EUROPEAN SOUTHERN OBSERVATORY

Organisation Européene pour des Recherches Astronomiques dans l'Hémisphère Austral
Europäische Organisation für astronomische Forschung in der südlichen Hemisphäre

ESO - European Southern Observatory
Karl-Schwarzschild Str. 2, D-85748 Garching bei München

Very Large Telescope
Paranal Science Operations
FORS data reduction cookbook

Doc. No. VLT-MAN-ESO-13100-4030

Issue 1.1, Date 27/08/2007

Prepared	`K. O'Brien`	Date	Signature
Approved	`G. Marconi`	Date	Signature
Released	`O. Hainaut`	Date	Signature

This page was intentionally left blank

Change Record

Issue/Rev.	Date	Section/Parag. affected	Reason/Initiation/Documents/Remarks
0.1	12/08/07	All	Draft version
1.1	27/08/07		First version

This page was intentionally left blank

Contents

1 Introduction **1**
1.1 Purpose 1
1.2 Reference documents and papers 1
1.3 Abbreviations and acronyms 1
1.4 Stylistic conventions 1

2 Imaging **3**
2.1 Introduction 3
2.2 Flat field 3
2.2.1 Simple methods 5
2.2.2 Advanced Methods 6
2.2.3 Estimate the quality of the flatfied 10
2.3 Sky subtraction and normalization 11
2.4 Combination of frames 11
2.4.1 Centering 11
2.4.2 Co-adding 12

3 FORS long-slit data reduction **14**
3.1 Introduction 14
3.2 Processing the calibration data 14
3.2.1 Running the recipe fors_calib 15
3.2.2 Checking the results 16
3.2.3 Troubleshooting 17
3.3 Processing the scientific data 18
3.3.1 Running the recipe fors_science 19
3.3.2 Checking the results 20
3.3.3 Troubleshooting 21

4 FORS MOS/MXU data reduction **23**
4.1 Introduction 23
4.2 Processing the calibration data 23
4.2.1 Running the recipe fors_calib 24
4.2.2 Checking the results 26
4.2.3 Troubleshooting 29
4.3 Processing the scientific data 34
4.3.1 Running the recipe fors_science 34
4.3.2 Checking the results 36
4.3.3 Troubleshooting 38

5 Multi Object Spectroscopy **41**

5.1 Introduction . . . 41
5.2 Basic reduction steps . . . 41
5.2.1 Removing the detector bias . . . 41
5.2.2 Flat fielding . . . 43
5.2.3 Mapping the slit curvature . . . 48
5.2.4 Applying the distortion correction (optional and not recommended) . . 51
5.2.5 Image combining and sky subtraction . . . 52
5.2.6 Object extraction . . . 54
5.3 Wavelength Calibration . . . 55
5.4 Flux Calibration . . . 57

6 HIgh Time resolution (HIT) modes 60
6.1 Introduction . . . 60
6.2 general comments . . . 61
6.3 High time resolution imaging (HIT-I) . . . 61
6.4 High time resolution spectroscopy (HIT-MS) . . . 61
6.4.1 Preparation . . . 61
6.4.2 Tracing the spectra . . . 63
6.4.3 Extracting the target and sky spectra . . . 65
6.4.4 fixing the header items . . . 69
6.4.5 extracting the arcs . . . 70
6.4.6 extracting the flux standard . . . 70

A Reducing FORS data using the pipeline recipes 71
A.1 Overview . . . 71
A.2 Launching pipeline recipes: Gasgano and Esorex . . . 71

1 Introduction

FORS is the visual and near-UV FOcal Reducer and low dispersion Spectrograph for the Very Large Telescope (VLT). Two versions of FORS have been built and are installed on the cassegrain foci of two of the unit telescopes that comprise the VLT. The FORSes are multi-mode instruments that are capable of imaging, imaging polarimetry, spectropolarimetry, long-slit spectroscopy, and multi-object spectroscopy (with moveable slit jaws or laser-cut masks). In addition they have a number of "HIgh Time resolution" (HIT) modes. This document brings together descriptions of how to reduce data taken with many of the different modes.

1.1 Purpose

The purpose of this document is to aid users in reducing data taken in one the many different modes of FORS1&2. Whilst the different chapters follow the data reduction for a different mode in a step-by-step manner, they are written to highlight possible pitfalls specific to FORS data and should not be mis-interpreted as the only way to reduce the data. The authors of the different chapters work in very different fields of astronomy. It is the hope of the authors that the many different methods used highlight the fact that the solutions to many problems are often specific to the application the data is intended for. In addition, many different software packages have been used and examples are given from each. This again highlights the many different approaches to astronomical data analysis.

1.2 Reference documents and papers

1 FORS User Manual - VLT-MAN-ESO-13100-1543
2 FORS Pipeline User Manual - VLT-MAN-ESO-19500-4106
3 Osterbrock, D. et al. 1996, PASP. 108, 277
4 Osterbrock, D. et al. 1997, PASP. 109, 614

1.3 Abbreviations and acronyms

The following abbreviations and acronyms are used in this document:

SciOp	Science Operations
ESO	European Southern Observatory
Dec	Declination
ESO-MIDAS	ESO's Munich Image Data Analysis System
FITS	Flexible Image Transport System
IRAF	Image Reduction and Analysis Facility
PAF	PArameter File
RA	Right Ascension
UT	Unit Telecope
VLT	Very Large Telescope

1.4 Stylistic conventions

The following styles are used:

bold	in the text, for commands, etc., as they have to be typed.
italic	for parts that have to be substituted with real content.
$\boxed{\text{box}}$	for buttons to click on.
`teletype`	for examples and filenames with path in the text.

Bold and *italic* are also used to highlight words.

2 Imaging

Written by Olivier Hainaut, ESO Paranal Science Operations

2.1 Introduction

In this chapter I shall make extensive use of the European Southern Observatory Munich Image Data Analysis System (MIDAS). For those not familiar with MIDAS, see "MIDAS for the Dummies", which can be downloaded from `http://www.sc.eso.org/~ohainaut/midas.html`, for an introduction.

The VLT images come with keywords matching the pixel numbers of the images and the RA-DEC coordinates, the "world coordinate system". The WCS takes into account the scale and the rotation of the instrument, but no further distortion. The accuracy of these WCS is typically 1-3″, i.e. suitable for identifying the field and possibly an object, but they are *NOT* accurate enough for any decent astrometric work.

The first step I recommend is to get rid of the WCS keywords, and to work with the pixel numbers until the final steps, when a real astrometric calibration can be performed. To do so:

```
COPY/DD image STEP image STEP_RATAN
COPY/DD image START image START_RATAN
```

Copy the original WCS keywords for further use

`WRITE/DES image STEP 1.,1.` for FORS1
`WRITE/DES image STEP 2.,2.` for FORS2 in standard bin 2x2
`WRITE/DES image START 1.,1.`
`WRITE/DES image CUNIT "PIXEL PIXEL"`

In MIDAS, any operation combining images is done in such way that pixels with the same coordinates are worked on. If one works with images which have the same `START`, the images will be combined pixel by pixel. If the `START` of one of the images is different, this is equivalent to shifting the second image with respect to the first one. If an image is cropped (eg with `EXTRAIMAGE`), the `START` keywords of the resulting image are adjusted so that its pixel keep the same coordinates as in the original image.

2.2 Flat field

The starting point for this section is a series of de-biased frames.

The purpose of the flat-field is to remove the sensitivity variations across thee detector. As these variations act as a noise source, the precision of the flatfield correction will have direct consequences on the photometric precision that can be acheived, and on limiting magnitude of the observations. Those are caused by (*i*) intrinsic sensitivity variations of the pixels (either caused by the substrate, or the coating), (*ii*) extrinsic variations, eg a dust grain or hair sitting on the chip, (*iii*) optical design, and (*iv*) dust or external bodies on the optics. The first 2 causes are extremely stable over time, with a time scale of years for (*i*) and (*ii*). The optical effect can vary with a time-scale of hours, eg because of flexures; fortunately, in FORS, these effects are both extremely small and stable. As FORSes are fairly well sealed and not often opened, the dust on the optics is reasonablely stable, with a timescale of weeks.

Various types of frames are suitable to generate a master Flat field, depending on the needs.

- Internal flat-fields: using an internal pupil screen. These are usually disgusting for imaging flat-fields, and are not taken with FORS.
- Dome flat-fields: pointing the telescope at a fairly uniformly illuminated screen. Their pro: unlimited in numbers, therefore in SN. Con: the illumination is usually an halogen lamp, whose color is very different from that of the sky (introducing some weird color effects in broad band filters), and the pupil illumination is only very approximately matching that of real observations, introducing some slopes, gradients and low spatial frequencies variations in the flatfield. They are not used for FORS.
- Twiligh flat-fields: pointing the telescope at a field devoided of bright stars during twilight. Pros: the pupil illumination matches pretty well that of real observations. Cons: the brightness level changes rapidly (this is taken into account by the FORS twilight flatfield templates); their number is rather limited, so the SN acheivable is also limited. Also, stars can become visible toward the dark end of the twilight, and have to be dealt with. These are the best "simple" flatfields, which explains their popularity.
- Night sky flat fields: long night exposures can have a sky background of a few thousands counts. Pros: perfect match of the science frames. Cons: low exposure level; for instruments affected by sky concentration, the night sky flats are very difficult to use (This is NOT the case of the FORSes). Works only on fairly empty fields (hint: load the frame setting the cuts at mean-3sigma, mean+10sigma; if more than 25% of the pixels appear white, either because of the number of objects, or because some very bright objects are in the field, then forget about using this frame as a flatfield). Using these flatfields is sometimes called "super-flatfielding".

For Night Sky flat field to work, it is absolutely critical that the images be taken with offsets applied to the telecsope between each exposure. The size of the offset must be sufficiently large so that the halo of an object does not overlap from frame to frame. So, a safe minimum distance between exposures is 3-4$\times$seeing. Also, it is better that the offests be computed using a random number generator, in order to avoid the various exposures of a series being aligned along the lines and/or the columns, as this would potentially create some systematic effects. Some observers prefer NOT to offset the telescope between exposures, leaving the object always on the same pixel. While there might be some very specific cases in which the benefits of that method are significant, doing so will absolutely prevent a proper flatfielding of the data. Moreover, because of the instrumental flexures and because of the seeing variations, the object of interest will *not* always illuminate the same set of pixels. As a good flat field will not be possible, that method will lead to some uncontrolled systematics that will render any photometric variation suspicious as it will be difficult to discriminate it from a systematic detector effect. In summary, offset.

In the following section, I'll describe the standard, quick textbook method to generate master flatfields. In the next sections, I will deal with advanced flatfielding methods that will create the ultimate master flatfield. The amount of work is much larger, but they make the difference for very deep imaging.

2.2.1 Simple methods

In most cases, this simple method, used on twilight flats, will lead to $\sim 1-3\%$ flatfielding level, which is sufficient for many projects. This method is applicable to all the flats described above. The steps are the following:

- de-bias the raw frames (see previous section)
- mask the stars that can affect the frames
- normalize the de-biased frames
- combine the individual frames into a master flatfield
- divide the debiased science frames by the master flatfield

Normalization In order to combine different individual flatfield frames, they have first to be brought to the same level. While this normalization could be performed to any arbitrary level, normalizing to 1. has the advantage to preserve the natural adu units of the science frames that will be divided. It remains trivial to check for saturation. Some people prefer to normalize to the gain of the detector, so that the final images be in electron units rather than adus.

Normalization area: the region of the chip that is used in order to measure the level of exposure has to be carefuly selected AND has to be kept constant for all frames.

Area: pick a large region close to the center of the chip, completely in the exposed region, and without major detector blemishes (bad column). In the case of a chip multiple read-out such as FORS1, pick the same area in each of the amplifiers.

Suggested areas are listed in the following table:

FORS1: [x0,y0:x1,y1] = [@424,@541:@1624,@1541]

this region is centered on the chip center (so it uses the same surface from the 4 amplifiers), and avoids a big beletic at 1700,700.

FORS2- chip1:	=	[@400,@20:@3700,@1800]
FORS2- chip2:	=	[@400,@700:@3700,@2050]

In order to measure the number of counts in the region, use the median, which will minimize the effect of the stars. In MIDAS, this is done with

```
STATISTIC/IMA image.bdf [@400,@700:@3700,@2050]
```

and the median is returned to the screen and in keyword OUTPUTR(8). I recommend keeping the normalization factor in an image descriptor for further use:

```
WRITE/DESC image norm_ff/r/1/1 outputr(8)
```

The normalization is then obtained simply by dividing the frame by the median. In MIDAS:

```
COMPUTE/IMA n_image = image / outputr(8)
```

Combination into a master flatfield Once all the frames are normalized, create catalogues per type of flat (as defined in the introduction) and per filter. In MIDAS, the simplest is to create a catalogue with all the normalized frames, and then to edit it manually.

The simple combination is done through a median with a threshold, or better, a rejecting mean.

- Median with threshold: a pixel of the result frame is the median of the values of the same pixel in the normalized frames, not taking into account the pixels that are brighter than a given threshold (1.15 is a good values for the FORSes). In MIDAS:

  `AVERAGE/IMAGE ff_R = twilight_R.cat M 0 median 0.5,1.15`

  (M is not relevant, undefined pixels are set to 0, and 0.5,1.15 define the interval of valid pixel values).

- Rejecting mean: the result is the mean of the pixel's values, rejecting the extremes. As a rule of thumb, keep the interval [median-40%:median+30%] (reflecting the fact that you can have bright cosmics and stars, but no black holes). In MIDAS:

  `AVERAGE/IMAGE ff_R = twilight_R.cat M 0 median,5,4 0.5,1.15`

  Assuming the catalogue contains 15 frames, this will take the mean (even though the parameter says median) of the pixels (by intensity) with numbers in [3:11], rejecting pixels with ordered numbers 1,2,12,13,14: median is pixel number 7, and we go from 7-5 to 7+4. This also assumes that all pixels are in the validity interval 0.5,1.15. Those which are not are rejected at the begining.

Final steps It may be worth re-normalizing the final flatfield.

Also, I like to crop the flatfield, removing all the unexposed regions of the frames. In that way, dividing the science frame by the flatfield will actually crop it at the same time. Note that this requires you use a DRS that preserves the physical pixel number in the header.

The final step is then to apply the flatfield to the science frames:

```
COMPUTE/IMAGE f_science = science / ff_R
```

(don't mix up filters...)

2.2.2 Advanced Methods

A word of warning While the previous section was rather simple, with fairly predictible results, what is discussed here is half way between art and black magic. Obtaining a good flatfield requires a lot of work, several attempts adjusting parameters to the actual dataset. The difference between the final result using a good flatfield or a bad flatfield can easily be more than 1mag difference in limiting magnitude, or a photometric accuracy of 0.1 or 0.01 magnitude. If you are observing bright sources (mag brighter than 25) and are satisfied with a precision of 10%, don't lose your time here.

In all this section, it is assumed that the frames are already decently flat to start with. For FORS2, the 2 chips are sufficiently flat. For FORS1, I recommend either starting with frames (including flatfields...) flatfielded with the simple methods, or pipeline processed frames (those with r.FORS*_0000.fits).

Normalization The objects that are visible on the frames are contaminating the sky level estimate, even if one uses the median, and then reject the objects when combining the images.

A first possible improvement consists in making a more robust estimate of the background. The optimal way to deal with the stars is to find them and to mask them before the background estimate and the combination in a master flatfield.

Robust background estimate Let us perform a proper estimation of the sky level by removing all the objects from the frames. SExtractor, a source extraction software included in SciSoft is very good at that.

```
EXTRACT/IMAGE work = image[@400,@20:@3700,@1800]
```
Extract a sub-image defined by the normalization region

```
OUTDISK/FITS work.bdf work.fits
```
Save the extracted frame in a temporary FITS file.

```
sex work.fits -CHECKIMAGE_TYPE BACKGROUND -CHECKIMAGE_NAME check.fits
```
Process it with SExtractor, generating a background image. Note that SExtractor has many parameters. Setting them carefully is critical in order to get good performance of that system for source extractions. For a background determination, the default parameter will do a good job.

```
INDI/FITS check.fits check.bdf
STAT/IMA check
```
Perform the sky evaluation. The median is returned on the screen and in keyword OUTPUTR(8).

Perform then the normalization

```
COMP n_image = image  outputr(8)
```

The main advantage of this method is that it works on the debiased frames. It is ideal for twilight flats. For night sky flat fields, I recommend it as a first iteration.

Masking the stars In order to build a flatfield that is abolutely not contaminated by the stars and galaxies visible on individual night frames, we have to find these objects, and mask them so that the affected pixels are not taken into account in the combination.

One way to mask the stars is to look for any group of pixel that is significany above the background, and to mark these pixels. However, for deep imaging, many objects are too faint to be seen in a single frame, so that the pixels they cover are within the noise. Nevertheless, an object is there, so that object actually contrinutes to increasing the noise at that position. So we have to mask the object. In order to do so, one has to perform a first flatfielding, then re-center and co-add the frames to produce a master field image (see Section 2.4, identify the objects on that deep image, and use it to mask the individual frames. Proceed through this document till you have obtained the master field image, then come back here for a new iteration.

Creation of the mask: Our input is a sky frame (either individual one, or deep master field image).

```
STAT/IMA image.bdf [@400,@20:@3700,@1800]
```

where the region is an example. This returns an estimate of the background value (median, returned in OUTPUTR(8)) and standard deviation (in OUTPUTR(4)). Now we can compute the thresholds for what should be considered an object:

```
max = background + 3* stdDev
```

Whilst we are at it, let's also compute a safe value for pixels that are too low (they will correspond to some bad pixels and blemishes, and also to regions that are vignetted by the guideprobe, as is often the case for FORS' corner) (the careful reader will realize that this step is useful only if you are currently working on individual frames. Indeed, if working on a recombined image, the CCD defects have been averaged out).

```
min = background - 3* stdDev
```

```
FILTER/GAUSS image w_smooth
```

Smear the objects of the input image into a temporary frame. The default parameters work well for FORS.

```
REPLACE/IMA image w_mask w_smooth/max,>=-9999.
```

We replace pixels from the input image, and write the result in the temporary output w_mask. The replaced pixels are those for which w_smooth are between the max threshold and infinity. They are set to an easily identified flag value. Doing so, we will mask the objects and a good fraction of their wings (thanks to the smoothing).

```
REPLACE/IMA w_mask masked_image image/<,min=-9999
```

This time, we replace all the pixels that were below the minimum validity threshold in the original image. Note that we measure on the original image, not the smoothed one: we don't want to remove more than the bad columnts, bad pixels or blemishes.

The final masked_image has the same value as the original one on background pixels, and -9999 where an object or a blemish was found. If the original frame was an individual frame, you are ready to proceed. If you were working on a master field frame, you have now to apply the mask (let's call it masked_field) to all the individual images:

```
REPLACE/IMA image masked_image masked_field<,-9990=-9999
```

All pixels below -9990 in the master field mask are masked in the individual frame. Re-set the original detector pixel coordinates (if you have actually generated a master field frame, this will make sence - if not, it is explained below).

```
WRITE/DES masked_image START 1.,1.
```

Normalize and combine Normalization of the masked frames should not considered the flagged pixels:

```
STAT/IMA masked_image [@400,@20:@3700,@1800] ?  -999,99999
```

(the region is an example) only those whose value is between -999 and 99999 are considered.

Median is in outputr(8).

```
COMP/IMA n_image = masked_image/outputr(8) Normalize
```

Create a catalogue with all the normalized frames, then edit it to isolate the frames fpr a given filter.

```
AVERAGE/IMAGE ff_R = night_R.cat M 0 median,20,20 0.5,1.15
```

As for the simple method, we take the mean of the valid pixels (i.e. between 0.5 and 1.15 - which rejects the flagged pixels) that, once sorted by value, have their number between median-20 and median+20. Note that we are doing deep imaging, so there are many frames. I'd recommend to reject the top and bottom 10 frames on a series of 50.

The result is a pretty good night sky flatfield, that should provide a very uniform correction over the whole field.

Optimal combination of flatfield from different sources Dome flats (not available for FORS) present very strong gradients because the pupil illumination is very different from the one of science frames, but they have a very high SN (we can reach almost any arbitrary value of SN). Twilight flats are much better in terms of gradient. Night flats have the ideal illumination pattern, but they have very low SN. In this section, we will see how to combine the best of each flatfield. The idea is to split each type of available flatfield in frames that contain only a given range of spatial frequencies, then to recombine all these giving different weight to the different spatial frequencies.

Before proceeding in this section, it is critical that the frames don't have any sharp edge anymore, as these affect nastily any frequency-based analysis. For FORS2, chop out the unexposed region of the frames, and for FORS1, perform a first flatfield iteration using a basic method to remove the 4 quadrant effects.

There are various methods to split a frame in a stack of spatial frequencies. The simplest is the good old unsharp masking, which isolate frequencies above and below a threshold:

```
FILTER/GAUS ff_R ff_R_lowFrq 100,100
```

remove all the high frequencies by smoothing

```
COMP ff_R_hiFrg = ff_R - ff_R_lowFrq
```

A more powerful way to proceed is to use the wavelet transform, which can isolate several range of frequencies. In MIDAS, the wavelet transform is available as a context:

```
SET/CONTEXT wavelets
TRANSFORM/WAVE ff_R ff_R_wav
VISU/WAVE ff_R_wav
```

The first command computes the wavelet transform, and the second "visualizes" it by extracting 6 planes of spatial frequencies (corresponding to frequencies of 12^0, 12^1, 12^2, 12^3, 12^4, and lower frequencies). The first one contains all the pixel-to-pixel information, and the last one all the very large-scale gradients. Note that, by construction, the first five frames have a 0 average, and that the last one contains all the flux, i.e has an average = 1. Let's number these frames _00 to _05.

