA-CSIC

Optical Spectroscopy School
University of Rwanda
August 24th - September 2nd, 2018