Process in the same way all the master flatfields you have, e.g one twilight flat for each evening and morning, one sky flat for different field. The ultimate master flatfield is then obtained by summing these components with weights:

$$\begin{aligned}
\mathit{Ultimate\_FF} &= \sum_{o=0}^{6} w_{o,\mathrm{DOME}} * \mathrm{Dome}_o \\
&+ w_{o,\mathrm{TWI1}} * \mathrm{Twi1}_o \\
&+ w_{o,\mathrm{TWI2}} * \mathrm{Twi2}_o \\
&+ w_{o,\mathrm{FLD1}} * \mathrm{Fld1}_o \\
&+ w_{o,\mathrm{FLD2}} * \mathrm{Fld2}_o \\
&+ \ldots
\end{aligned} \tag{1}$$

with $\sum w_{o,*} = 1$.

The important part is of course to set the proper weights to the various sources. Some rules apply:

- Boost the weight of the (dome and) twilight flats high-frequencies: there is a lot of flux in these, so get as much SN as you can. On the contrary, because of the low SN of the pixel-to-pixel night sky frames, their weight should not be high.
- For completeness, the low frequencies from the dome flats are totally unreliable, so their weight should be set to 0. Those from the twilight flats are decent, and from the night flats are really good.
- For the intermediate frequencies, maximize the SN.

A typical result would then be:

$$\begin{aligned}
\mathit{Ultimate\_FF} &= 1.0 * \mathit{Twi\_0} \\
&+ 1.0 * \mathit{Twi\_1} \\
&+ 0.8 * \mathit{Twi\_2} + 0.2 * \mathit{Night\_2} \\
&+ 0.5 * \mathit{Twi\_3} + 0.5 * \mathit{Night\_3} \\
&+ 0.2 * \mathit{Twi\_4} + 0.8 * \mathit{Night\_4} \\
&+ 1.0 * \mathit{Night\_5}
\end{aligned} \tag{2}$$

Next recommendation is: experiment, try, try again.

2.2.3 Estimate the quality of the flatfied

The basic test criteria ("If it looks flat, it is flat" and similar) are good enough for most programs, thanks to the eyes power to detect small gradients and faint structures.

Nevertheless, for deep imaging or accurate photometry, it is useful to have a more quantitative estimator, in order to determine if the resulting images are sufficiently flat, or if additional steps have to be taken (if possible).

If the flatfielded images are absolutely perfectly flatfielded, the result is equivalent to those obtained with a detector with an absolutely uniform sensitivity. The flatfield therefore does not contribute anymore to the noise. In deep imaging, the detector read-out noise is completely negligible compared to the sky noise, which is therefore the dominating noise. This constitutes an excellent estimator of the quality of the flatfield.

Measure the statistics of the sky background:

```
STAT/IMAGE image
```

(hint: you still have the masks: use the one corresponding to the image you work on to mask the objects). The mean (moment of order 1) gives the level of the background. Using the gain (stored in the CONAD keyword), converts this to electrons. The sky photon noise is a pure poisonian one, so it must be equal to the square root of the sky value. Compare that value to the standard deviation (moment of order 2); if they match, the sky is flat. The moment of order 3 is also useful, as it indicates the flattening of the distribution compared to a perfect poisonian one. In this case, a flattening corresponds to a broadening caused by an incorrect flatfielding. The moment of order 4 is normally not very useful, as the faint background objects (which can be unvisible on the frame) contribute to making a tail to the bright end of the sky pixel distribution, thereby boosting the moment of order 4.

2.3 Sky subtraction and normalization

You now have flatfielded all the frames. I personally like to time normalize the images at this stage, dividing the level by the exposure time, which is stored in `O_TIME(7)`, and to subtract the sky. For the sky subtraction, I use SExtractor, i.e. setting `CHECKIMAGE_TYPE` to `-BACKGROUND`.

2.4 Combination of frames

2.4.1 Centering

In order to combine a series of flatfielded images, one has to register them so that the objects are properly aligned. In order to do this you should select a series of objects, ideally stars, that are well exposed (not too faint, not saturated), and visible in all the frames. Then measure the centroid of all these stars in a reference frame (which would typically be the one with the best seeing, and/or the one with the lower airmass), and keep the values in a table:

```
LOAD/IMA ref_image
CENTER/GAUSS ? ref_xy.tbl
```

The same stars have to be measured in all the other frames. This can be done either clicking them all, or just measuring the first one, and using it to compute a guess offset, which is then used on `ref_xy` to search automatically for the stars in the new frame. The offsets are then computed, and applied to the `START` of the new frame.

```
COMPUTE/TABLE ref_xy :xcen_ref = :xcen
COMPUTE/TABLE ref_xy :ycen_ref = :ycen

LOAD/IMAGE image_1
CENTER/GAUS ? xy_1
```

Measure the first reference star in the new frame:

$$\begin{aligned} dx &= xy\_1.tbl, : xcen, @1 - ref\_xy.tbl, : xcen, @1 \\ dy &= xy\_1.tbl, : ycen, @1 - ref\_xy.tbl, : ycen, @1 \end{aligned} \tag{3}$$

compute the offsets

```
COMPUTE/TABLE ref_xy :xcen = :xcen_ref + dx
COMPUTE/TABLE ref_xy :ycen = :ycen_ref + dy
```

compute the guessed position for each reference star

```
COMPUTE/TAB ref_xy :xstart = :xcen - 20.
COMPUTE/TAB ref_xy :ystart = :ycen - 20.
COMPUTE/TAB ref_xy :xend = :xcen + 20.
COMPUTE/TAB ref_xy :yend = :ycen + 20.
```

Define a search box around the stars

```
CENTERGAUSS image_1.bdf,ref_xy.tbl ref_xy.tbl
```

measures the real xcen ycen of the reference star.

```
COMP/TAB ref_xy :xdiff = :xcen_ref - :xcen
COMP/TAB ref_xy :ydiff = :ycen_ref - :ycen
```

compute the offset for each start

```
SELE/TAB ref_xy :xerr.le.1.  .and.  :yerr.le.1.  .and.  :status.le.0.1
```

keep only the stars with a good measurement

```
STAT/TAB ref_xy :xdiff
STAT/TAB ref_xy :ydiff
```

Compute the mean offsets dx, dy. The star of the new frame must be set to

```
START(1) = ref_image.bdf,START(1) + dx
START(2) = ref_image.bdf,START(2) + dy
```

At that stage, I recommend to reload the image with its new `START`, and to check for a couple of stars that they actually have the same coordinates as in the reference frame. Do so for each frame.

Note that in order to blindly recenter the frames on a moving target, a crude but efficient way to work is to add the motion at this stage:

$$\begin{aligned} START(1) &= ref\_image.bdf, START(1) + dx \\ &+ (image\_1.bdf, O\_TIME(5) - ref\_image.bdf, O\_TIME(5)) * dxdt \end{aligned} \tag{4}$$

where *dxdt* is the motion of the object in x, in pixels per hour. Note that this should be used only if the time span covered by the observations is short, otherwise the apparent motion of the object cannot be approximated by a linear motion.

2.4.2 Co-adding

The safest way to recombine the frames is simply to co-add them (actually, take an average).

```
AVERAGE/IMAGE result = image.cat M 0 average
```

The option `M` ensures that the result will be the union of all the frames, not the intersection.

If one has many frames, the average can be replaced by the median, with the advantage that the cosmic rays and the moving objects (asteroids and such) will be removed from the combined frame. Note, however, that the median is *not* a linear process, and that this can affect the photometry of the combined frame. This should be done only if the number of frames is large, and the median is then becoming equal to the mean of the valid values (ie. not those affected by a cosmic), leading to a better result than the average.

```
AVERAGE/IMAGE result = image.cat M 0 median
```

3 FORS long-slit data reduction

Written by Carlo Izzo, ESO Software Development Division

3.1 Introduction

Long-slit observations are those which are performed either in LSS mode, or in MOS mode with the same offset applied to all slitlets. Observations in MOS mode with aligned slitlets will be indicated in the following as LSS-like observations: however the suffix to the data files classification tags will remain _MOS also in this case.

The algorithms applied for long-slit data processing are somewhat different from those applied in the case of a generic MOS/MXU observation: for instance, more robust methods can be used for the wavelength calibration, thanks to the greater homogeneity of the signal to process. On the other hand some of the procedures applicable to multi-slit spectroscopy cannot be used on LSS and LSS-like observations: for instance, because the slit ends are so far apart, and even not included in the detector, they cannot be used to determine a reliable spectral curvature solution: for this reason the spectral curvature related products cannot be created. In a future pipeline release it will be possible to import a curvature solution obtained from appropriate calibration masks.

The processing of both FORS1 and FORS2 LSS and LSS-like data is identical. In the following, a typical FORS1 LSS data reduction procedure is described: running the recipes, checking the results, and troubleshooting.

3.2 Processing the calibration data

In order to process the calibration exposures associated to a scientific observation, the recipe *fors_calib* must be used. This should be done before attempting to reduce any associated scientific exposure. In this example it is assumed that the following calibration data are available:

- Three flat field exposures:

```
FORS1.2007-09-27T18:59:03.641.fits  SCREEN_FLAT_LSS
FORS1.2007-09-27T19:00:07.828.fits  SCREEN_FLAT_LSS
FORS1.2007-09-27T19:01:14.252.fits  SCREEN_FLAT_LSS
```

- One arc lamp exposure:

```
FORS1.2007-09-27T19:13:03.631.fits  LAMP_LSS
```

- Five bias exposures:

```
FORS1.2007-09-27T08:00:27.821.fits  BIAS
FORS1.2007-09-27T08:01:05.604.fits  BIAS
FORS1.2007-09-27T08:01:44.091.fits  BIAS
FORS1.2007-09-27T08:02:22.070.fits  BIAS
FORS1.2007-09-27T08:03:01.042.fits  BIAS
```

All the listed data are meant to be obtained with the same FORS1 chip, grism, filter, and long slit position that were used for the scientific observation. The same considerations made in the case of MOS/MXU data reduction (Section 4.2, page 23) are valid here.

3.2.1 Running the recipe fors_calib

The input SOF may be defined as follows:

```
FORS1.2007-09-27T08:00:27.821.fits  BIAS
FORS1.2007-09-27T08:01:05.604.fits  BIAS
FORS1.2007-09-27T08:01:44.091.fits  BIAS
FORS1.2007-09-27T08:02:22.070.fits  BIAS
FORS1.2007-09-27T08:03:01.042.fits  BIAS
FORS1.2007-09-27T18:59:03.641.fits  SCREEN_FLAT_LSS
FORS1.2007-09-27T19:00:07.828.fits  SCREEN_FLAT_LSS
FORS1.2007-09-27T19:01:14.252.fits  SCREEN_FLAT_LSS
FORS1.2007-09-27T19:13:03.631.fits  LAMP_LSS
FORS1_ACAT_300I_11_OG590_72.fits    MASTER_LINECAT
FORS1_B_GRS_300I_11_OG590_72.fits   GRISM_TABLE
```

(the _B_ in the GRISM_TABLE name indicates that the table is valid for FORS1 data obtained after the blue CCD upgrade).

Please read also the MOS/MXU corresponding Section on page 24: the considerations made there about the entries in the input SOF are identically valid here.

When all input data and recipe parameters are settled, the *fors_calib* recipe can be launched. It is advisable to launch the recipe without modifying the default configuration provided by the grism table, at least the first time. The recipe would be run again with an opportunely modified configuration only in case the results were not satisfactory (see the Troubleshooting Section 3.2.3, page 17).

Typically *fors_calib* will run in less than half a minute (depending on the platform). Several products are created on disk, mainly for check purposes. The products which are required for the scientific data reduction are the following:

disp_coeff_lss.fits: coefficients of the wavelength calibration fitting polynomials.

master_bias.fits: master bias frame, produced only in case a sequence of raw BIAS exposures was specified in input.

master_norm_flat_lss.fits: normalised flat field image.

slit_location_lss.fits: slit positions on the CCD (at central wavelength).

The products for checking the quality of the result are:

delta_image_lss.fits: deviation from the linear term of the wavelength calibration fitting polynomials.

disp_residuals_lss.fits: residuals for each wavelength calibration fit, produced only if the recipe configuration parameter "check" is set.

disp_residuals_table_lss.fits: table containing different kinds of residuals for a sample of wavelength calibration fits.

master_screen_flat_lss.fits: bias corrected sum of all the input flat field exposures.

reduced_lamp_lss.fits: rectified and wavelength calibrated arc lamp image.

spectra_resolution_lss.fits: mean spectral resolution for each reference arc lamp line.

wavelength_map_lss.fits: map of wavelengths on the CCD.

See the FORS Pipeline User's Manual for a more detailed description of these products.

3.2.2 Checking the results

Things can go wrong. In this Section a number of basic checks are suggested for ensuring that the *fors_calib* recipe worked properly. Troubleshooting is given separately, in the next Section, in order to avoid too many textual repetitions: it often happens, in fact, that different problems have the same solution. Two basic checks are described here: wavelength calibration, and spectral resolution.

Was the spectrum properly calibrated in wavelength?

Check the *reduced_lamp_lss.fits* image first. This image contains the arc lamp spectrum resampled at a constant wavelength step. The spectral lines should all appear perfectly aligned and vertical. Particular attention should be given to lines at the blue and red ends of each spectrum, where the polynomial fit is more sensitive to small variations of the signal.

More detailed checks on the quality of the solution can be made by examining other pipeline products. The image *disp_residuals_lss.fits* contains the residuals of the wavelength solution for each row of each slit spectrum. This image is mostly padded with zeroes, with the only exception of the pixels where a reference line was detected and identified: those pixels report the value of the corresponding residual (in pixel). This image will in general be viewed applying small cuts (typically between -0.2 and 0.2 pixels): systematic trends in the residuals, along both the dispersion and the spatial directions, would appear as sequences of all-positive (white) followed by all-negative (black) residuals, in a wavy fashion, that could also be viewed by simply plotting a profile at different image rows (see Figure 3). Systematic residuals in the wavelength calibration are in general not acceptable, and they may be eliminated by increasing the order of the fitting polynomial.

Another product that can be used for evaluating the quality of the fit is the *disp_residuals_table_lss.fits* table. Here the residuals are reported in a tabulated form for each wavelength in the reference lines catalog, but just for one out of 10 CCD rows (*i.e.*, one out of 10 solutions). In conjunction with the *delta_image_lss.fits* image, plots like the ones in Figure 4 can be produced. See the FORS Pipeline User's Manual for more details on this.

Finally, the table *disp_coeff_lss.fits* might be examined to check how many arc lamp lines were used (column "nlines") and what is the mean uncertainty of the fitted wavelength calibration solution (column "error"), for each CCD row. The model mean uncertainty is given at a 1-σ level, and has a statistical meaning only if the fit residuals do not display any systematic trend and have a random (gaussian) distribution around zero. Typically this uncertainty will be of the order of 0.05 pixels, *i.e.*, much smaller than the root-mean-squared residual of the fit, depending on the number of fitted points (a fit based on a large number of points is more accurate than a fit based on few points). If the parameter "wmode" is set to 2 (the default), the wavelength calibration can be much more accurate than that, even at the extremes of the spectral range. The errors reported in *disp_coeff_lss.fits* always refer to the single calibrations (each CCD row is calibrated independently), but if "wmode" is set to 2 a global model is fitted to all the reference lines visible on the whole CCD, leading to a calibration accuracy of the order of 0.001 pixels (at least theoretically: systematic errors, *e.g.*, due to physical irregularities of the long slit, are not included in this estimate).

Is the spectral resolution as expected?

The table *spectra_resolution_lss.fits* reports on the observed mean spectral resolution. The same considerations made for MOS/MXU are valid here.

3.2.3 Troubleshooting

In this Section a set of possible solutions to almost any problem met with the *fors_calib* recipe is given. It is advisable to try them in the same order as they are listed here. It may be useful to go through this check list even in case the recipe seemed to work well: there might always be room for improvement.

In practice, almost any problem with the pipeline is caused by a failure of the pattern-recognition task. Pattern-recognition is applied to detect the spectra on the CCD, assuming that they all will include an illumination pattern similar to the pattern of wavelengths listed in the reference arc lamp line catalog (see the FORS Pipeline User's Manual for more details on this).

For an immediate visualisation of how successful was the pattern-recognition, just rerun the *fors_calib* recipe setting the "wmode" parameter to 0. This will disable the computation of a global model of the wavelength calibration. The image *reduced_lamp_lss.fits* may look a bit noisier than the same image obtained with "wmode" set to 2, and some of its rows may contain no signal. The *reduced_lamp_lss.fits* image covers an interval of CCD rows beginning with the first and ending with the last row where a local solutions was found. Within this interval, if at any CCD row the line catalog pattern is detected, the spectral signal is wavelength calibrated, resampled at a constant wavelength step, and written to the corresponding row of the *reduced_lamp_lss.fits* image. If a row of this image is empty it is because the pattern-recognition task failed for that row. A few failures (*i.e.*, a few empty rows) are generally acceptable, as they are easily recovered by the global model interpolation. However, a high failure rate is probably the reason why a bad (global) wavelength calibration was possibly obtained.

The reasons why the pattern-recognition task might fail are exactly the same described in the MOS/MXU Section 4.2.3, page 29.

On the other hand, if the pattern-recognition seems to have worked properly, the reason of a *fors_calib* recipe failure can be related to the way the recipe is run:

The wavelength calibration residuals display systematic trends:

Especially if the extracted spectral range is very large, the fitting polynomial may be incapable to replicate the physical relation between pixel and wavelength. In this case, any estimate of the statistical error (such as the fit uncertainties listed in *disp_coeff_lss.fits*) will become meaningless.

Solution: Increase the degree of the fitting polynomial, using the parameter "wdegree".

The calibrated spectra look distorted:

If the global wavelength calibration model is bad, it is either because some of the local solutions are bad, or because there are too few local solutions available for global modeling.

Solution: Rerun the recipe with the parameter "wmode" set to 0, and change opportunely the recipe configuration (especially the parameter "dispersion"), trying to maximise the number of obtained local solutions and to get more uniform results in the *reduced_lamp_lss.fits* image. Finally run the recipe with the new found configuration, but setting the parameter "wmode" back to 2.

The flat field is not properly normalised:

The master flat field is normalised by dividing it by a smoothed version of itself. For various reasons the result may be judged unsatisfactory.

Solution: Change the "sdegree" parameter, indicating the degree of the polynomial fitting the large scale illumination trend along the spatial direction. Alternatively, instead of the default polynomial fitting a median smoothing may be applied: "sdegree" should be set to -1, so that the smoothing box sizes can be specified with the parameters "dradius" and "sradius".

Valid reference lines are rejected:

Sometimes the peak detection algorithm may return inaccurate positions of the detected reference arc lamp lines. Outliers are automatically rejected by the fitting algorithm, but if those lines were properly identified, not rejecting their positions may really improve the overall accuracy of the wavelength calibration.

Solution: Increase the value of the "wreject" parameter. Extreme care should be used here: a tolerant line identification may provide an apparently good fit, but if this is based on misidentified lines the calibration would include unknown systematic errors.

3.3 Processing the scientific data

In order to process scientific exposures the recipe *fors_science* must be used. Currently the scientific exposures can only be reduced one by one, as no combination of jittered exposures is yet provided by the pipeline. In this example it is assumed that the following data are available:

- One scientific exposure:

```
FORS1.2007-09-27T02:39:11.479.fits  SCIENCE_LSS
```

- All the relevant products of the *fors_calib* recipe (see the previous Section 3.2):

```
master_bias.fits              MASTER_BIAS
master_norm_flat_lss.fits     MASTER_NORM_FLAT_LSS
disp_coeff_lss.fits           DISP_COEFF_LSS
slit_location_lss.fits        SLIT_LOCATION_LSS
```

The same recommendations given for the calibration data reduction are valid here.

3.3.1 Running the recipe fors_science

In general the same grism table that was used in the *fors_calib* recipe would be specified in the input set-of-frames, that will look like this:

```
FORS1.2007-09-27T02:39:11.479.fits  SCIENCE_LSS
master_bias.fits                    MASTER_BIAS
master_norm_flat_lss.fits           MASTER_NORM_FLAT_LSS
disp_coeff_lss.fits                 DISP_COEFF_LSS
curv_coeff_lss.fits                 CURV_COEFF_LSS
slit_location_lss.fits              SLIT_LOCATION_LSS
FORS1_B_GRS_300I_11_OG590_72.fits   GRISM_TABLE
```

It is also possible to include a catalog of sky lines: this table should contain a column listing the wavelengths of the reference lines (the name of this column can be specified using the parameter "wcolumn", which is defaulted to "WLEN"), and it should be tagged MASTER_SKYLINECAT. If no sky lines catalog is specified in the SOF, an internal catalog is used instead.

Typically *fors_science* will run in less than 20 seconds (depending on the platform). The products that will be created depend on the chosen configuration parameter setting. Here is a list of all the possible products:

disp_coeff_sci_lss.fits: wavelength calibration polynomials coefficients after alignment of the input solutions to the position of the sky lines. Only created if the parameter "skyalign" is set.

mapped_all_sci_lss.fits: image with rectified and wavelength calibrated spectrum. Always produced.

mapped_sci_lss.fits: image with rectified, wavelength calibrated, and sky subtracted spectrum. Only produced if at least one sky subtraction method is specified.

mapped_sky_sci_lss.fits: image with rectified and wavelength calibrated sky spectrum. Only produced if at least one sky subtraction method is specified.

object_table_sci_lss.fits: position of the spectrum on the CCD, and positions of the detected objects within the spectrum. Only created if at least one sky subtraction method is specified.

reduced_sci_lss.fits: image with extracted objects spectra. Only created if at least one sky subtraction method is specified.

reduced_sky_sci_lss.fits: image with the sky spectrum corresponding to each of the extracted objects spectra. Only created if at least one sky subtraction method is specified.

reduced_error_sci_lss.fits: image with the statistical errors corresponding to the extracted objects spectra. Only created if at least one sky subtraction method is specified.

sky_shifts_long_sci_lss.fits: table containing the observed sky lines offsets that were used for adjusting the input wavelength solutions. Only created if the parameter "skyalign" is set.

wavelength_map_sci_lss.fits: map of wavelengths on the CCD. Only created if the parameter "skyalign" is set.

See the FORS Pipeline User's Manual for a more detailed description of these products.

Currently there is no support for a spectro-photometric calibration. However, a standard star exposure (tag: either STANDARD_MOS or STANDARD_LSS) can be reduced by the *fors_science* recipe as any generic scientific frame.

3.3.2 Checking the results

In this Section a number of basic checks are suggested for ensuring that the recipe *fors_science* worked properly. Troubleshooting is given separately, in the next Section, in order to avoid too many textual repetitions: it often happens, in fact, that different problems have the same solution. Four basic checks are described here: wavelength calibration, sky subtraction, object detection, and object extraction. It is advisable to perform such checks in the given order, because some results make only sense under the assumption that some previous tasks were performed appropriately. For instance, an apparently good sky subtraction does not imply that the spectrum was properly wavelength calibrated.

Were all spectra properly wavelength calibrated?

The wavelength calibration based on calibration lamps, performed at day-time, may not be appropriate for an accurate calibration of the scientific spectra: systematic differences due to instrumental effects, such as flexures, may intervene in the meantime.

To overcome this, the day calibration may be upgraded by testing it against the observed positions of the sky lines in the scientific spectrum. The alignment of the input distortion models to the true sky lines positions is controlled by the parameter "skyalign", that as a default is set to 0 (*i.e.*, the sky lines correction will be a median offset computed for each CCD row, see the FORS Pipeline User's Manual for more details).

In the case of LSS or LSS-like data the sky alignment can be quite accurate, and therefore it would be advisable to always apply it, even in the case of very small corrections. The magnitude and the accuracy of the correction can be examined in the *sky_shifts_long_sci_lss.fits* table. This table is quite different from the *sky_shifts_slit_sci-_mxu.fits* table produced in the case of MOS/MXU data reduction: for its detailed description and usage see the FORS Pipeline User's Manual.

The overall quality of the wavelength calibration (whether a sky line alignment was applied or not) can be examined in the *mapped_all_sci_lss.fits* image. This image contains the scientific spectrum after resampling at a constant wavelength step. The visible sky lines should all appear perfectly aligned and vertical.

A further check on the quality of the solution can be made by examining the *disp_coeff_sc-i_lss.fits* table. This table is only produced in case a sky line alignment was performed.

Column "nlines" reports how many sky lines were used for the distortion model correction, while the "error" column reports the mean uncertainty of the new wavelength calibration solution for each slit spectrum row. The model uncertainty is given at a 1-σ level, and is computed as the quadratic mean of the input model accuracy and the sky line correction accuracy. Typically this uncertainty will be of the order of 0.1 pixel, *i.e.*, much smaller than the root-mean-squared residual of the lamp calibration and of the sky line correction, depending on the number of fitted points. Note however that the errors reported in *disp_coeff_sci_lss.fits* always refer to the single calibrations (each CCD row is calibrated independently): but if the day calibration was based on a global model, and the night alignment would be based on at least 5 sky lines, the final calibration accuracy would always be better than 0.05 pixels.

Is the sky background properly subtracted?

A quick check on sky subtraction can be made by examining the sky subtracted frame, *mapped_sci_lss.fits*. In general the sky subtraction performed by the pipeline for LSS and LSS-like data is not really accurate. The "skylocal" method in this case is equivalent to the "skymedian" method: the sky subtraction is always performed *after* data resampling at constant wavelength step. This problem would be fixed in the next pipeline releases, but in the meantime it is suggested to use the LSS mode sky subtraction just for quick-look, and not for scientific purposes.

Note that the subtracted sky can be viewed in the image *mapped_sky_sci_lss.fits*. The image containing the extracted sky spectra, *reduced_sky_sci_lss.fits*, contains the sky spectra that were extracted applying to the slit sky spectrum exactly the same weights that were used in the object extraction.

Were all objects detected?

See the corresponding MOS/MXU item on page 38 about this. The list of detected objects can be found in the *object_table_sci_lss.fits* table.

Were all the detected objects properly extracted?

See the corresponding MOS/MXU item on page 38 about this, where the *_mxu* suffix should be read *_lss*.

3.3.3 Troubleshooting

In this Section a set of possible solutions to the most common problems with the *fors_science* recipe is given. It is advisable to try them in the same order as they are listed here. It may be useful to go through this check list even in case the recipe seemed to work well: there might always be room for improvement.

The wavelength calibration is bad:

Aligning the wavelength calibration to the position of the observed sky lines may be inaccurate, especially if very few reference lines are used.

Solution: If very few reference sky lines are used, supplying a sky line catalog including more lines (even if weak and/or blended) may help a lot.

Solution: If the wavelength calibration appears to be bad just at the blue and/or red ends of the spectra, go back to the *fors_calib* recipe to obtain a more stable wavelength calibration in those regions.

The sky alignment of the wavelength solution failed:

In case a blue grism is used, or if a spectrum has a large offset toward the red, no sky lines may be visible within the observed spectral range.

Solution: None. It is however possible to modify the columns of coefficients in the input *disp_coeff_lss.fits* table, if the correction can be evaluated in some other way. For instance, the solution can be shifted in wavelength by adding a constant value (in pixel) to column "c0".

Cosmic rays are not removed:

As a default the *fors_science* recipe does not remove cosmic rays hits, leaving them on the sky-subtracted spectrum: if the optimal spectral extraction is applied, most of the cosmics are removed anyway from the extracted spectra. Optimal extraction is however not always applicable, especially in the case of resolved sources.

Solution: Set the "cosmics" parameter to *true*. This will apply a cosmics removal algorithm to the sky subtracted spectrum. The removed cosmic rays hits will be included in the (modeled) sky image, *mapped_sky_sci_lss.fits* (see the FORS Pipeline User's Manual for more details).

The sampling of the remapped scientific spectrum is poor:

When the spectrum is wavelength calibrated, it is remapped undistorted to images such as *mapped_sky_sci_lss.fits* or *mapped_sci_lss.fits*. This remapping may be judged to undersample the signal along the dispersion direction.

Solution: Change the value of the "dispersion" parameter. This parameter doesn't need to be identical to the one used in the *fors_calib* recipe. See however the corresponding MOS/MXU item on page 40 about this.

The extracted spectra are normalised in time:

The default behaviour of this recipe is to normalise the results to the unit exposure time.

Solution: Set the parameter "time_normalise" to *false*.

Some "obvious" objects are not detected:

Examining the *mapped_sci_lss.fits* image it may appear that some clearly visible object spectra are not detected (let alone extracted) by the recipe.

Solution: Setting "cosmics" to *true* (cleaning cosmic rays hits) may help.

Solution: Try different set of values for the parameters "ext_radius" and "cont_radius".

4 FORS MOS/MXU data reduction

Written by Carlo Izzo, ESO Software Development Division

4.1 Introduction

The processing of both MXU and MOS FORS1/2 data is identical: the only difference lies in the suffix, either _MXU or _MOS, assigned to the classification tags of input and output files. There is one exception: if a MOS observation is performed assigning the same offset to all the slitlets, the aligned slitlets are for all purposes equivalent to a single long-slit. This case is handled differently from what is described in this Section. A description of the reduction of LSS and LSS-like observations is given in Section 3, page 14.

In the following, a typical FORS2 MXU data reduction procedure is described: running the recipes, checking the results, and troubleshooting.

4.2 Processing the calibration data

In order to process the calibration exposures associated to a scientific observation, the recipe *fors_calib* must be used. This should be done before attempting to reduce any associated scientific exposure. In this example it is assumed that the following calibration data are available:

- Three flat field exposures:

```
FORS2.2004-09-27T18:59:03.641.fits  SCREEN_FLAT_MXU
FORS2.2004-09-27T19:00:07.828.fits  SCREEN_FLAT_MXU
FORS2.2004-09-27T19:01:14.252.fits  SCREEN_FLAT_MXU
```

- One arc lamp exposure:

```
FORS2.2004-09-27T19:13:03.631.fits  LAMP_MXU
```

- Five bias exposures:

```
FORS2.2004-09-27T08:00:27.821.fits  BIAS
FORS2.2004-09-27T08:01:05.604.fits  BIAS
FORS2.2004-09-27T08:01:44.091.fits  BIAS
FORS2.2004-09-27T08:02:22.070.fits  BIAS
FORS2.2004-09-27T08:03:01.042.fits  BIAS
```

All the listed data are meant to be obtained with the same FORS2 chip, grism, filter, and mask that were used for the scientific observation. Grism and filter are important for the selection of the appropriate static calibration tables (line catalog and recipe configuration tables – see ahead): in this example it is assumed that respectively grism 300I and filter OG590 are in use.

In the following it is also assumed for simplicity that the recipe launcher (either *Gasgano* or *Esorex*) is configured in such a way that the products file names will be identical to their tags, with an extension *.fits*. Moreover, it is assumed that all the handled files (both inputs and products) are located in the current directory (if this were not the case, the appropriate directory path would also be added to the names of the input files). To have *Esorex* behave in this way, just set

```
esorex.caller.suppress-prefix=TRUE
esorex.caller.output-dir=.
```

in the *$HOME/.esorex/esorex.rc* file. In general this is not an issue when using *Gasgano*, because this application keeps track of the pipeline products automatically, applying appropriate tagging and maintaining the association of the products with the raw frames they were derived from.

4.2.1 Running the recipe fors_calib

The input SOF may be defined as follows:

```
FORS2.2004-09-27T08:00:27.821.fits  BIAS
FORS2.2004-09-27T08:01:05.604.fits  BIAS
FORS2.2004-09-27T08:01:44.091.fits  BIAS
FORS2.2004-09-27T08:02:22.070.fits  BIAS
FORS2.2004-09-27T08:03:01.042.fits  BIAS
FORS2.2004-09-27T18:59:03.641.fits  SCREEN_FLAT_MXU
FORS2.2004-09-27T19:00:07.828.fits  SCREEN_FLAT_MXU
FORS2.2004-09-27T19:01:14.252.fits  SCREEN_FLAT_MXU
FORS2.2004-09-27T19:13:03.631.fits  LAMP_MXU
FORS2_ACAT_300I_21_OG590_32.fits    MASTER_LINECAT
FORS2_GRS_300I_21_OG590_32.fits     GRISM_TABLE
```

The input BIAS frames are used to generate a median bias frame that is internally subtracted from all the input raw images, and eventually written to disk for further use. A master bias frame may also be produced using other means (taking care of trimming the overscan regions from the final result). This user-produced master bias frame may be specified instead of the sequence of raw BIAS frames, using the tag MASTER_BIAS:

```
master_bias.fits                    MASTER_BIAS
FORS2.2004-09-27T18:59:03.641.fits  SCREEN_FLAT_MXU
FORS2.2004-09-27T19:00:07.828.fits  SCREEN_FLAT_MXU
FORS2.2004-09-27T19:01:14.252.fits  SCREEN_FLAT_MXU
FORS2.2004-09-27T19:13:03.631.fits  LAMP_MXU
FORS2_ACAT_300I_21_OG590_32.fits    MASTER_LINECAT
FORS2_GRS_300I_21_OG590_32.fits     GRISM_TABLE
```

The MASTER_LINECAT and the GRISM_TABLE are static calibration tables that are available in the calibration directories delivered with the pipeline recipes.

The file *FORS2_ACAT_300I_21_OG590_32.fits* is the default catalog of reference arc lamp lines for grism 300I and filter OG590 of the FORS2 instrument. This catalog may be replaced by alternative ones provided by the user, if found appropriate.

The *FORS2_GRS_300I_21_OG590_32.fits* table contains the default *fors_calib* recipe configuration parameters for grism 300I and filter OG590 of the FORS2 instrument. If this file is not specified, appropriate values for the parameters must all be provided by the user, either on the command line if using *Esorex*, or in the Parameters panel if using *Gasgano*. Note that if a GRISM_TABLE is specified, the parameter values explicitly provided by the user will always have priority on those listed in the grism table.

When all input data and recipe parameters are settled, the *fors_calib* recipe can be launched. It is advisable to launch the recipe without modifying the default configuration provided by the grism table, at least the first time. The recipe would be run again with an opportunely modified configuration only in case the results were not satisfactory (see the Troubleshooting Section 4.2.3, page 29).

Typically *fors_calib* will run in less than half a minute (depending on the platform). Several products are created on disk, mainly for check purposes. The products which are required for the scientific data reduction are the following:

curv_coeff_mxu.fits: coefficients of the spectral curvature fitting polynomials.

disp_coeff_mxu.fits: coefficients of the wavelength calibration fitting polynomials.

master_bias.fits: master bias frame, produced only in case a sequence of raw BIAS exposures was specified in input.

master_norm_flat_mxu.fits: normalised flat field image.

slit_location_mxu.fits: slit positions on the CCD (at central wavelength).

The products for checking the quality of the result are:

curv_traces_mxu.fits: table containing the y CCD positions of the detected spectral edges at different x CCD positions, with their modeling.

delta_image_mxu.fits: deviation from the linear term of the wavelength calibration fitting polynomials.

disp_residuals_mxu.fits: residuals for each wavelength calibration fit, produced only if the recipe configuration parameter "check" is set.

disp_residuals_table_mxu.fits: table containing different kinds of residuals for a sample of wavelength calibration fits.

global_distortion_table.fits: table containing the modeling of the coefficients listed in the *curv_coeff_mxu.fits* and *disp_coeff_mxu.fits* tables, only produced if more than 6 slits are available.

master_screen_flat_mxu.fits: bias corrected sum of all the input flat field exposures.

reduced_lamp_mxu.fits: rectified and wavelength calibrated arc lamp image.

slit_map_mxu.fits: map of the grism central wavelength on the CCD, produced only if the recipe configuration parameter "check" is set.

spatial_map_mxu.fits: map of spatial positions along each slit on the CCD.

spectra_detection_mxu.fits: result of preliminary wavelength calibration applied to the input arc lamp exposure, produced only if the recipe configuration parameter "check" is set.

spectra_resolution_mxu.fits: mean spectral resolution for each reference arc lamp line.

wavelength_map_mxu.fits: map of wavelengths on the CCD.

See the FORS Pipeline User's Manual for a more detailed description of these products.

4.2.2 Checking the results

Things can go wrong. In this Section a number of basic checks are suggested for ensuring that the *fors_calib* recipe worked properly. Troubleshooting is given separately, in the next Section, in order to avoid too many textual repetitions: it often happens, in fact, that different problems have the same solution. Three basic checks are described here: spectra localisation, wavelength calibration, and spectral resolution. It is advisable to perform such checks in the given order, because some results make only sense under the assumption that some previous tasks were performed appropriately. For instance, an apparently good wavelength calibration does not imply that the slit spectra were all properly traced.

Were all spectra detected and properly traced?

Compare (blink) the *master_norm_* and the *master_screen_flat_mxu.fits* images. The normalised flat field image can be used as a map showing where the spectra were found and how they were cut out from the CCD, while the master flat image shows where the spectra actually are. A quick visual inspection will immediately expose any badly traced, or even lost, spectrum. This kind of failure may not be so apparent in the *reduced_lamp_mxu.fits* image, which includes just what has been successfully extracted.

The *curv_traces_mxu.fits* table enables a closer look at the tracing accuracy. The tracings of the top and bottom edges of the spectrum from slit 10, for instance, are given in the table columns labeled "t10" and "b10", for each CCD pixel along the horizontal direction given in column "x". Each tracing may be compared with the fitted model: for instance, the modeling of the tracing "t10" is given in the table column "t10_mod", together with the fit residuals in column "t10_res", enabling the generation of plots like those shown in Figure 1. In order to reduce the residuals, the degree of the fitting polynomial may be increased (using the configuration parameter "cdegree"): it is however advisable to never use polynomials above the 2nd order, unless the residuals are really not acceptable. In Figure 1 the residuals are less than 3 hundreds of a pixel, and this is acceptable even if they display a systematic trend that may be easily eliminated by fitting a 3rd degree polynomial. When systematic trends in the residuals are so small (with respect to the pixel size), they can no longer be considered "physical", but rather an effect of the pixelisation of the edge changing with the position along the CCD. See the FORS Pipeline User's Manual for more details on this.

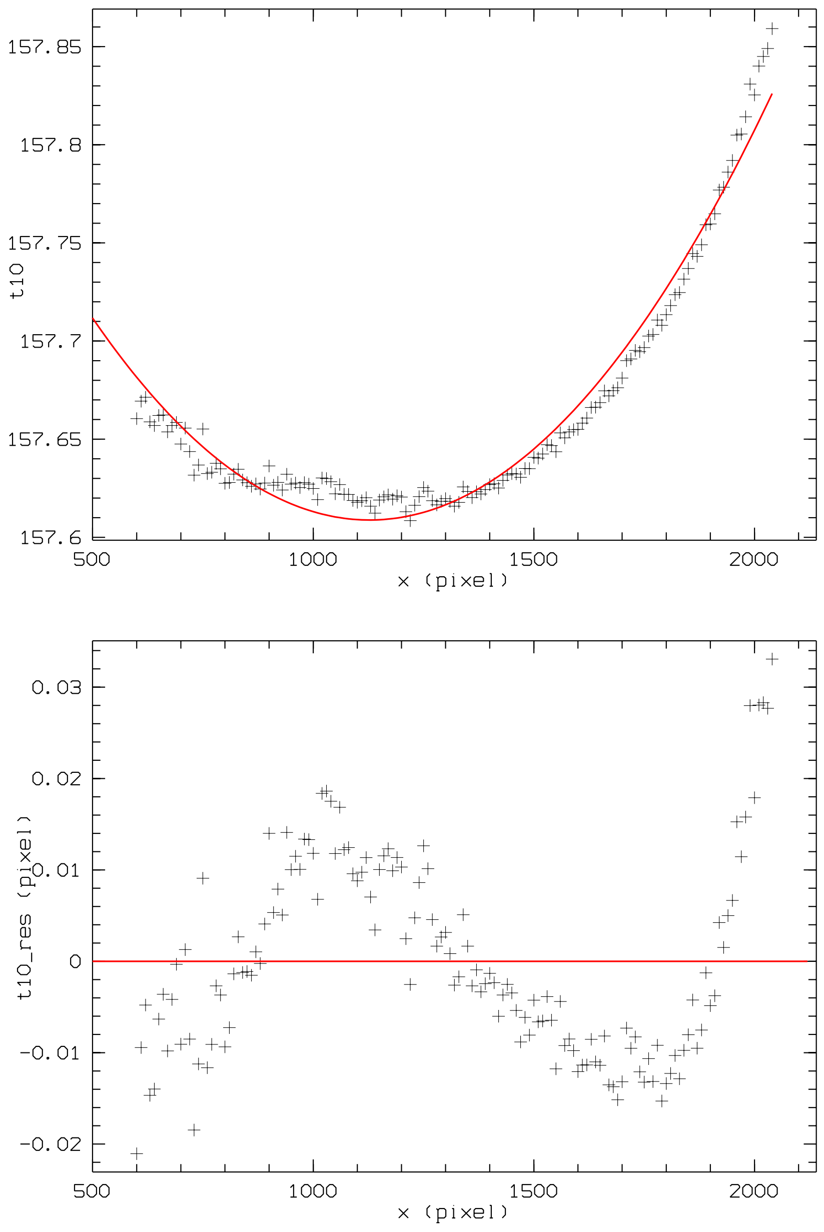


Figure 1: *Tracing, modeling, and systematic residuals (in pixel) of one spectral edge tracing.*

Were all spectra properly calibrated in wavelength?

Check the *reduced_lamp_mxu.fits* image first. This image contains the arc lamp spectra from each slit with all the optical and spectral distortions removed. The spectral lines should all appear perfectly aligned and vertical. Particular attention should be given to lines at the blue and red ends of each spectrum, where the polynomial fit is more sensitive to small variations of the signal. The calibrated slit spectra are vertically ordered as in the original CCD frame. The boundaries between individual slit spectra are generally easy to recognise: both because they are often dotted by the emission lines from nearby spectra on the original CCD frame, and because each slit spectrum may cover different wavelength intervals according to its position within the original CCD frame (see Figure 2). The position of each spectrum in the calibrated image is always reported in the table *slit_location_mxu.fits*, at the columns "position" and "length".

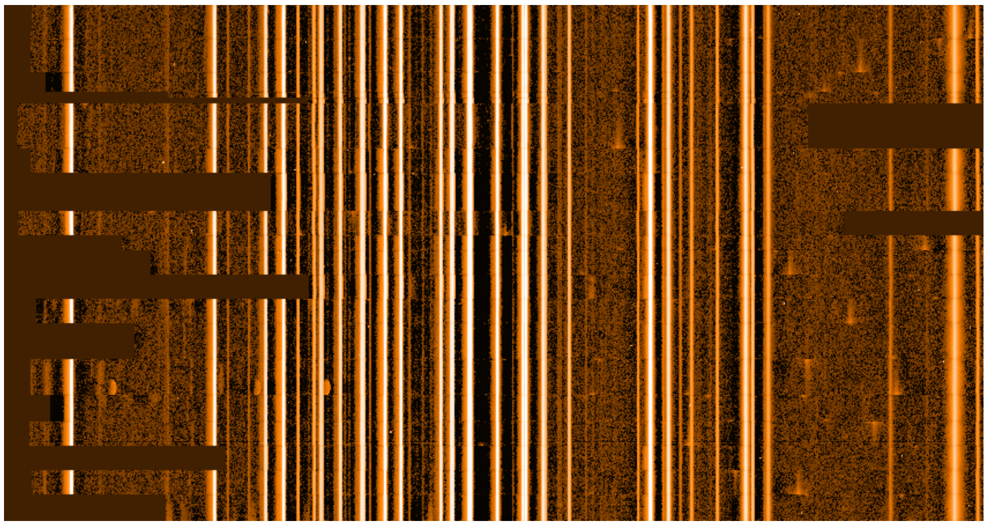
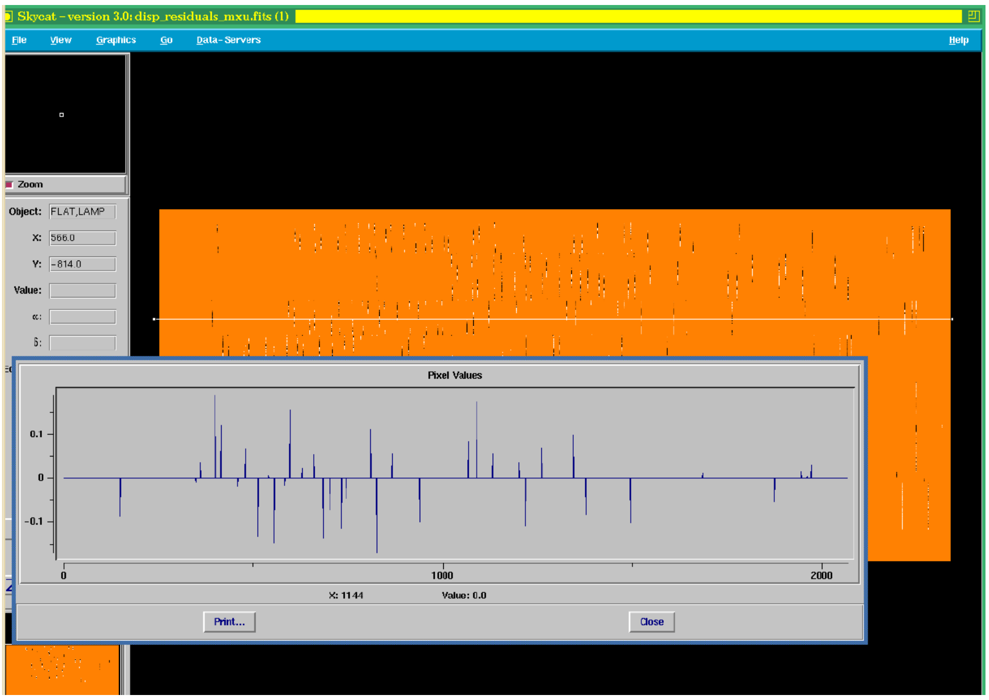


Figure 2: *Example of well calibrated FORS2 MXU 300I arc lamp spectra.*

More detailed checks on the quality of the solution can be made by examining other pipeline products. The image *disp_residuals_mxu.fits* contains the residuals of the wavelength solution for each row of each slit spectrum. This image is mostly padded with zeroes, with the only exception of the pixels where a reference line was detected and identified: those pixels report the value of the corresponding residual (in pixel). This image will in general be viewed applying small cuts (typically between -0.2 and 0.2 pixels): systematic trends in the residuals, along the dispersion direction, would appear as sequences of all-positive (white) followed by all-negative (black) residuals, in a wavy fashion, that could also be viewed by simply plotting a profile at different image rows (see Figure 3). Systematic residuals in the wavelength calibration are in general not acceptable, and they may be eliminated by increasing the order of the fitting polynomial.

Another product that can be used for evaluating the quality of the fit is the *disp_residuals_table_mxu.fits* table. Here the residuals are reported in a tabulated form for each wavelength in the reference lines catalog, but just for one out of 10 rectified image rows (*i.e.*, one out of 10 solutions). In conjunction with the *delta_image_mxu.fits* image, plots like the ones in Figure 4 can be produced. See the FORS Pipeline User's Manual for more details on this.

Finally, the table *disp_coeff_mxu.fits* might be examined to check how many arc lamp lines were used (column "nlines") and what is the mean uncertainty of the fitted wavelength calibration solution (column "error"), for each row of each slit spectrum. The model mean uncertainty is given at a 1-σ level, and has a statistical meaning only if the fit residuals do not display any systematic trend and have a random (gaussian) distribution around zero. Typically this uncertainty will be of the order of 0.05 pixels, *i.e.*, much smaller than the root-mean-squared residual of the fit, depending on the number of fitted points (a fit based on a large number of points is more accurate than a fit based on few points). It should be anyway kept in mind that the model uncertainty can be much larger than that (up to 1 pixel in the worst cases) at the blue and red ends of the fitted wavelength interval, as shown in Figure 5. This is because in the pipeline the wavelength solution is obtained by fitting a polynomial, rather than a physical model of



Object: FLAT,LAMP
X: 566.0
Y: -814.0
X: 1144 Value: 0.0

Figure 3: *Residual map from a FORS2 MXU 600RI arc lamp calibration. In the foreground is a plot of the residuals from one image row. In this ideal case the residuals do not display obvious systematic trends, and they are of the order of a tenth of a pixel.*

the instrument behaviour.

Is the spectral resolution as expected?

The table *spectra_resolution_mxu.fits* reports on the mean spectral resolution, defined as $R = \lambda/\Delta\lambda$ (with $\Delta\lambda$ determined at half-maximum), that was measured for each reference arc lamp line (see Figure 6). The standard deviation from this mean is also given, together with the number of independent determinations of R in column "nlines".

4.2.3 Troubleshooting

In this Section a set of possible solutions to almost any problem met with the *fors_calib* recipe is given. It is advisable to try them in the same order as they are listed here. It may be useful to go through this check list even in case the recipe seemed to work well: there might always be room for improvement.

In practice, almost any problem with the pipeline is caused by a failure of the pattern-recognition task. Pattern-recognition is applied to detect the slit spectra on the CCD, assuming that they all will include an illumination pattern similar to the pattern of wavelengths listed in the reference arc lamp line catalog (see the FORS Pipeline User's Manual for more details on this).

For an immediate visualisation of how successful was the pattern-recognition, just rerun the *fors_calib* recipe setting the "check" parameter to *true*. This will produce a number of extra (intermediate) products. One of them is the *spectra_detection_mxu.fits* image, a by-product of the pattern-recognition task, displaying a preliminary wavelength calibration of the CCD. This image has as many rows as the CCD: if at any CCD row the line catalog pattern is

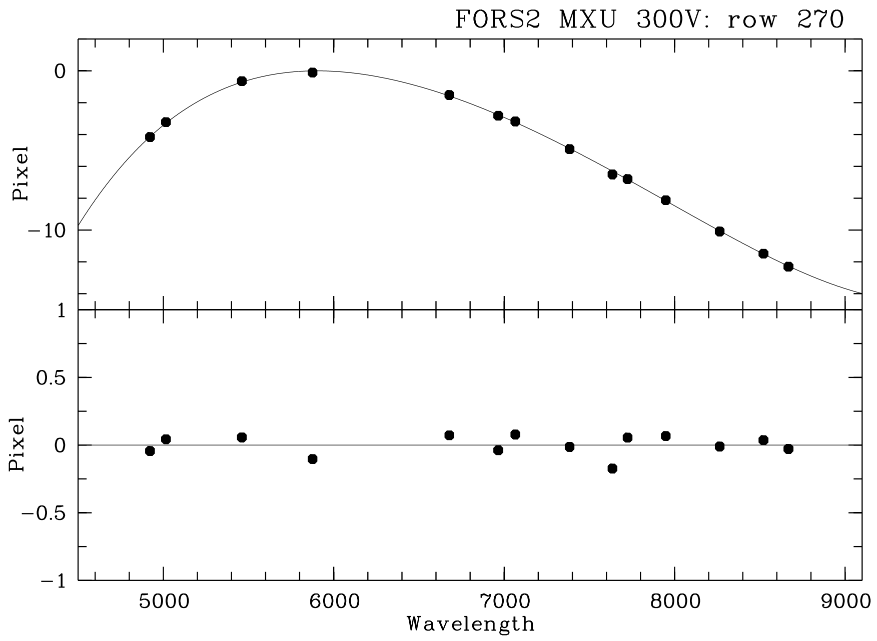


Figure 4: *Top panel: deviation of the identified peaks from the linear term of the 270th fitting polynomial (column d270 of the* disp_residuals_table_mxu.fits *table). The solid line is the polynomial model with the linear term subtracted, drawn from row 270 of the* delta_image_mxu.fits *image. Bottom panel: fit residuals of the identified peaks (identical to the residuals recorded at row 270 of the* disp_residuals_mxu.fits *image).*

detected, the spectral signal is wavelength calibrated, resampled at a constant wavelength step, and written to the same row of the *spectra_detection_mxu.fits* image. If a row of this image is empty, it is either because the corresponding CCD row doesn't contain any spectrum, or because the pattern-recognition task failed for that row. The check image may simply be placed side by side with the original CCD exposure, in order to see if and how frequently a spectral signal was not recognised as such. A few failures (*i.e.*, a few empty rows) are generally acceptable, as they are recovered by interpolation during the final wavelength calibration task. However, a high failure rate is probably the reason why a bad spectral localisation, or tracing, or final wavelength calibration, were possibly obtained.

What can make the pattern-recognition task fail? One or more of the following causes may be determined:

Some arc lamp reference lines are missing:

It is possible that the searched pattern is simply not present in the data: for instance, the Neon lamp was off, so only Argon + Helium lines are present.

Solution: Change line catalog accordingly.

Some arc lamp reference lines are very faint:

It is possible that the exposure time for the arc lamp frame is too short, or one of the lamps got too faint with age. If some of the reference lines listed in the catalog do not peak above a given threshold, they are not used by the pattern-matching task.

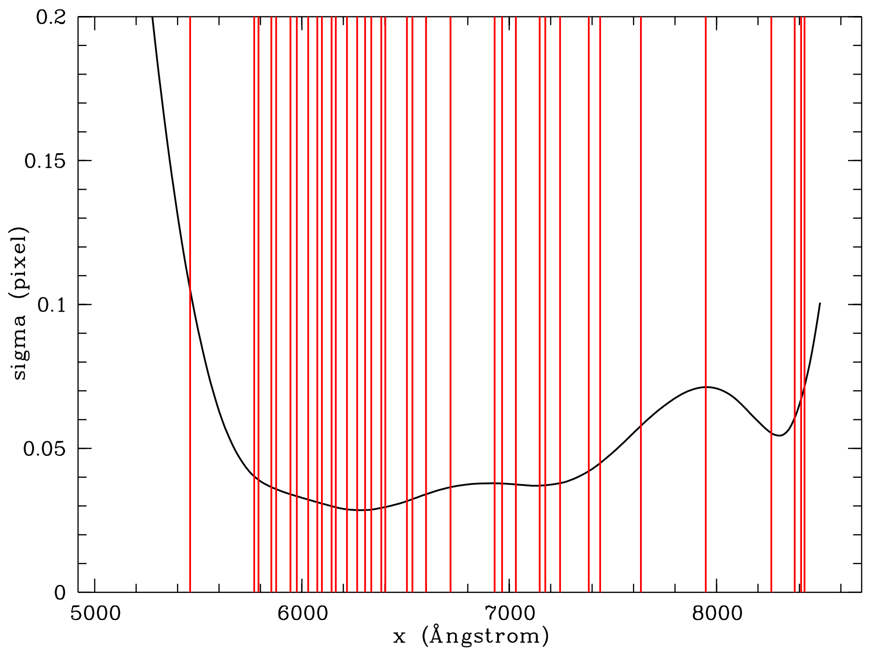


Figure 5: *Montecarlo simulation of wavelength calibration model accuracy (from FORS2/MXU 600RI calibration lamp). The curve describes the changing accuracy (in pixel), given at 1-σ confidence level. The vertical lines indicate the positions of the used reference arc lamp lines. It is possible to see that, as expected, the calibration is more accurate where the density of lines is higher. The accuracy of the model degrades rapidly at the blue and red ends of the spectrum, where uncertainties due to overfitting and extrapolation are greater.*

Solution: Specify a lower value for the "peakdetection" parameter. Alternatively, if this gets too close to the noise level, remove the faint lines from the reference line catalog.

The reference lines in the arc lamp exposure are very broad:

If very wide slits are used, the reference lines would become accordingly wider (and would display a box-like, flat-top profile). The calibration recipe can handle this in case of well isolated lines, but if nearby lines blend together it is impossible to safely determine their positions.

Solution: None. These spectra cannot be calibrated.

The spectral dispersion is not what expected:

The actual mean spectral dispersion is significantly higher (or lower) than expected. The first-guess spectral dispersion is specified via the parameter "dispersion", and is tabulated for each grism in the FORS1+2 User Manual, or in the grism tables that are included in the distributed FORS pipeline package. In general the pattern-recognition algorithm is quite robust against changes of the spectral dispersion (up to 20% from expectation), but for some grisms (such as the 600B in FORS1 and FORS2) good results can only be obtained within a much narrower window of values of the first-guess. For this reason a small change of the spectral dispersion (perhaps caused by a large temperature variation) may cause the wavelength calibration to fail.

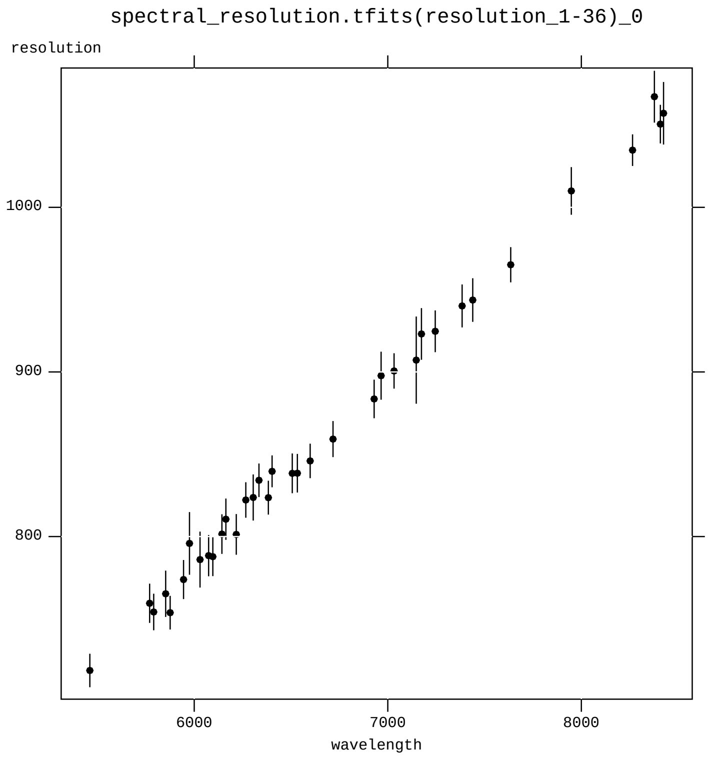


Figure 6: *Resolution vs. wavelength in a spectral_resolution_mxu.fits table derived from a FORS2 MXU 600RI arc lamp exposure.*

Solution: Try different values of the "dispersion" parameter around the expected (default) value, and select the one producing the lowest failure rate of the pattern-recognition task.

There are spectra at very large offsets:

The CCD may include spectra at such large x offsets that only part (red or blue) of their full wavelength range is really included in the CCD. If the line catalog contains too few reference lines in this region (say, less than 5), they might not be enough to define an unambiguous pattern to detect.

Solution: Add extra reference lines to the line catalog, for a more complete coverage of the bluest/reddest parts of the complete spectral range. If there are no extra lines to be used as a reference, the truncated spectra will then be definitely lost.

If the pattern-recognition seems to have worked properly, the reason of a *fors_calib* recipe failure can be found elsewhere:

The (MXU) mask includes tilted or curved slits:

Using the *fors_calib* recipe in its default configuration will have the pattern-matching based wavelength solution to be reused as a first-guess for a second iteration. Unfortunately this iteration doesn't work with tilted or curved slits (this will be fixed in the next pipeline releases).

Solution: The default of the parameter "wradius" is 4: setting it to zero would suppress the iteration step. In this way the wavelength solution will be based on pattern-matching only, and it will be insensitive to any strange shape the slits might have. This is however not robust in some cases.

Solution: Alternatively the iteration step could be maintained, but "wradius" should then be set to a larger value (at least the number of x pixels spanned by the tilted slit, typically around 10). This can only be attempted if the spectrum doesn't include lines that are closer to each others than "wradius", or misidentification may occur.

The spectra are too tightly packed:

If slits are too close to each others, there is a risk that (some of) the spectra would not be properly traced, or not traced at all, on the flat field frames. As a default, the *fors_calib* recipe tries to recover untraceable edges by interpolating a global curvature model based on other traceable edges (if they are available). Using this global description of the spectral curvature helps to extract also those spectra whose edges cannot be traced. In some cases however the recipe may find and accept a bad tracing as if it were good, producing a bad global curvature model, and therefore a bad spectral extraction.

Solution: Setting the parameter "cmode" to zero will suppress the usage of the global curvature model. In this case the recovery strategy of lost spectral edges will consist in replicating the trace of the other available spectral edge (opportunely shifted) of the same slit spectrum. This may improve the results in some cases: however, if a tracing is missing for both edges of a slit spectrum, the spectrum will not be extracted.

Other possible problems are simply related to the way an algorithm is run:

The wavelength calibration residuals display systematic trends:

Especially if the extracted spectral range is very large, the fitting polynomial may be incapable to replicate the physical relation between pixel and wavelength. In this case, any estimate of the statistical error (such as the fit uncertainties listed in *disp_coeff_mxu.fits*) will become meaningless.

Solution: Increase the degree of the fitting polynomial, using the parameter "wdegree". Beware that this may introduce overfitting, especially at the red and blue ends of the spectra (*i.e.*, the polynomial is so poorly constrained in those regions where few points are available, that it also fits their position uncertainty, incorporating this noise into the solution: the corresponding residuals may therefore look very small, and yet the calibrated spectra will appear to be badly calibrated; an extreme case of overfitting is, for instance, fitting 4 points with a 3rd degree polynomial: the residuals will be exactly zero, and yet the obtained model will be highly inaccurate).

The calibrated spectra look "noisy" at their ends:

This problem is symmetric to the previous one: the fit residuals may look very small, and yet the calibrated spectra will appear to be badly calibrated at their blue and red ends. This is the effect of model overfitting (see Figure 5).

Solution: Decrease the degree of the fitting polynomial, using the parameter "wdegree". Beware that this may introduce systematic fit residuals.

The flat field is not properly normalised:

The master flat field is normalised by dividing it by a smoothed version of itself. For various reasons the result may be judged unsatisfactory.

Solution: Change the smoothing box sizes using the parameters "dradius" and "sradius". Alternatively, instead of the default median smoothing, a polynomial may be used to fit the large scale trend: the degree of the fitting polynomial should be specified via the "ddegree" parameter.

Valid reference lines are rejected:

Sometimes the peak detection algorithm may return inaccurate positions of the detected reference arc lamp lines. Outliers are automatically rejected by the fitting algorithm, but if those lines were properly identified, not rejecting their positions may really improve the overall accuracy of the wavelength calibration.

Solution: Increase the value of the "wreject" parameter. Extreme care should be used here: a tolerant line identification may provide an apparently good fit, but if this is based on misidentified lines the calibration would include unknown systematic errors.

4.3 Processing the scientific data

In order to process scientific exposures the recipe *fors_science* must be used. Currently the scientific exposures can only be reduced one by one, as no combination of jittered exposures is yet provided by the pipeline. In this example it is assumed that the following data are available:

- One scientific exposure:

```
FORS2.2004-09-27T02:39:11.479.fits  SCIENCE_MXU
```

- All the relevant products of the *fors_calib* recipe (see the previous Section 4.2, page 23):

```
master_bias.fits            MASTER_BIAS
master_norm_flat_mxu.fits   MASTER_NORM_FLAT_MXU
disp_coeff_mxu.fits         DISP_COEFF_MXU
curv_coeff_mxu.fits         CURV_COEFF_MXU
slit_location_mxu.fits      SLIT_LOCATION_MXU
```

The same recommendations given for the calibration data reduction are valid here.

4.3.1 Running the recipe fors_science

In general the same grism table that was used in the *fors_calib* recipe would be specified in the input set-of-frames, that will look like this:

```
FORS2.2004-09-27T02:39:11.479.fits  SCIENCE_MXU
master_bias.fits                    MASTER_BIAS
master_norm_flat_mxu.fits           MASTER_NORM_FLAT_MXU
disp_coeff_mxu.fits                 DISP_COEFF_MXU
curv_coeff_mxu.fits                 CURV_COEFF_MXU
slit_location_mxu.fits              SLIT_LOCATION_MXU
FORS2_GRS_300I_21_OG590_32.fits     GRISM_TABLE
```

It is also possible to include a catalog of sky lines: this table should contain a column listing the wavelengths of the reference lines (the name of this column can be specified using the parameter "wcolumn", which is defaulted to "WLEN"), and it should be tagged MASTER_SKYLINECAT. If no sky lines catalog is specified in the SOF, an internal catalog is used instead.

Typically *fors_science* will run in less than 20 seconds (depending on the platform). The products that will be created depend on the chosen configuration parameter setting. Here is a list of all the possible products:

disp_coeff_sci_mxu.fits: wavelength calibration polynomials coefficients after alignment of the input solutions to the position of the sky lines. Only created if the parameter "skyalign" is set.

mapped_all_sci_mxu.fits: image with rectified and wavelength calibrated slit spectra. Always produced.

mapped_sci_mxu.fits: image with rectified, wavelength calibrated, and sky subtracted slit spectra. Only produced if at least one sky subtraction method is specified.

mapped_sky_sci_mxu.fits: image with rectified and wavelength calibrated slit sky spectra. Only produced if at least one sky subtraction method is specified.

object_table_sci_mxu.fits: positions of the spectra on the CCD and on the mapped images, and positions of the detected objects within the slits. Only created if at least one sky subtraction method is specified.

reduced_sci_mxu.fits: image with extracted objects spectra. Only created if at least one sky subtraction method is specified.

reduced_sky_sci_mxu.fits: image with the sky spectrum corresponding to each of the extracted objects spectra. Only created if at least one sky subtraction method is specified.

reduced_error_sci_mxu.fits: image with the statistical errors corresponding to the extracted objects spectra. Only created if at least one sky subtraction method is specified.

sky_shifts_slit_sci_mxu.fits: table containing the observed sky lines offsets that were used for adjusting the input wavelength solutions. Only created if the parameter "skyalign" is set.

unmapped_sci_mxu.fits: image with the sky subtracted scientific spectra on the CCD. Only created if either of the parameters "skylocal" and "skyglobal" are set.

unmapped_sky_sci_mxu.fits: image with the modeled sky spectra on the CCD. Only created if either of the parameters "skylocal" and "skyglobal" are set.

wavelength_map_sci_mxu.fits: map of wavelengths on the CCD. Only created if the parameter "skyalign" is set.

See the FORS Pipeline User's Manual for a more detailed description of these products.

Currently there is no support for a spectro-photometric calibration. However, a standard star exposure (tag: either STANDARD_MOS or STANDARD_LSS) can be reduced by the *fors_science* recipe as any generic scientific frame.

4.3.2 Checking the results

In this Section a number of basic checks are suggested for ensuring that the recipe *fors_science* worked properly. Troubleshooting is given separately, in the next Section, in order to avoid too many textual repetitions: it often happens, in fact, that different problems have the same solution. Four basic checks are described here: wavelength calibration, sky subtraction, object detection, and object extraction. It is advisable to perform such checks in the given order, because some results make only sense under the assumption that some previous tasks were performed appropriately. For instance, an apparently good sky subtraction does not imply that the slit spectra were all properly wavelength calibrated.

Were all spectra properly wavelength calibrated?

The wavelength calibration based on calibration lamps, performed at day-time, may not be appropriate for an accurate calibration of the scientific spectra: systematic differences due to instrumental effects, such as flexures, may intervene in the meantime.

To overcome this, the day calibration may be upgraded by testing it against the observed positions of the sky lines in the scientific slit spectra. The alignment of the input distortion models to the true sky lines positions is controlled by the parameter "skyalign", that as a default is set to 0 (*i.e.*, the sky lines correction will be a median offset, see the FORS Pipeline User's Manual for more details).

It is possible, however, that an alignment of the distortion models is unnecessary: if this were the case, it would be better to avoid it entirely (any extra manipulation increases the statistical uncertainties on the final product). In order to decide whether a sky alignment is necessary or not, the *sky_shifts_slit_sci_mxu.fits* table can be examined. This table has a column labeled "wave", listing the wavelengths of all the reference sky lines found within the extracted spectral interval, and a number of columns labeled "offset_*id*", listing the median offset in pixels for each sky line from its expected position, for the slit identified by "id" (see Figure 7). *Beware*: the listed offsets are not the residuals of the final sky line alignment, but really the comparison of the sky line positions against expectations from the input distortion models. In case the sky line offsets are compatible with zero, the sky line alignment is really unnecessary, and the *fors_science* recipe may be run again setting the "skyalign" parameter to -1 (*i.e.*, the sky lines correction will be disabled). This is not strictly necessary, but it is often wise to keep data manipulation to a minimum. On the other hand, observing systematic offsets would confirm that an alignment of the distortion model to the true sky lines positions was in order, and there would be no need to reprocess the data. In case the offset appears to depend on the wavelength, it may be appropriate to set the parameter "skyalign" to 1 (see also the Troubleshooting Section 4.3.3, page 38).

The overall quality of the wavelength calibration (whether a sky line alignment was applied or not) can be examined in the *mapped_all_sci_mxu.fits* image. This image contains the scientific spectra from each slit after removing the optical and spectral distortions. The visible sky lines should all appear perfectly aligned and vertical. The position of each spectrum in the calibrated image is listed in the table *object_table_sci_mxu.fits*, at the columns "position" and "length".

A further check on the quality of the solution can be made by examining the *disp_coeff_sci_mxu.fits* table. This table is only produced in case a sky line alignment was performed.

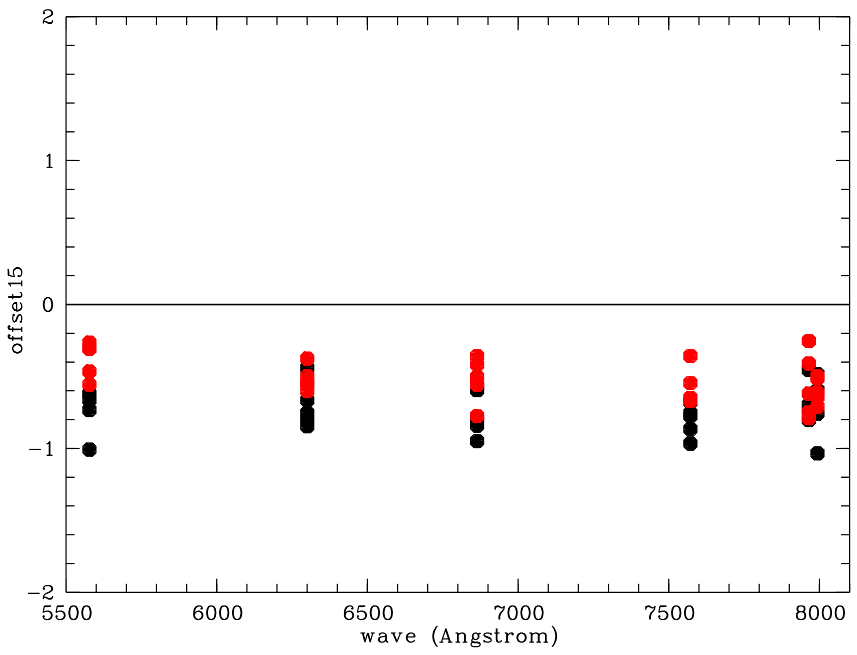


Figure 7: *Systematic sky line offsets (in pixel) from day-calibration expectation, observed in a FORS2 MXU 600RI scientific exposure. The offsets from all slits are plotted, black and red points belonging to slits from the upper and lower part of the CCD. All offsets are systematically negative, and therefore a sky alignment of the wavelength solution is due. Offsets depending on the slit position on the CCD, as shown by the different distribution of black and red points in figure, indicate that the scientific exposure is not just translated (e.g., by flexures) but also rotated with respect to day calibrations.*

Column "nlines" reports how many sky lines were used for the distortion model correction, while the "error" column reports the mean uncertainty of the new wavelength calibration solution for each slit spectrum row. The model uncertainty is given at a 1-σ level, and is computed as the quadratic mean of the input model accuracy and the sky line correction accuracy. Typically this uncertainty will be of the order of 0.1 pixel, *i.e.*, much smaller than the root-mean-squared residual of the lamp calibration and of the sky line correction, depending on the number of fitted points. It should be anyway kept in mind that the model uncertainty can be much larger than that (up to 1 pixel in the worst cases) at the blue and red ends of the fitted wavelength interval, as shown in Figure 5. This is because in the pipeline the wavelength solution is obtained by fitting a polynomial, rather than a physical model of the instrument behaviour.

Is the sky background properly subtracted?

A quick check on sky subtraction can be made by examining the sky subtracted frames, *mapped_sci_mxu.fits* and *unmapped_sci_mxu.fits* (if available, depending on how the recipe was run). The spectra should have a generally smooth look, and will only appear to be noisier in those regions where bright sky lines were subtracted.

The best way to ensure that the sky was subtracted optimally, at least at the positions of the objects to extract, is to check that the residual noise is compatible with the statistical error associated to the extracted object spectra. The extracted spectra are contained in the *reduced_sci_mxu.fits* image (one extracted spectrum for each row). Their error spectra (at a 1-σ level) are contained in the *reduced_error_sci_mxu.fits* image. The regions of the extracted spectra corresponding to a (bright) sky line will include a few noisier points, whose deviation from the spectral continuum should (almost) never pass the 3-σ deviation. If this condition is fulfilled, the sky subtraction is probably as good as it can get.

Note that the subtracted sky can be viewed in the images *mapped_* and *unmapped_sky_sci_mxu.fits*. More useful is perhaps the image containing the extracted sky spectra, *reduced_sky_sci_mxu.fits*: such spectra are extracted applying to the modeled slit sky spectra exactly the same weights that were used in the object extraction.

Were all objects detected?

The answer to this question is almost always "no". The pipeline, after removing the instrument signature and the sky background from each slit spectrum, will run an object detection algorithm in order to find all the objects that need to be extracted. There will always be a detection threshold beyond which an object will not be significant enough for selection – no matter what detection algorithm is applied. Using more tolerant detection criteria would not eliminate the threshold effect, and may increase the number of false detections to the point of making the object detection task impractical.

The list of detected objects can be found in the *object_table_sci_mxu.fits* table.

Were all the detected objects properly extracted?

As a default the *fors_science* recipe would apply an optimal extraction algorithm to each detected object spectrum. This algorithm (see the FORS Pipeline User's Manual for more details) is only appropriate for point-like objects emitting over (almost) all the extracted wavelength interval, while it is not appropriate for extended objects, and it is ineffective for objects having a spectrum only consisting of emission lines with no continuum.

The statistical noise on the extracted object spectra should in principle decrease if the spectra are optimally extracted. In order to check the improvement of the noise level, it is easy to compare the *reduced_error_sci_mxu.fits* images obtained by running the recipe with and without optimal extraction. A 30% increase of the signal-to-noise ratio can be obtained for faint-objects (background-noise limited), while there would be little or no improvement for brighter sources. The photometric accuracy of the optimal extraction can be checked by simply computing the ratio (or the difference) of the *reduced_sci_mxu.fits* images obtained once with the standard and once with the optimal extraction: the result should be a flat image, displaying no trends or systematic deviations from 1 (or 0).

4.3.3 Troubleshooting

In this Section a set of possible solutions to the most common problems with the *fors_science* recipe is given. It is advisable to try them in the same order as they are listed here. It may be useful to go through this check list even in case the recipe seemed to work well: there might always be room for improvement.

The wavelength calibration is bad:

Aligning the wavelength calibration to the position of the observed sky lines may be inaccurate, especially if very few reference lines are used. If a sky line alignment is really required (see previous Section), then action needs to be taken to solve this problem.

Solution: If very few reference sky lines are used, supplying a sky line catalog including more lines (even if weak and/or blended) may help a lot.

Solution: If the wavelength calibration appears to be bad just at the blue and/or red ends of the spectra, go back to the *fors_calib* recipe to obtain a more stable wavelength calibration in those regions (*e.g.*, either by adding new reference arc lamp lines, or by decreasing the fitting polynomial degree).

The sky alignment of the wavelength solution failed:

In case a blue grism is used, or if a spectrum has a large offset toward the red, no sky lines may be visible within the observed spectral range.

Solution: None. It is however possible to modify the columns of coefficients in the input *disp_coeff_mxu.fits* table, if the correction can be evaluated in some other way. For instance, the solution can be shifted by adding a constant value (in pixel) to column "c0".

The sky subtraction failed for resolved sources:

In case of extended objects filling most or all of the slit, the evaluation of the sky may be strongly biased by the inclusion of signal that actually belongs to the object to extract. Subtracting this contaminated background would actually destroy the object spectrum.

Solution: The default sky subtraction method (parameter "skylocal") performs very well for point-like sources where plenty of sky is directly observable within the slit. An alternative method is made available for extended objects (parameter "skyglobal"). Setting "skylocal" to *false* and "skyglobal" to *true* would subtract from all spectra a supersampled model of the median sky spectrum observed in all slits. This method would perform optimally only in case the spectral resolution were the same all over the detector: in practice, this method is always less accurate than the "skylocal" method (see the FORS Pipeline User's Manual for more details on this). But even if this method is less accurate, it remains the extended sources best friend. It is always possible to process the scientific exposures in both ways, one for processing point-like sources and the other for processing spatially resolved sources.

The sky subtraction failed for curved or tilted slits:

Obvious residuals related to the sky subtraction are visible on the extracted slit spectra.

Solution: Change sky subtraction method: set "skylocal" to *false* and "skymedian" to *true*. The difference between the two methods is that "skylocal" would subtract the sky *before*, and "skymedian" *after* the rectification of the spectral data. The second method performs very poorly in comparison to the first, but in the case of curved or slanted slits there is at the moment no other choice than using it. This problem will be fixed in the next pipeline releases.

Cosmic rays are not removed:

As a default the *fors_science* recipe does not remove cosmic rays hits, leaving them on the sky-subtracted slit spectra: if the optimal spectral extraction is applied, most of the cosmics are removed anyway from the extracted spectra. Optimal extraction is however not always applicable, especially in the case of resolved sources.

Solution: Set the "cosmics" parameter to *true*. This will apply a cosmics removal algorithm to the sky subtracted spectra. The removed cosmic rays hits will be included in the (modeled) sky images, *mapped_sky_sci_mxu.fits* and *unmapped_sky_sci_mxu.fits* (see the FORS Pipeline User's Manual for more details).

The sampling of the remapped scientific spectra is poor:

When the slit spectra are rectified and wavelength calibrated, they are remapped undistorted to images such as *mapped_sky_sci_mxu.fits* or *mapped_sci_mxu.fits*. This remapping may be judged to undersample the signal along the dispersion direction.

Solution: Change the value of the "dispersion" parameter. This parameter doesn't need to be identical to the one used in the *fors_calib* recipe.

It should be noted, however, that making the sampling step smaller will not really increase the information contained in the remapped spectra. As a matter of fact, even maintaining a resampling step close to the original CCD pixel size, the remapped pixel values would still be obtained by interpolating the values from a number of original pixels that are close to the interpolation point: nearby interpolation points would surely share common evaluators, and this is what introduces correlated noise in the result. Decreasing the resampling step would just increase this effect. In general, working on remapped spectra means to accept that the spectral signal was heavily manipulated, and it is for this very reason that the *fors_science* recipe also produces reduced – but still unmapped – spectra, as in the *unmapped_sci_mxu.fits* image.

The extracted spectra are normalised in time:

The default behaviour of this recipe is to normalise the results to the unit exposure time.

Solution: Set the parameter "time_normalise" to *false*.

There are often spurious objects detected at the slit edges:

As a default the *fors_science* recipe excludes objects that are detected within 3 pixels from the slit ends. This might not be enough in some cases.

Solution: Increase the value of the "slit_margin" parameter.

Some "obvious" objects are not detected:

Examining the *mapped_sci_mxu.fits* and *unmapped_sci_mxu.fits* images it may appear that some clearly visible object spectra are not detected (let alone extracted) by the recipe.

Solution: Setting "cosmics" to *true* (cleaning cosmic rays hits) may help.

Solution: Try different set of values for the parameters "ext_radius" and "cont_radius".

5 Multi Object Spectroscopy

Written by Chris Lidman, ESO Paranal Science Operations

5.1 Introduction

The guidelines described here are just that - guidelines. How you choose to reduce the data depends on the scientific goals of the program and the manner in which the data were taken. For example, if you have not dithered the source along the slit, then it is not possible to use the 2 dimensional sky subtraction technique that is described later.

The methods described here are the methods used by the author. They have been developed with aim of extracting the spectra of very faint targets that have been observed at very red wavelengths. It is not the only way to reduce the spectroscopic data and it may not be the best. It is not my purpose to go into exhaustive detail of all the possibilities as that would lead to a very, very large document and you would never read it.

I encourage you to experiment with different reduction strategies. The extra effort is worth it. You'll gain a better understanding of the quality of the data and the capabilities of the instrument, and you'll better prepared for your next observating run. As a general rule, be very wary of realtime pipelines. They are good for getting a rough idea of the quality of the data, but that is all.

The procedures to reduced MOS data, whether it is taken with MXU masks or the MOS blades, can also be used to reduce long slit data.

To reduce MOS data, you need bias frames (although these are not essential), flat fields, arc frames, a standard (and the associated calibrations that go with the standard) and the data itself.

5.2 Basic reduction steps

5.2.1 Removing the detector bias

This should be applied to all data, whether it is science frame or a calibration frame.

There are several ways to do this. One can subtract a bias frame, which is usually the clipped average of the 5 biases that are taken during morning calibrations, one can subtract a fit to the overscan region or one can subtract a constant. The second method - using the overscan region - is adequate, since there is not a lot of structure in the bias frames of FORS1 and FORS2. Subtracting a 2 dimensional bias frame just adds noise to the data.

FORS has an overscan and a prescan. The size and location of these regions are specified in the FITS header. As an example, the regions for the master chip of FORS2 are listed below.

```
HIERARCH ESO DET OUT1 PRSCX  =  0 / Prescan region in X
HIERARCH ESO DET OUT1 OVSCX  =  0 / Overscan region in X
HIERARCH ESO DET OUT1 PRSCY  =  5 / Prescan region in Y
HIERARCH ESO DET OUT1 OVSCY  =  5 / Overscan region in Y
```

My preferred routine for removing the overscan is the IRAF ccdproc task. The most important parameters of this task are

```
PACKAGE  = ccdred
TASK     = ccdproc
images   = frame        List of CCD images to correct
(oversca = yes          ) Apply overscan strip correction?
(readaxi = column       ) Read out axis (column|line)
(functio = legendre     ) Fitting function
(order   = 5            ) Number of polynomial terms or spline pieces
(sample  = *            ) Sample points to fit
(naverag = 1            ) Number of sample points to combine
(niterat = 1            ) Number of rejection iterations
(low_rej = 3.           ) Low sigma rejection factor
(high_rej= 3.           ) High sigma rejection factor
(grow    = 0.           ) Rejection growing radius
```

This routine does many other things other than fitting and subtracting the overscan. For example, it can be used to trim the image and to flat field it. A full listing of all the parameters is given below.

```
PACKAGE  = ccdred
TASK     = ccdproc

images   = frame        List of CCD images to correct
(output  =              ) List of output CCD images
(ccdtype =              ) CCD image type to correct
(max_cac = 0            ) Maximum image caching memory (in Mbytes)
(noproc  = no           ) List processing steps only?

(fixpix  = no           ) Fix bad CCD lines and columns?
(oversca = yes          ) Apply overscan strip correction?
(trim    = no           ) Trim the image?
(zerocor = no           ) Apply zero level correction?
(darkcor = no           ) Apply dark count correction?
(flatcor = no           ) Apply flat field correction?
(illumco = no           ) Apply illumination correction?
(fringec = no           ) Apply fringe correction?
(readcor = no           ) Convert zero level image to readout correction?
(scancor = no           ) Convert flat field image to scan correction?

(readaxi = column       ) Read out axis (column|line)
(fixfile =              ) File describing the bad lines and columns
(biassec = [*,1:4]      ) Overscan strip image section
(trimsec =              ) Trim data section
(zero    =              ) Zero level calibration image
(dark    =              ) Dark count calibration image
(flat    =              ) Flat field images
(illum   =              ) Illumination correction images
```

```
(fringe  =              ) Fringe correction images
(minrepl = 1.           ) Minimum flat field value
(scantyp = shortscan    ) Scan type (shortscan|longscan)
(nscan   = 1            ) Number of short scan lines

(interac = yes          ) Fit overscan interactively?
(functio = legendre     ) Fitting function
(order   = 5            ) Number of polynomial terms or spline pieces
(sample  = *            ) Sample points to fit
(naverag = 1            ) Number of sample points to combine
(niterat = 1            ) Number of rejection iterations
(low_rej = 3.           ) Low sigma rejection factor
(high_re = 3.           ) High sigma rejection factor
(grow    = 0.           ) Rejection growing radius
(mode    = ql           )
```

After subtracting the overscan, there can still be a residual positive offset. In part, this is due to dark current, which is typically a few electrons per hour. However, most of it is due to features of the instrument, such as scattered light. Subtracting a constant is sufficient to remove this offset, although the author rarely does this.

5.2.2 Flat fielding

First, one has to create the flat field. Typically, three to five flats are taken for each setup. After the overscan has been removed, the frames can be averaged with suitable clipping for cosmic rays. Alternatively, the median can be used. The routine of choice is the IRAF imcombine task. This flexible and powerful task can do just about anything. It is well worth the effort required to master it.

Since the readout noise and gain of each chip is known (I've used FORS2 as an example), one can set

```
(reject  = crreject     ) Type of rejection
(hsigma  = 8.           ) Upper sigma clipping factor
(rdnoise = 0.7          ) ccdclip:  CCD readout noise (electrons)
(gain    = 2.9          ) ccdclip:  CCD gain (electrons/DN)
```

however, setting

```
(reject  = minmax       ) Type of rejection
(nlow    = 1            ) minmax:  Number of low pixels to reject
(nhigh   = 1            ) minmax:  Number of high pixels to reject
```

works just as well, although one is unecessarily throwing away good pixels.

In both cases don't forget to set

```
(scale   = median      ) Image scaling
```

The next step is to remove the lamp shape.

For MOS or MXU data, I usually slice an image according to the position and number of slits. I then work on each slit independently. This allows me greater flexibility when reducing data, although it can increase the number of files drastically. The most important step is to trim the left and right hand edges of the spectra where the flux drops to zero. Failure to do so will cause some of the following tasks (`apnormalise` and `apflatten`) to produce wiggles near the ends of the spectrum when the lamp shape is fitted with a polynomial.

Removing the lamp shape can be complex, because the spectral trace does not necessarily follow detector rows and there is scattered light from neighbouring slits that contaminate the edges of the slit. Both effects become stronger as one reaches the edge of the field of view. A good way to see the effect is to look at an arc frame.

There are various routines one can use. One of the simplest tasks is the response task in the longslit package.

```
PACKAGE  = longslit
TASK     = response

calibrat = flat        Longslit calibration images
normaliz = flat[*,30:80]Normalization spectrum images
response = flat_norm   Response function images
(interac = yes         ) Fit normalization spectrum interactively?
(thresho = 500.        ) Response threshold
(sample  = *           ) Sample of points to use in fit
(naverag = 1           ) Number of points in sample averaging
(functio = legendre    ) Fitting function
(order   = 30          ) Order of fitting function
(low_rej = 5.          ) Low rejection in sigma of fit
(high_re = 5.          ) High rejection in sigma of fit
(niterat = 1           ) Number of rejection iterations
(grow    = 0.          ) Rejection growing radius
(graphic = stdgraph    ) Graphics output device
(cursor  =             ) Graphics cursor input
(mode    = h           )
```

In this particular example, we only use the cetral 51 rows to do the fit, and all pixels that have counts below 500 ADU are set to 1. A fitting order of 30 is necessary to remove large scale variations, but leaves small scale structure - most notably - the fringes intact.

This task does not account for curvature in the trace, nor does it remove scattered light. The author has experimented with tasks such as apnormalise to account for the former and apflatten for the latter, but has chosen to simply trim problematic regions, since these regions usually amount to a few pixels in most cases. Also, scattered light in the flat field and/or the science images does not affect the 2 dimensional sky subtraction techniques that are discussed later. It has a bigger impact on 1 dimension sky subtraction techniques.

If the trace of the spectra are heavily curved, then you might want to consider using the apnormalise or apflatten tasks in the IRAF apextact package. It is very similar to the response package. The main difference is that the fit is done parallel to the trace that is defined in the reference image.

```
PACKAGE  = apextract
TASK     = apnormalize

input    = flat        List of images to normalize
output   = flat_norm   List of output normalized images
(apertur =             ) Apertures
(referen = reference   ) List of reference images

(interac = yes         ) Run task interactively?
(find    = no          ) Find apertures?
(recente = no          ) Recenter apertures?
(resize  = no          ) Resize apertures?
(edit    = yes         ) Edit apertures?
(trace   = no          ) Trace apertures?
(fittrac = yes         ) Fit traced points interactively?
(normali = yes         ) Normalize spectra?
(fitspec = yes         ) Fit normalization spectra interactively?

(line    = INDEF       ) Dispersion line
(nsum    = 10          ) Number of dispersion lines to sum or median
(cennorm = no          ) Normalize to the aperture center?
(thresho = INDEF       ) Threshold for normalization spectra

(backgro = none        ) Background to subtract
(weights = none        ) Extraction weights (none|variance)
(pfit    = fit1d       ) Profile fitting type (fit1d|fit2d)
(clean   = no          ) Detect and replace bad pixels?
(skybox  = 1           ) Box car smoothing length for sky
(saturat = INDEF       ) Saturation level
(readnoi = 0.          ) Read out noise sigma (photons)
(gain    = 1.          ) Photon gain (photons/data number)
(lsigma  = 4.          ) Lower rejection threshold
(usigma  = 4.          ) Upper rejection threshold

(functio = legendre    ) Fitting function for normalization spectra
(order   = 30          ) Fitting function order
(sample  = *           ) Sample regions
(naverag = 1           ) Average or median
(niterat = 1           ) Number of rejection iterations
(low_rej = 3.          ) Lower rejection sigma
```

```
(high_re  = 3.          ) High upper rejection sigma
(grow     = 0.          ) Rejection growing radius
(mode     = ql          )
```

The main disadvantages are that only the data with the aperture are normalised (there is no normalize parameter) and the threshold parameter is interpreted differently. This task does not treat scattered light.

To remove scattered light, use apflatten. This will do a "line-by-line" fit to the spectral shape. It has the same disadvantages as those discussed for apnormalise with the additional disadvantage that it is more senstive to regions of low quantum efficiency. Note that this routine will remove the slit profile completely, whereas the two previous tasks retain it. In MOS, MXU and LLS data, you generally want to keep the slit ptofile in the flat.

```
PACKAGE   = apextract
TASK      = apflatten

input     = flat        List of images to flatten
output    = flat_norm   List of output flatten images
(apertur  =             ) Apertures
(referen  = reference   ) List of reference images

(interac  = yes         ) Run task interactively?
(find     = no          ) Find apertures?
(recente  = no          ) Recenter apertures?
(resize   = no          ) Resize apertures?
(edit     = yes         ) Edit apertures?
(trace    = no          ) Trace apertures?
(fittrac  = yes         ) Fit traced points interactively?
(flatten  = yes         ) Flatten spectra?
(fitspec  = yes         ) Fit normalization spectra interactively?

(line     = INDEF       ) Dispersion line
(nsum     = 10          ) Number of dispersion lines to sum or median
(thresho  = 10.         ) Threshold for flattening spectra

(pfit     = fit1d       ) Profile fitting type (fit1d|fit2d)
(clean    = no          ) Detect and replace bad pixels?
(saturat  = INDEF       ) Saturation level
(readnoi  = 0.          ) Read out noise sigma (photons)
(gain     = 1.          ) Photon gain (photons/data number)
(lsigma   = 4.          ) Lower rejection threshold
(usigma   = 4.          ) Upper rejection threshold

(functio  = legendre    ) Fitting function for normalization spectra
(order    = 30          ) Fitting function order
(sample   = *           ) Sample regions
```

```
(naverag = 1            ) Average or median
(functio = legendre     ) Fitting function for normalization spectra
(order   = 30           ) Fitting function order
(sample  = *            ) Sample regions
(naverag = 1            ) Average or median
(niterat = 2            ) Number of rejection iterations
(low_rej = 3.           ) Lower rejection sigma
(high_re = 3.           ) High upper rejection sigma
(grow    = 1.           ) Rejection growing radius
```

The scattered light seems to be less in the science images. It might be possible to use these images to remove the scattered light in the flats. This has never been investigated in much depth by this author.

Once you have the flat, you simply divide it into the science images. The ccdproc task can be used for this.

```
PACKAGE  = ccdred
TASK     = ccdproc

images   = 2dspectrum  List of CCD images to correct
(output  =             ) List of output CCD images
(ccdtype =             ) CCD image type to correct
(max_cac = 0           ) Maximum image caching memory (in Mbytes)
(noproc  = no          ) List processing steps only?

(fixpix  = no          ) Fix bad CCD lines and columns?
(oversca = no          ) Apply overscan strip correction?
(trim    = no          ) Trim the image?
(zerocor = no          ) Apply zero level correction?
(darkcor = no          ) Apply dark count correction?
(flatcor = yes         ) Apply flat field correction?
(illumco = no          ) Apply illumination correction?
(fringec = no          ) Apply fringe correction?
(readcor = no          ) Convert zero level image to readout correction?
(scancor = no          ) Convert flat field image to scan correction?

(readaxi = column      ) Read out axis (column|line)
(fixfile =             ) File describing the bad lines and columns
(biassec =             ) Overscan strip image section
(trimsec =             ) Trim data section
(zero    =             ) Zero level calibration image
(dark    =             ) Dark count calibration image
(flat    = flat_norm   ) Flat field images
(illum   =             ) Illumination correction images
(fringe  =             ) Fringe correction images
(minrepl = 1.          ) Minimum flat field value
```

```
(scantyp = shortscan   ) Scan type (shortscan|longscan)
(nscan   = 1           ) Number of short scan lines

(interac = yes         ) Fit overscan interactively?
(functio = legendre    ) Fitting function
(order   = 5           ) Number of polynomial terms or spline pieces
(sample  = *           ) Sample points to fit
(naverag = 1           ) Number of sample points to combine
(niterat = 1           ) Number of rejection iterations
(low_rej = 3.          ) Low sigma rejection factor
(high_re = 3.          ) High sigma rejection factor
(grow    = 0.          ) Rejection growing radius
(mode    = ql          )
```

5.2.3 Mapping the slit curvature

Generally speaking, night sky lines do not lie along detector columns and are often curved. This is commonly called slit curvature. Characterising the slit curvature is relatively straightforward and is usually done with arc spectra or night sky lines, although the latter might not work so well in short exposures, where they will be faint, or at bluer wavelengths, where there are few.

Most frequently, the map is done from detector pixel space to a space in which the horizontal axis in units of wavelength and the vertical axis is in pixels. Alternatively, and this is the author's prefered method, you can choose to map from detector pixel space to a second pixel space. The wavelength calibration is done after the spectra are extracted.

There are three IRAF tasks that one uses for this - `identify`, `reidentify` and `fitcoords`. A fourth IRAF task - `transform` - can be used to straighten the lines. It is beyond the purpose of this doucument to explain all the options in these tasks. However, we'll describe the most important ones.

The IRAF `identify` task extracts the a one dimentional spectrum from the 2 dimensional image (it can be used on 1d images as well) and the user is required to identify features. The task will then fit the identified features with a polynomial.

The most important parameters are

```
(section = middle line ) Section to apply to two dimensional images
```

which identifies which part of the two dimentional image to extract and

```
(coordli =             ) User coordinate list
(order   = 2           ) Order of coordinate function
```

If you are mapping from pixel space to wavelength space then a 4th order fit is more appropriate.

A full parameter listing of this task follows.

```
PACKAGE  = onedspec
TASK     = identify

images   = Arc          Images containing features to be identified
(section = middle line ) Section to apply to two dimensional images
(databas = database    ) Database in which to record feature data
(coordli =             ) User coordinate list
(units   =             ) Coordinate units
(nsum    = 10          ) Number of lines/columns/bands to sum in 2D images
(match   = -3.         ) Coordinate list matching limit
(maxfeat = 50          ) Maximum number of features for automatic identification
(zwidth  = 100.        ) Zoom graph width in user units
(ftype   = emission    ) Feature type
(fwidth  = 4.          ) Feature width in pixels
(cradius = 6.          ) Centering radius in pixels
(thresho = 0.          ) Feature threshold for centering
(minsep  = 2.          ) Minimum pixel separation
(functio = legendre    ) Coordinate function
(order   = 2           ) Order of coordinate function
(sample  = *           ) Coordinate sample regions
(niterat = 0           ) Rejection iterations
(low_rej = 3.          ) Lower rejection sigma
(high_re = 3.          ) Upper rejection sigma
(grow    = 0.          ) Rejection growing radius
(autowri = no          ) Automatically write to database
(graphic = stdgraph    ) Graphics output device
(cursor  =             ) Graphics cursor input
crval    =              Approximate coordinate (at reference pixel)
cdelt    =              Approximate dispersion
(aidpars =             ) Automatic identification algorithm parameters
(mode    = ql          )
```

Once run, the task will write a file called idArc to the database directory.
The IRAF `reidentify` task reidentifies features found during when the `identify` task was run over the entire 2 dimensional image. The most important parameters of this complex task are

```
referenc =              Arc Reference image
images   =              Arc Images to be reidentified
```

which need to be the same as the name of the image used in the `identify` task,

```
(section = middle line ) Section to apply to two dimensional images
```

which identifies the direction to extract spectra, and

```
(trace   = yes         ) Trace reference image?
(step    = 5           ) 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.          ) Search radius
(nlost   = 2           ) Maximum number of features which may be lost
```

which define how the trace along arc lines is done.
A full parameter listing of this task follows.

```
PACKAGE  = onedspec
TASK     = reidentify

referenc =             Arc Reference image
images   =             Arc Images to be reidentified
(interac = yes         ) Interactive fitting?
(section = middle line ) Section to apply to two dimensional images
(newaps  = yes         ) Reidentify apertures in images not in reference?
(overrid = yes         ) Override previous solutions?
(refit   = yes         ) Refit coordinate function?

(trace   = yes         ) Trace reference image?
(step    = 5           ) 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.          ) Search radius
(nlost   = 2           ) Maximum number of features which may be lost

(cradius = 8.          ) Centering radius
(thresho = 0.          ) Feature threshold for centering
(addfeat = no          ) Add features from a line list?
(coordli =             ) User coordinate list
(match   = -3.         ) Coordinate list matching limit
(maxfeat = 50          ) Maximum number of features for automatic identification
(minsep  = 2.          ) Minimum pixel separation

(databas = database    ) Database
(logfile = logfile     ) List of log files
(plotfil =             ) Plot file for residuals
(verbose = no          ) Verbose output?
(graphic = stdgraph    ) Graphics output device
(cursor  =             ) Graphics cursor input
```

```
answer   = YES          Fit dispersion function interactively?
crval    =              Approximate coordinate (at reference pixel)
cdelt    =              Approximate dispersion
(aidpars =              ) Automatic identification algorithm parameters
(mode    = ql           )
```

Once run, the task will update the file called idArc in the database directory.

The IRAF `fitcoords` task fits a polynomial to the features that were reidentified in the identify task. The task is in the `twodspec.longslit`. Before executing the task, don't forget to set the following package parameter.

```
PACKAGE  = twodspec
TASK     = longslit

dispaxis = 1            Dispersion axis (1=along lines, 2=along columns, 3=along z)
```

```
PACKAGE  = longslit
TASK     = fitcoords

images   =              Arc Images whose coordinates are to be fit
(fitname =              ) Name for coordinate fit in the database
(interac = yes          ) Fit coordinates interactively?
(combine = no           ) Combine input coordinates for a single fit?
(databas = database     ) Database
(deletio = deletions.db) Deletion list file (not used if null)
(functio = chebyshev    ) Type of fitting function
(xorder  = 2            ) X order of fitting function
(yorder  = 4            ) Y order of fitting function
(logfile = STDOUT,logfile) Log files
(plotfil = plotfile     ) Plot log file
(graphic = stdgraph     ) Graphics output device
(cursor  =              ) Graphics cursor input
(mode    = ql           )
```

Again, if you are mapping from pixel space to wavelength space the x order should be 4.

Experience has shown that the functional form of the distortion is generally more complex than what can be fitted with this task. The author has not investigated what might be better.

5.2.4 Applying the distortion correction (optional and not recommended)

This task involves interpolation. In the author's experience, interpolation makes the noise correlated, blurs cosmic rays and leads to large residuals when subtracting the sky near bright night sky lines. If you still want to do it, then you can use the IRAF task `transform`.

```
PACKAGE  = longslit
TASK     = transform

input    = frame4      Input images
output   = frame4_corrected Output images
(minput  =             ) Input masks
(moutput =             ) Output masks
fitnames = Arc1        Names of coordinate fits in the database
(databas = database    ) Identify database
(interpt = linear      ) 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
(logfile = STDOUT,logfile) List of log files
(mode    = ql          )
```

5.2.5 Image combining and sky subtraction

These two tasks are placed in the same section as the author sometimes combines the images first and then removes the sky and sometimes does the sky subtraciton first and then combines the images.

Let's talk about the first method.

Here ones combines the 2d data with a task like `imcombine`. The advantage of this technique is that it is quick and works for most observations. But there are disadvantages as well. Here is a listing of the most important parmaeters.

```
PACKAGE  = immatch
TASK     = imcombine
```

First the input list and the output filename.

```
input    = @input.list List of images to combine
output   = output      List of output images
```

Then a useful parameter which, if set, results in cube of rejected pixels.

```
(rejmask =             ) List of rejection masks (optional)
```

Then the choice of how to average and reject pixels. Since the gain and readnoise are known, use the `crreject` task.

```
(combine = average     ) Type of combine operation
(reject  = crreject    ) Type of rejection
```

Then an offset list. This is not necessary of you have not offet object along the slit.

```
(offsets = offsets.list) Input image offsets
```

Frames should be scale. This is one of the weakneses of this technique as you will soon see.

```
(scale   = median      ) Image scaling
```

And, finally, the rejection parameters. Here I've used the ones for the master chip of FORS2.

```
(hsigma  = 12          ) Upper sigma clipping factor
(rdnoise = 2.9         ) ccdclip:  CCD readout noise (electrons)
(gain    = 0.7         ) ccdclip:  CCD gain (electrons/DN)
```

Once the data have been combined, the spectrum of the object can be extracted using the `apall` task, which is described later. In many situations, this is a perfectly adequate way of treating the data. However, their are limitations. The IRAF `imcombine` routine has been designed with images in mind, so the scaling is done on the entire image. In spectra, the night sky lines vary with respect to eachother. So it may happen that some cosmic rays are missed or the night sky lines themselves are wrongly selected as cosmic rays if the relative variation is large. The latter problem is exaserbated if slit curvature is strongly non-linear or if thier is significant flexure. Thankfully, flexure in FORS is small, so the latter effect is not a significant problem.

The solution is to subtract the sky first and to then combine the data. This can be done in two ways. The first way is to do a 1 dimensional sky subtraction on the two dimensional spectra resulting in a 2d spectrum without sky. At the same time as one subtracts the sky one finds and excludes cosmic rays. In IRAF, there is routine called background that can do this, but it has weaknesses. Firstly, it does not trace the spectrum and it does not follow the curvature of the sky lines when estimating the sky. The `apall` routine does the former but not the latter and results in a 1d spectrum and not a 2d spectrum. The weakness of this method is that it does not remove fringes.

The second way is to do a full 2d sky subtraction, as IR astronomers have been doing for years. The disadvantage of this method is that it increases the statistical noise. The increase depends on how many frames were used to create the sky frame. If there are n sky frames, the noise in the object frame increases by sqrt(1+1/n). For non-and-shuffle, n=2. For offseting the object along the slit, n can be as large as you like; however, for practical reasons, n is usually between 5 and 10. The sky frames need not form a contiguous data set. The author has created sky frames from data that spans several days. Only the setup needs to stay constant. The BIG advantage of this method is that it removes fringes in the sky very, very well and it enables you to extract objects that are on slit edges - something that occurs frequently in MOS mode. At the same time as one removes the sky, one finds and excludes cosmic rays. For faint object spectroscopy at red wavelengths, the second way is the method of choice and it is what the author uses. None of this coded as an IRAF task, so the author wrote a routine in C to do it.

Note that this method only works if you dithered the object along the slit. For simplicity, let us consider a data set in which the night sky lines are perfectly aligned along detector columns. This is not true in real data, however, it simplifies the description of how to do the two dimentional sky subtraction. The procedure can be generalised to handle tilted sky lines.

Let us consider one column at time. Hence, a single frame simplifies to a one dimentional vector, the length of which is the length of the slit. If there were n frames with n unique offsets, then there will be $n-1$ frames for which we can compute the sky. For pixel, k, in frame i, the sky is then

$$\mathrm{Sky}_{ik} = \sum_{j \neq i} I_{jk}/(n-1)$$

where I_{jk} is the intesity of pixel k in frame j.

In total, there will be n skies, one for each frame. This sky is then subtracted from frame i.

$$\mathrm{Sub}_{\mathrm{ik}} = \mathrm{Frame}_{ik} - s_i * \mathrm{Sky}_{ik}/S_i$$

where s_i is the median of frame i and S_i is the median of sky i. Object and bad pixel masking can be easily incorporated into the procedure.

The end product of both methods is the same - a set of sky subtracted 2d spectra. One then combines the 2 spectra to produce a combined 2d sky subtracted spectrum. Since one has identified cosmic rays during the process of sky subtraction, one simply averages the 2d sky subtracted spectra with the appropriate pixel shifts between frames.

5.2.6 Object extraction

The author works with 2d data that are without the sky. The author also prefers to use a simple aperture rather than a weighted extraction based on the profile of the object and the noise properties of the array.

The extraction is done with the `apall` task. One defines the aperture, traces it (if it is bright enough), and extracts it without sky subtraction.

The key parameters in apall for this are:

Input and output

```
input    = skySub2dSpectra List of input images
(output  = spectrum.ms ) List of output spectra
```

Define and edit the apertures

```
(edit    = yes          ) Edit apertures?
```

Turn off background subtraction and variance weighting

```
(backgro = none         ) Background to subtract
(skybox  = 1            ) Box car smoothing length for sky
(weights = none         ) Extraction weights (none|variance)
(pfit    = fit1d        ) Profile fitting type (fit1d|fit2d)
(clean   = no           ) Detect and replace bad pixels?
```

The result is a 1d spectrum. The next steps are wavelength and flux calibration.

5.3 Wavelength Calibration

In most cases, one uses the arcs to determine the dispersion solution and the night sky lines to correct for the small offsets caused by flexure. If the resolution is not too low, one can calibrate on the night sky lines directly, although one should avoid strong blends when doing this. Osterbrock et al. (1996, 1997) have published a high resolution atlas of the night sky emission lines.

In general, one should extract the arc with the aperture that was used for extracting the science spectrum. In `apall`, this can be done by setting the references parameter

```
input    = skySub2dSpectra List of input images
(output  = spectrum.ms ) List of output spectra

(referen = skySub2dSpectra) List of aperture reference images
```

The relationship between pixels and wavelengths space (the dipsersion relation) is usually well described with a 4th order polynomial. A line list is given in the FORS2 manual. Note that you should avoid blends or closely separated lines. The residuals should be better than 1/10th of a pixel.

The dispersion solution is obtained with the IRAF `identify` task. Here is a listing of the most critical parameters.

```
images   = Arc.ms      Images containing features to be identified
(databas = database    ) Database in which to record feature data
(coordli = linelists$idhenear.dat) User coordinate list
(fwidth  = 4.          ) Feature width in pixels
(cradius = 5.          ) Centering radius in pixels
(thresho = 0.          ) Feature threshold for centering
(minsep  = 2.          ) Minimum pixel separation
(functio = legendre    ) Coordinate function
(order   = 4           ) Order of coordinate function
```

Once the dispersion relation is obtained, the next step is to apply it to the extracted science spectrum. In IRAF, this is done with two tasks. The first task (`refspectra`) is used to associate a dispersion solution to a science frame. The second task (`dispcor`) does the actual calibration. These tasks can be used to calibrate multiple apertures simultaneously, so some care in the parameter settings are required. As with other IRAF tasks, it is imprortant to read the help file.

```
PACKAGE  = onedspec
TASK     = refspectra

input    = spectrum.ms List of input spectra
(referen = arc.ms      ) List of reference spectra
(apertur =             ) Input aperture selection list
(refaps  =             ) Reference aperture selection list
(ignorea = yes         ) Ignore input and reference apertures?
(select  = interp      ) Selection method for reference spectra
(sort    =             ) Sort key
(group   =             ) Group key
(time    = no          ) Is sort key a time?
(timewra = 17.         ) Time wrap point for time sorting
(overrid = yes         ) Override previous assignments?
(confirm = yes         ) Confirm reference spectrum assignments?
(assign  = yes         ) Assign the reference spectra to the input spectr
(logfile = STDOUT,logfile) List of logfiles
(verbose = yes         ) Verbose log output?
answer   = yes         Accept assignment?
```

```
PACKAGE = onedspec
TASK = dispcor

input    = spectrum.ms List of input spectra
output   = spectrum.wc List of output spectra
(lineari = yes         ) Linearize (interpolate) spectra?
(databas = database    ) Dispersion solution database
```

```
(table   =              ) Wavelength table for apertures
(w1      = INDEF        ) Starting wavelength
(w2      = INDEF        ) Ending wavelength
(dw      = INDEF        ) Wavelength interval per pixel
(nw      = INDEF        ) Number of output pixels
(log     = no           ) Logarithmic wavelength scale?
(flux    = yes          ) Conserve flux?
(blank   = 0.           ) Output value of points not in input
(samedis = yes          ) Same dispersion in all apertures?
(global  = no           ) Apply global defaults?
(ignorea = yes          ) Ignore apertures?
(confirm = no           ) Confirm dispersion coordinates?
(listonl = no           ) List the dispersion coordinates only?
(verbose = yes          ) Print linear dispersion assignments?
```

At this point, one can choose to interpolate to a linear wavelength scale and to "conserve flux". Note that the choice of interpolation is set from the `onedspec` package parameter.

```
PACKAGE = noao
TASK = onedspec

(observa = observatory ) Observatory for data
(caldir  =             ) Standard star calibration directory
(interp  = poly5       ) Interpolation type
(dispaxi = 1           ) Image axis for 2D/3D images
(nsum    = 1           ) Number of lines/columns to sum for 2D/3D images
(records =             ) Record number extensions
```

5.4 Flux Calibration

Flux calibration is usually done by observing a star that has a well measured SED. Unless you actually submitted an OB to observe a star with the long slit, these spectrophotometric stars are usually observed with the MOS mode, which have their own flats and arcs.

The steps to producing a wavelength calibrated standard star spectrum are identical to the steps that are used to produce a wavelength calibrated science spectrum, with the exception that it not necessary to use the sophiticated sky subtraction techniques described above. One dimensional sky subtraction with the `apall` task is sufficient.

After the spectrum of the star has been extracted and wavelength callibrated, the next step is to create a file (called the sensitivity file) that tabulates the counts measured in the standard star spectrum with published values. In IRAF, this is done with the `standard` task in the `onedspec` package. In the example that follows, the sensitivity file is called standard.sens.

The treatment of extinction in these tasks is not simple, so you should read the IRAF help files carefully. A good cross check is to flux and extinction correct the standard. You should be able to recover the tabulated fluxes.

```
PACKAGE  = onedspec
TASK     = standard

input    = standard.ec Input image file root name
output   = standard.sens Output flux file (used by SENSFUNC)
(samesta = yes          ) Same star in all apertures?
(beam_sw = no           ) Beam switch spectra?
(apertur =              ) Aperture selection list
(bandwid = INDEF        ) Bandpass widths
(bandsep = INDEF        ) Bandpass separation
(fnuzero = 3.68000E-20 ) Absolute flux zero point
(extinct = extinction.dat) Extinction file
(caldir  = onedstds$ctionewcal/) Directory containing calibration data
(observa = paranal      ) Observatory for data
(interac = yes          ) Graphic interaction to define new bandpasses
(graphic = stdgraph     ) Graphics output device
(cursor  =              ) Graphics cursor input
star_nam = 14816        Star name in calibration list
airmass  = 1.15         Airmass
exptime  = 30.          Exposure time (seconds)
mag      =              Magnitude of star
magband  =              Magnitude type
teff     =              Effective temperature or spectral type
answer   = yes          (no|yes|NO|YES|NO!|YES!)
```

Don't forget the trailing slash in the caldir parameter.

A useful feature of the task is that it allows one to reject points. The author usually rejects points that have little flux and points that overlap with the strong telluric A and B bands, and the strong telluric feature that lies at 9300 Angstroms.

The extinction for Paranal has been published by Patat et al (2004) in the ESO Messenger. It is similar to the one that is used at CTIO.

In most cases, only one standard would have been observed, so the sensitivity file will contain just one star; however, one can add additional stars to the sensitivity file, which can then be used to determine the extinction function as well as the sensitivity function in the next task.

The `senfunc` task in the onedspec package takes the sensitivity file and fits the sensitivity function. Generally, a 11th order polynomial will give a satisfactory fit to the sensitivey function; however, one often sees wiggles which are generated by deviant points. These points should be deleted and the fit redone.

```
PACKAGE  = onedspec
TASK     = sensfunc

standard = standard_sens Input standard star data file (from STANDARD)
sensitiv = sens         Output root sensitivity function imagename
(apertur =              ) Aperture selection list
```

```
(ignorea = yes         ) Ignore apertures and make one sensitivity functio
(logfile = logfile     ) Output log for statistics information
(extinct = extinction.dat) Extinction file
(newexti =             ) Output revised extinction file
(observa = paranal     ) Observatory of data
(functio = legendre    ) Fitting function
(order   = 11          ) Order of fit
(interac = yes         ) 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)
```

Once the senistivity funciton has been measured, one uses the `calibrate` task to calibrate the wavelength calibrated spectrum.

```
PACKAGE  = onedspec
TASK     = calibrate

input    = spectrum.wc spectra to calibrate
output   = spectrum.fc Output calibrated spectra
(extinct = yes         ) Apply extinction correction?
(flux    = yes         ) Apply flux calibration?
(extinct = extinction.dat) Extinction file
(observa = paranal     ) Observatory of observation
(ignorea = yes         ) Ignore aperture numbers in flux calibration?
(sensiti = sens        ) Image root name for sensitivity spectra
(fnu     = no          ) Create spectra having units of FNU?
airmass  = 1.1         Airmass
exptime  = 900.        Exposure time (seconds)
```

For most people, including the author of these lines, this is good enough. However, there are additional steps that one might consider. For example, the above steps do not take slit losses into the acccount, so the spectrum has not been calibrated to an absolute scale. If the wavelength range is very large, then one might have differential slit losses, which means that the relative wavelength calibrtion may not be good either. And finally, some people like to remove telluric features, such as the A and B bands, the strong feature at 9300 Angstroms and the weaker features at 7200, 8200 and 9000 Angstroms.

6 HIgh Time resolution (HIT) modes

Written by Kieran O'Brien, Paranal Science Operations

6.1 Introduction

The HIT modes are specific to FORS2 and for this reason it is important that users understand some of the basic principles of operation before they plow on ahead and begin to reduce data. Usually this experience comes when the user plans the observations and perhaps even takes the data themselves. However, this is not always the case, especially in the case of archival data analysis. So, with this in mind I will explain a few of the basics and how these affect the steps needed to calibrate the data.

The general principle of the HIT mode is to increase the duty cycle of imaging and spectroscopic observations by reducing the exposed region of the CCD and shifting the charge from this small region to an unexposed 'storage' region of the CCD. By shifting the charge in this manner a number of times, it is possible to store a time series of images or spectra on the CCD without needing to read-out the CCD after each exposure. The shutter remains open throughout the observation and is only closed once a predefined number of shifts have occurred. The CCD is then read-out using the standard low noise read-out mode used for all spectroscopic observations.

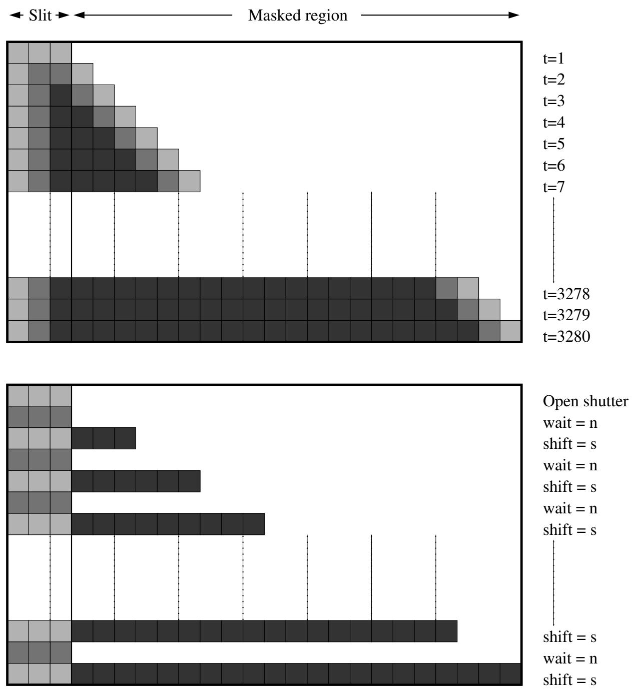


Figure 8: The basic operation of the different HIT mode clock patterns. The top panel shows the clock pattern for the HIT-I and HIT-OS patterns, where each row is shifted individually until the first row shifted reaches the end of the unvignetted region. The bottom panel shows the clock pattern for the HIT-MS mode, where a number of rows are grouped together and shifted out from the exposed region into the masked region where they wait for a time while the pixels under the slit are exposed to light from the source. 's' refers to the size in unbinned pixels of the slit, whilst 'n' is the number of seconds to wait between shifts.

The basic scheme of the HIT-I clock pattern is shown in the top panel of Figure 8. The imaging mode is available with five variations on this clock pattern. In each the charge is shifted across the unvignetted region of the chip (3280 unbinned pixels) in the direction away from the serial register, but the speed at which the charge is moved is varied to simulate a longer or shorter exposure time. The transfer from t=1-3280 takes place in either 1, 4, 16, 64 or 256-seconds, which identifies the clock pattern (ie HIT-OS1-1sec ... HIT-OS5-256sec). This means each row is shifted every 0.61, 2.4, 9.8, 39.0 or 156 milliseconds. The time resolution of the resulting lightcurve depends on the slit width, as each pixel is exposed as long as it remains in the unmasked region of the CCD. As can be seen in Figure 8 there are also some edge effects if the image of the slit is projected onto more than one pixel.

The HIT-MS mode allows users to operate the CCD with a "shift-and-wait" clock pattern, as shown in the bottom panel of Figure 8. In contrast to the HIT-I and HIT-OS modes, where the charge is constantly being shifted from one row to the next and only the rate at which it is shifted is changed, the HIT-MS clock pattern shifts a pre-defined number of rows very fast (∼2.5 microseconds per line) and then integrates for a pre-defined 'wait' time before the sequence is repeated. This continues until the maximum number of line-shifts has occurred and the CCD is read-out using the standard low read-out noise spectroscopic mode (100kHz,2x2,high).

6.2 general comments

The first thing to realise with HIT mode data is that there is often a lot of it, well actually there is always a LOT of it. For the HIT-MS mode, even if you choose a moderate 1-second per spectrum, this leads to ∼ 20,000 spectra per night. It is clear from this that you have to automate the data reduction as much as possible. This means writing scripts. My chosen format is PERL, as I find it gives me the right degree of flexibility whilst remaining simple to understand.

6.3 High time resolution imaging (HIT-I)

6.4 High time resolution spectroscopy (HIT-MS)

The Multiple-Shift (MS) submode of the HIgh-Time resolution (HIT) mode does not behave like another other mode of FORS and perhaps like any other mode anywhere. However, users will be relieved to know that, apart from a few peculiarities, it can be treated like a simple MOS observation. The peculiarity is that instead of the different spectra on a MOS image representing the spectra of different targets, in the HIT-MS mode they are multiple spectra of the same source (or perhaps sources). In addition, the field is rotated by 90degrees as it is necessary to have the dispersion parallel to the serial register. I work on each chip separately and then combine the 1-D spectra.

6.4.1 Preparation

For data reduction I use the STARLINK software freely available on-line. In addition to this, I use a number of the routines available in PAMELA and MOLLY to work on the 2-D and 1-D spectra respectively. The reason I chose these packages above more 'mainstream' ones is that I have a lot of confidence in the error propagation and they are designed to work in a batch

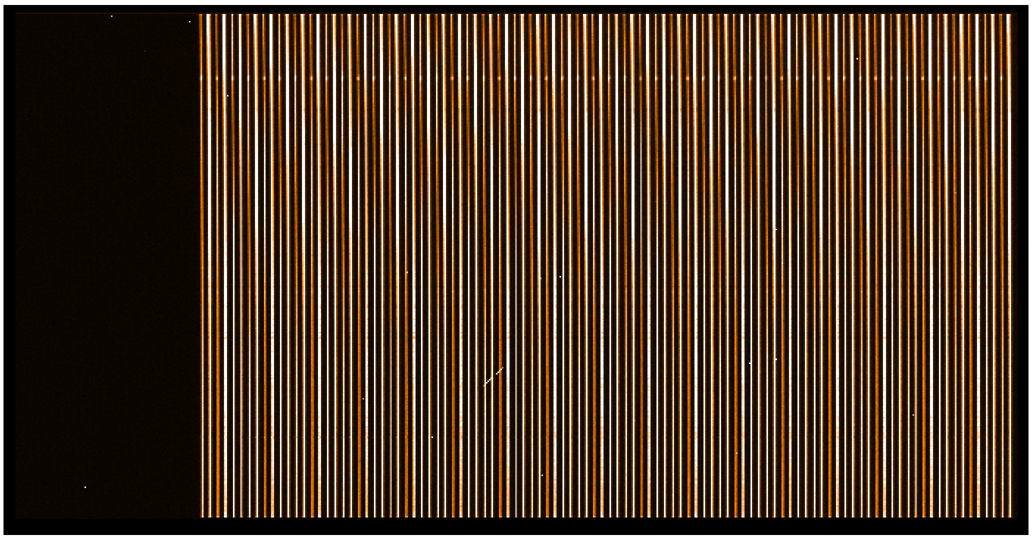


Figure 9: An example data-frame taken from chip1 of the data-set used as an example. The pairs of spectra can be clearly seen. The target spectra are the left-hand spectra, whilst the brighter comparison star spectra are on the right. Time runs from right to left, as the spectra on the extreme right are the ones that have undergone the most shifts and are therefore the first ones taken.

way. In addition, the source code is freely available, easy to follow and sufficiently flexible to allow one to create their own sub-routines and integrate them.

The first stage of the reduction is to convert the files into NDF files that can be understood by the STARLINK software. For this you should use the fits2ndf task of the convert package. However, the filenames used by ESO contain fullstops, which confuses the software, as it uses them to mark the beginning of its header structure. The easiest way to avoid this problem is to rename the files as soon as you copy them off the DVD. Once this is done, you can convert the files

```
COMMAND: fits2ndf
IN       - Input FITS file(s) /@F2hit_0.fits/ >
OUT      - Output NDF data structure(s) /@F2hit_0/ >
```

you should see a lot of files with the extension *.sdf*. Now we can begin to analyse the data, the first stage as always is to remove the bias level from the frames. In order to do this, I use the underscan region present on all FORS2 frames. A suitable command to do this in `PAMELA` is `PICSTAT`.

```
COMMAND: picstat
IMAGE    - Input data file /@F2hit_0/ >
STREG    - Statistics region input file /'statreg'/ >
CLIP     - Sigma rejection threshold /5/ >
PLOT     - Do you want histogram plot ?  /TRUE/ >
RANGE    - Sigma range for plot /5/ >
NBIN     - Number of bins /100/ >
```

where *stat_file* is a file containing the region (or regions) from which to build the statistic. An example of the output of this is

```
Number of pixels = 189281
Minimum = 172 at X,Y: 164, 837
Maximum = 3669 at X,Y: 142, 66
Straight mean value = 190.942
Straight standard deviation = 12.5663
7 pixels rejected at 5 sigma.
Revised mean value = 190.892
Revised standard deviation = 4.49602
Robust standard deviation = 4.51291
Median value = 191
```

I use the mean value after sigma-clipping, in this case 190.9 ADU. I then remove this level from the frame using the `FIGARO` command `ICSUB`.

```
COMMAND: icsub
IMAGE    - (IMage) Name of image to be subtracted from /@F2hit_0/ >
FACTOR   - (FACtor) Constant to be subtracted from image /190.9/ >
OUTPUT   - (OUTput) Name of resulting image /@db_F2hit_0/ >
```

The final stage in preparing the data-frames is to edit the headers of the files so that the exposure time for an individual dwell (in this example 2.5 seconds) is parsed to the final spectra, rather than the total integration time for exposure. This could equally be corrected later when other parameters are set, but I find it easier to make the correction at this stage.

```
COMMAND: fitset
FILE     - (FIle) Name of file in which FITS keyword is to be
      set /@db_F2hit_0/ >
KEYWORD  - (KEYword) FITS keyword to set or modify /'EXPTIME'/ >
VALUE    - (VALue) New value of FITS keyword /'2.5'/ >
COMMENT  - (COMment) Comment associated with FITS keyword /'Total
      Integration time'/>
```

6.4.2 Tracing the spectra

Now it is necessary to find the spectra in order to trace the spectral curvature and final extract the spectrum. In order to do this, I collapse the spectrum over a small number of pixels (so as not to smear the resulting profile too much if it is very curved/tilted) in the dispersion direction and fit this profile to find the centroid. I use the combination of `MEDPRF` in `PAMELA` and `FITGAUSS` in `FIGARO` to do this

```
COMAND: medprf
IMAGE    - Data frame /@db_F2hit_0/ >
FLAT     - Balance frame /@flat/ >
TRACE    - Load a trace of spectrum distortion ?  /FALSE/ >
PROFILE  - Output profile /@db_F2hit_0_gauss/ >
XSTART   - Lower X limit /2003/ >
```

```
XEND     - Upper X limit /2023/ >
YSTART   - Lower Y limit /1/ >
YEND     - Upper Y limit /1034/ >
NWIDTH   - Median filter width /3/ >
METHOD   - L(inear), Q(uadratic) or S(inc) rebinning?  /'Q'/ >
```

where FLAT is the flatfield image and the XSTART, XEND, YSTART, YEND parameters define the region of interest. The output of this is a file (*prof.sdf*) containing the spatial profile.

At this point I want to point out that I start extracting spectra at the right-hand side of the image, as this is the first spectrum taken (i.e. the one that has undergone the most shifts). This makes it a bit easier later on, as the spectrum numbers increase with time, but this is only my preferred way of doing it.

Secondly, I fit the profile with a Gaussian, in this case non-interactively,

```
COMMAND: fitgauss
IN       - Input NDF /@db_F2hit_0_gauss/ >
DEVICE   - Graphics device /!/ >
MASK1    - Mask interval lower bound(s) /[2002.5]/ >
MASK2    - Mask interval upper bound(s) /[2022.5]/ >
NCOMP    - Number of components /1/ >
CONT     - Continuum level /0/ >
CENTRE   - Gauss positions /2012.5/ >
PEAK     - Gauss heights /800/ >
FWHM     - Gauss widths (FWHM) /4/ >
CF       - Fit flags for line centres /0/ >
PF       - Fit flags for line peaks /0/ >
WF       - Fit flags for line widths /0/ >
COMP     - Component numbers for storage /1/ >
LOGFIL   - ASCII file name for result output /gauss_fit/ >
```

where DEVICE sets the output device name (use ! for no output), MASK1 and MASK2 set the lower and upper spatial limits of the fit respectively, CONT the continuum level, which is almost certainly negligible in such short exposure times and CENTRE, PEAK and FWHM set the initial guesses for the fit parameters. You should play around with these in interactive mode (DIALOG=T) a bit before you begin cranking through lots of spectra, as they fitting routine can easily get lost and give bad results. You can always check these results afterwards to look for outliers/problems.

Now you can use this information to trace the profile across the columns of the CCD. Tracing the spectrum is important for the high dispersion VPH grisms, as they are often curved, especially when observing away from the field centre, as we are forced to do with the HIT mode in order to try and 'stack' as many spectra as possible between read-outs. For this I use the PAMELA command TRACK. Again, I run it interactively a few times to find good initial parameters before starting the perl-script.

```
COMMAND: track
IMAGE    - Data frame to process /@db_F2hit_0/ >
FLAT     - Balance frame /@flat/ >
```

```
OLD      - Do you want to update an old fit ?  /FALSE/ >
TRACK    - File for track coefficients /@tr_targ/ >
NDPOLY   - Order of polynomial fit /4/ >
XSTART   - Lower X limit /2003/ >
XEND     - Upper X limit /2023/ >
YSTART   - Lower Y limit /1/ >
YEND     - Upper Y limit /1034/ >
PICK     - Pick spectrum point automatically ?  /TRUE/ >
NOBJ     - Number of objects /1/ >
PLOT     - Do you want to plot ?  /TRUE/ >
AUTO     - Scale frame plot automatically?  /TRUE/ >
WIDTH    - Width of window /10/ >
ESIG     - FWHM of gaussian /3/ >
FWHM     - FWHM of profile /3/ >
READOUT  - Readout noise (RMS ADU) /4.2/ >
PHOTON   - Number of ADU/count /1.43/ >
CLIP     - Sigma rejection threshold /3/ >
NBLOCK   - Block size /1/ >
TCYCLE   - Number of tweak cycles /5/ >
CHANGE   - Maximum shift during a tweak /0.5/ >
```

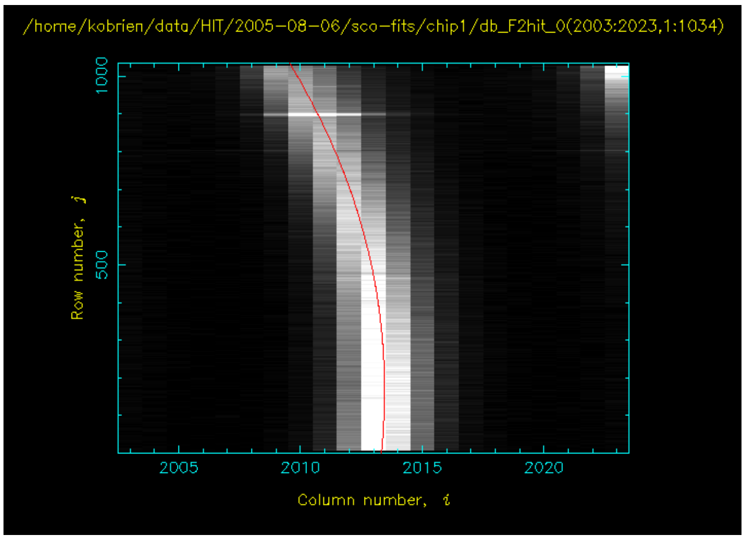


Figure 10: The output from TRACK. The 2-D spectral profile can clearly be seen. The red-line shows the 4th order polynomial fit to the track. The next spectrum can clearly be seen entering the frame in the top-right. This illustrates that it is essential to extract a 'tracked' spectrum to avoid contamination from nearby spectra.

6.4.3 Extracting the target and sky spectra

Now that we know where the spectrum is and how its position changes across the CCD, we need to define the different 'regions' of the spatial profile. These are the spectrum and sky regions that will be used in the subsequent extraction. In order to do this I use the REGPIC

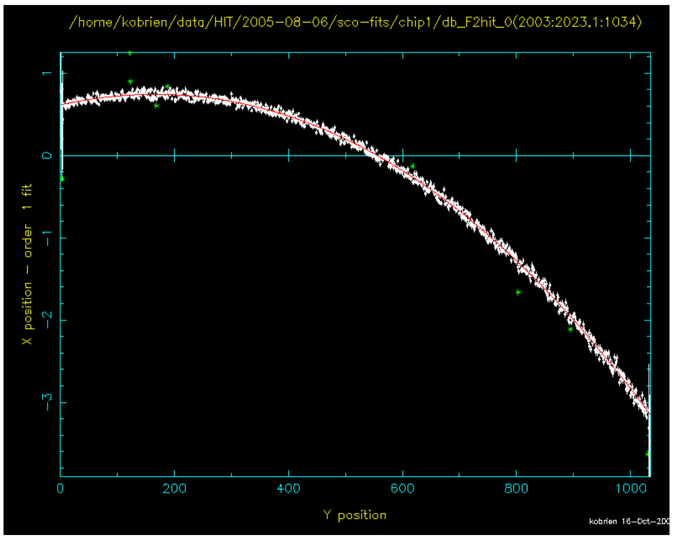


Figure 11: The resulting fit to the trace from `TRACK`. This clearly shows that there is a curvature of 4 pixels across the chip. This example is for the X600B grism and chip1.

command from `PAMELA`. As the slits are quite short (5"), It is difficult to determine sky regions that are sufficiently distant from the target to ensure you are not including flux from the target. Also, it is important to make sure that the sky region does not overlap with the sky region from the comparison star (if one is present), as these will have a different spectrum (and curvature) due to the offset in the dispersion direction (i.e. the central wavelength falls at a different position).

```
COMMAND: regpic
IMAGE    - Data frame /@db_F2hit_0/ >
FLAT     - Balance frame /@flat/ >
TRACE    - Load a trace of spectrum distortion ?  /TRUE/ >
TRACK    - Distortion map file /@tr_targ/ >
REGION   - Sky region output file /@targ/ >
GREY     - Do you want grey scale plot ?  /TRUE/ >
XSTART   - Lower X limit /2003/ >
XEND     - Upper X limit /2023/ >
YSTART   - Lower Y limit /1/ >
YEND     - Upper Y limit /1034/ >
AUTO     - Do you want automatic choice of scale ?  /TRUE/ >
METHOD   - Select regions with C(ursor) or T(erminal)?  /'T'/ >
```

Now that you have defined the regions to use as sky, you can extract a 2-dim sky model for the target region based on polynomial fits to pixel values in the sky regions. I use the `PAMELA` command `skyfit` to do this.

```
COMMAND: skyfit
IMAGE    - Data frame /@db_F2hit_0/ >
FLAT     - Balance frame /@flat/ >
```

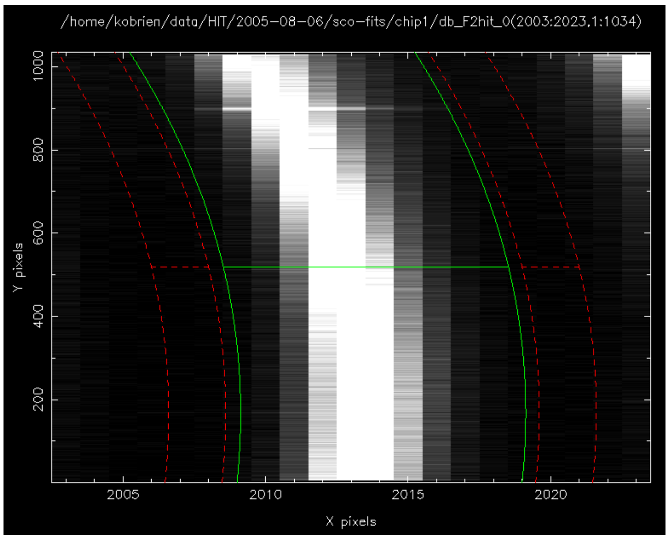


Figure 12: The regions selected to represent the sky (dashed red) and target (green) regions for the next stage of the extraction.

```
REGION   - Sky region input file /@targ/ >
TRACE    - Load a trace of spectrum distortion ?  /TRUE/ >
TRACK    - Distortion map file /@tr_targ/ >
SKY      - Sky fit output file /@sky_targ/ >
XSTART   - Lower X limit /2003/ >
XEND     - Upper X limit /2023/ >
YSTART   - Lower Y limit /1/ >
YEND     - Upper Y limit /1034/ >
NPOLY    - Number of terms for the sky fit /3/ > 2
THRESH   - Threshold for sky fit /5/ >
READOUT  - Readout noise (RMS ADU) /4.2/ >
PHOTON   - Detected photons/ADU /1.43/ >
```

It is important to use a low-order polynomial in this fit, as the spatial extent of the sky region is small and will not vary very much.

Now we have everything we need to perform the final stage, which is to extract the 1-D spectrum. The final important decision is how to extract the spectrum. The two choices are 'normal' or 'optimal' extraction, but you also need to decide whether or not to include the sky fit in the extraction or not. In my experience, the only way to decide these things is to try all the different methods and compare the results. The results depend on a number of factors, such as the brightness of the object and the atmospheric and sky conditions. For instance, unless your sky spectrum contributes a significant fraction to the flux in you extracted target spectrum, it is often best not to extract it, as you will only add noise to the extracted spectrum. Likewise, unless your target is faint, it is often not necessary to perform an optimal extraction. However, as I said before, the only way is to try all of the combinations and compare the results.

```
COMMAND: extnor
IMAGE    - Data frame /@db_F2hit_0/ >
```

```
FLAT     - Balance factor frame /@flat/ >
REGION   - Object region input file /@targ/ >
SKY      - Sky fit input file /@sky_targ/ >
TRACE    - Load a trace of spectrum distortion ?  /TRUE/ >
TRACK    - Distortion map file /@tr_test/ >
SPECT    - Spectrum output file /@targ_nor/ >
YSTART   - Lower Y limit /1/ >
YEND     - Upper Y limit /1034/ >
READOUT  - Readout noise, RMS data numbers /4.2/ >
PHOTON   - Photons/data number /1.43/ >
```

In the case of optimal extraction, in order to determine the weights for the contributions of each pixel to the final spectrum, it is necessary to run the `PAMELA` command `PROFIT`.

```
COMMAND: profit
IMAGE    - Data frame /@db_F2hit_0/ >
FLAT     - Balance factor frame /@unit/ >
REGION   - Object region input file /@targ/ >
SKY      - Sky fit input file /@sky_targ/ >
TRACK    - Distortion map file /@tr_targ/ >
FRACT    - Fraction output file /@targ_fract/ >
YSTART   - Lower Y limit /1/ >
YEND     - Upper Y limit /1034/ >
READOUT  - Readout noise, RMS data numbers /4.2/ >
PHOTON   - Photons/data number /1.43/ >
NPOLY    - Profile poly order /1/ > 5
SIZEX    - X separation of polynomials /0.5/ >
THRESH   - Rejection threshold /5/ >
NSLOW    - Maximum number of one-by-one rejections /10/ >
BADVAL   - Bad pixel limit (reject above) /65535/ >
NBLOCK   - Wavelength blocking factor /1/ >
NMED     - Width of median filter for sums /5/ >
PLOT     - Do you want to plot ?  /TRUE/ >
```

This fits polynomials to the flux levels of the columns covered by the target region defined by the `REGION` file. The polynomials are used to reject outliers caused by cosmics, as the tilt/curvature of the spectrum is assumed to vary smoothly along the columns. The output from this is a 2-D weight map that can is used in the next stage of the `PAMELA` optimal extraction process. The weight map is used by the command `OPTEXT` to extract the spectrum of the source. This routine is suitable for curved spectra and should not be confused with the annoyingly similar command `EXTOPT`, which is only suitable for straight spectra.

```
COMMAND: OPTEXT
IMAGE    - Data frame to process /@db_F2hit_0/ >
FLAT     - Balance factor frame /@flat/ >
REGION   - Object region input file /@targ/ >
SKY      - Sky fit input file /@sky_targ/ >
TRACK    - Distortion map file /@tr_targ/ >
```

```
FRACT    - Fraction file /@targ_fract/ >
SPECT    - Spectrum output file /@targ_optext/ >
YSTART   - Lower Y limit /1/ >
YEND     - Upper Y limit /1034/ >
READOUT  - Readout noise, RMS data numbers /4.2/ >
PHOTON   - Photons/data number /1.43/ >
ZAPRATS  - Do you want to reject cosmic rays ?  /TRUE/ >
RATLO    - Cosmic rat lower rejection threshold /-5/ >
RATHI    - Cosmic rat upper rejection threshold /5/ >
EPS      - Fudge factor to prevent bad rejections /0.01/ >
PLOT     - Do you want to plot ?  /TRUE/ >
IAVE     - Number of sky rows to average /1/ >
```

It is important to check the types of events that the code is rejecting, so for the first few times, I would recommend plotting the rejected pixels. This might tell you that the cosmic rejection threshold needs to be increased a bit. Once you are happy with the results of the extraction, you need to go back and do everything again for the comparison star. This is necessary, as several of the parameters will be quite different for the comparison star due to the different wavelength coverage compared to the science targets. If you were not able to use a comparison star within a few arcseconds of the target, then this difference could be quite large.

6.4.4 fixing the header items

The next step is to update the header items to correct the MJD and integration time. The following command takes the MJD-OBS keyword from the header and puts it into a file named *head.file*.

`hdstrace` *filename*`.more nlines=a newline eachline | grep MJD-OBS > head.file`

The represents the MJD at the beginning of the exposure, so it is simple to add an integer number of sub-integration times to this number to give you the MJD of each sub integration. It is then necessary to put this back into the header of the extracted spectrum. It is also necessary to include the sub-integration time as the exposure time in a further header item for consistency. I use the `FIGARO` command `FITSET` to do this. For instance,

```
COMMAND: FITSET
FILE     - (FIle) Name of file /@db_F2hit_0/ >
KEYWORD  - (KEYword) FITS keyword to set or modify /'EXPTIME'/ >
VALUE    - (VALue) New value of FITS keyword /'2.5'/ >
COMMENT  - (COMment) Comment associated with keyword /'Integration Time'/ >
```

In order to ensure that all of the header items are in the correct place, with the correct names and values, it is necessary to run the perl script (`fithead.pl`) kindly provided by Tom Marsh as part of PAMELA.

Finally, once you have the parameters for the target and comparison star, it is best to start the automated process. I use the outputs from the previous extraction as the inputs to the next which assumes that things change smoothly combined with the strict periodicity of the clock pattern to help in determining the centroids of the next target+comparison pair.

6.4.5 extracting the arcs

Now that you have extracted the science frames, it is necessary to extract the products needed to calibrate the data. The first of these are the arcs, which in the case of FORS, means the arcs taken during the daytime. In the case of the XGRIS_600B, the He and HgCd lamps are used, whereas for the XGRIS_300I, the He and Ar lamps are used. In order to minimize the effects of the tilt of the spectral trace, I use the trace created from the target and the comparison star frames to extract the arc spectra for each of the sub-integrations on the chip. It is not necessary to use a flatfield for this stage and the sky spectrum should be a zero frame to avoid it misidentifying the arc lines as sky lines. Normal (as opposed to optimal) extraction is best for this process, as the S/N is very high.

6.4.6 extracting the flux standard

In order to properly flux calibrate the science frames it is necessary to have two things:

- an observation of a spectro-photometric standard

  This is used to calculate the instrumental response for the night. It should be taken with a slitwidth wide enough to ensure that none of the light is lost

- a 'wide-slit' observation of the comparison star used in the science observations

  This should be taken with the same wide slit as used for the spectro-photometric standard. It is used to measure the losses caused by having the star over-fill the slit. I prefer to use the HIT-MS mode to take the data, as it allows me to be confident that nothing else has changed in the set-up. This might change with time, but it seems to work well this way.

The spectra from these frames should be extracted in the same manner as the science, including the arc frames taken during the daytime. Now that you have all of the pieces, you can use your favourite piece of software to wavelength and flux calibrate your data. I use MOLLY for this, the steps for which can be found on the webpage for the software (`http://deneb.astro.warwick.ac.uk/phsaap/software/`).

A Reducing FORS data using the pipeline recipes

A.1 Overview

Since May 21, 2007, the FORS Spectroscopic Pipeline recipes can be downloaded from `http://www.eso.org/pipelines`. The pipeline is currently based on only two recipes, *fors_calib* for reducing calibration frames and producing an extraction mask, and *fors_science* for applying the *fors_calib* products to the reduction of scientific exposures. The recipes can be applied to both FORS1 and FORS2 spectroscopic data obtained in the LSS, MOS, and MXU observation modes. Development work is still ongoing, as these recipes do not yet support important tasks such as the combined reduction of jittered scientific exposures, or the determination of the instrument response curves. Also the reduction of polarimetric and imaging data is not supported yet.

A.2 Launching pipeline recipes: Gasgano and Esorex

Pipeline recipes can be launched using either of two host applications, *Gasgano* and *Esorex*. A description about how to use them is given in the FORS Pipeline User's Manual, that can be downloaded from `http://www.eso.org/pipelines`. It is not the purpose of this Cookbook to give a complete description of their usage, but rather to concentrate on data reduction issues. Very briefly, *Gasgano* is a graphic interface that, besides launching recipes, also helps to classify and associate the data to be processed; *Esorex*, on the other hand, is a command line utility for just launching pipeline recipes: data classification and association must be provided by the user, on the basis of the information attached to the data (*e.g.*, FITS header keywords). The advantage of *Esorex* is that it may be embedded into scripts for the automation of data reduction tasks.

Each pipeline recipe will process a set of input FITS data files that has been opportunely selected and classified. To classify an input file means assigning to it an indentification tag, the so-called Data Organiser (DO) category. For instance, a master bias frame is identified by the tag MASTER_BIAS, and a raw flat field lamp exposure obtained in MOS mode is identified by the tag SCREEN_FLAT_MOS. When using *Esorex* the names of the input files must be listed, together with their classification tags, in an ASCII file that is conventionally called *set-of-frames* (SOF). The SOF corresponds to the *Input Frames* panel of the *Gasgano* Recipe Execution window.

Here is an example of SOF, valid for the *fors_calib* recipe:

```
../../cal/FORS1_MBIAS.fits         MASTER_BIAS
FORS1.2006-05-10T12:58:27.122.fits SCREEN_FLAT_MOS
FORS1.2006-05-10T12:59:45.326.fits SCREEN_FLAT_MOS
FORS1.2006-05-10T13:00:20.930.fits SCREEN_FLAT_MOS
FORS1.2006-05-10T13:01:17.711.fits SCREEN_FLAT_MOS
FORS1.2006-05-10T13:02:14.559.fits SCREEN_FLAT_MOS
FORS1.2006-05-10T13:03:37.926.fits LAMP_MOS
FORS1_ACAT_300I_11_OG590_72.fits   MASTER_LINECAT
```

Each input frame is identified by its path and name, followed by the appropriate classification tag. *Gasgano* provides the classification tags automatically.

In the case of files created by a pipeline recipe, the classification tag is identical to the content of the header keyword ESO.PRO.CATG, while for raw files the classification tag is generally based on the content of the keywords ESO.DPR.CATG, ESO.DPR.TYPE, and ESO.DPR.TECH (see the FORS Pipeline User's Manual for more details on the data files classification).

Note that the FORS pipeline recipes do not verify in any way the correctness of the classification tags specified in the SOF: correct tagging must be supplied by external applications, such as *Gasgano* or a script provided by the user. A recipe handling an incorrectly tagged file may stop and display unclear error messages at best. In the worst cases, the recipe would apparently run successfully, producing results that may look reasonable, but are actually flawed.

A complete list of the recipes configuration parameters and of their meaning can be found in the FORS Pipeline User's Manual. It is also displayed in the *Gasgano* Recipe Execution window, and can be generated by *Esorex*: for instance, the command

```
  esorex -help fors_calib
```

would generate a list of the available parameters for recipe *fors_calib*, with their default and a brief explanation of their meaning.

___oOo___