INTEGRAL Data Reduction Manual

Doc. Reference:	INT-IAC-MNL-004
Doc. Internal Code:	-
Issue:	draft 1
Rev.:	-
Date:	17-2-1998
Author:	Carlos del Burgo Díaz

Instituto de Astrofísica de Canarias

0. Control section

0.1. Approval Page

Prepared by:

Carlos del Burgo Díaz, *IAC Science Processing*

Authorized by:

0.2. Document Change Record

Issue	Rev.	Sec.	Page	Date	Changes
draft 1	-	-	All	17-2-98	-

0.2. Distribution List

Recipient	Project Responsibility	Other	Institute	No. of copies
S. Arribas	*Principal Investigator*		IAC	1
E. Mediavilla	*IAC Project Scientist*			1
N. García *(Documentation)*	*Project Secretary*			1
C. del Burgo	*Science Processing*			1

1. Introduction

The *integral* package allows to INTEGRAL user to perform the reduction of the data obtained with this instrument in the IRAF environment. This package contains *standard* tasks of other spectra reduction packages (*onedspec* and *specred* and *ccdred*), and *special* tasks what have been developed to perform different operations with the INTEGRAL data. In particular, the INTEGRAL user dispose of several *integral* tasks (e.g.: *int_rec*) for the on-line reduction of the images.

Some of these tasks contain FORTRAN programs, which has been implemented inside the IRAF environment with the IMFORT Interface (see *A User's Guide to Fortran Programming in IRAF. The IMFORT Interface.* Doug Tody. NOAO. September 1986).

This manual includes: i) the *integral* tasks for a quick data examination and on-line data reduction during observing, and ii) the steps (and tasks) to perform the INTEGRAL Data Reduction.

2. How to access to *integral* package

The *integral* package can be installed inside IRAF environment. For the installation see *Ref:* INT-IAC-MNL-003.

For load in IRAF, type *cl* (and **[Return]**) from the directory (e.g. /package/integral) where you have the file/s login.cl and loginuser.cl (see *Ref:* INT-IAC-MNL-003):

/package/integral> cl

You can load the *integral* package typing *integral* like whichever IRAF package:

cl> integral

For get a general description of the *integral* package type [1]:

cl> help integral opt=sys helpdb="(int_helpdb)"

If you want obtain a paper copy of this help you have to type:

cl> help integral opt=sys helpdb="(int_helpdb)" | lprint

2.1. Listing and brief help on tasks of *integral* package

When you load *integral*, all *integral special tasks* (specific tasks developed for INTEGRAL) are showed on the display. The *integral standard tasks* (like *apall*) aren't showed, but are loaded.

If the user type:

cl> help integral helpdb="(int_helpdb)"

then a list with the *integral special tasks* and a brief description of each one of them is printed on display.

The *integral* tasks are summarized on the Table I.

[1] if *integral* package has been installed in LOCAL ("IRAF Site Mananger's Guide"), you have not to type helpdb "(int_helpdb)"

Table I

Task	**Function**
int_hart1	Hartmann test for a couple of images
int_hart	Hartmann test for a list of couples of images
int_fwhm1	FWHM of a emission lines array for an 2D image
int_fwhm	FWHM of a emission lines array for a list of 2D images
int_stat	Statistics data for a list of 2D images
ms2ascii	Converts a multispec format 2D image spectra to an ASCII text spectra
readfxcor	Export to a file the values of radial velocities or widths or pixels of the task fxcor
int_map	2D reconstruction from a ASCII file with fiber number, x, y, parameter to be mapped, and fibre status columns
int_rec	2D reconstruction from a non-reduced 2D image
int_spd1f	Spatial shift between two images for a fiber
int_spdXf	Spatial shift between two images for a fiber list
int_nspec	Calibrated and extracted spectrum of a fiber/aperture
int_xyfib	Shows on display the spatial coordinate of a fiber
int_daref	r.a. and dec. shifts produced by differential atmospheric refraction

2.2. Description and help on *integral* tasks

Help is available for each *integral special task*:

cl> help int_hart helpdb="(int_helpdb)"

Likewise, you can see help of whichever *standard integral task* typing *help* followed by the name of the task, e.g.:

cl> help apall

3. Quick data examination

Now, you are in the directory where you have installed *integral* package (/package/integral/, according to our example). You should have to change to the directory where you have the images to reduce [2]. You have to do:

cl> cd /work/images/

In order to prepare the files required to the images reconstruction run int_status. This task requires the file **databundles.txt** in /work/images/database, according to our example. If this subdirectory is not already in existence, you have to create it:

cd> mkdir database

And copy the files databundles.txt, SB#.data and SB#.status (where #: 1,2,3) from /package/images/database/databundles.txt (which was created during the *integral* package installation):

cl> cp /package/images/database/databundles.txt database

cl> cp /package/images/database/SB.* database

Now, you can run *int_status*:

cl> int_status

In order to use int_daref, we recommed copy the file example.ref, which must be in /work/images/database (according to our example):

cl> cp /package/images/database/example.ref database

For a raw analysis of the images *integral standard tasks*, like *imexamine* or *implot* can be used. With *imexamine* the user can generate profiles along the columns or rows (type j and k respectively), contours (type e) etc.

More complex operations (e.g. extraction of a fibre/aperture, building up a map, etc.) require the knowledge of *integral special tasks*.

These are the *special task* of *integral* package:

3.1. int_hart

Function: Hartmann test for a list of couples of images

Syntax: int_hart H_file P_file dispaxis outfile1 outfile2

[2] If you are observing on the WHT, you have to change to the subdirectory corresponding where you are receiving the images

Parameters:

- H_file: The file with the focus, HRimages and HLimages names.
- P_file: File with the coordinates (X,Y) of emission features array.
- dispaxis (= 0): Dispersion axis (0: horizontal axis, 1: vertical axis).
- outfile1: Output with the outfiles names containing the offsets.
- outfile2: Output file with the values of best focus.

Description: This task allow us determine the shifts between the centers along the dispersion axis of several emission features (emission lines array of an arc image) for several images couples. Each couple correspond to an image obtained with the right and the left shutters open and close respectively (HRimage), and the other one with the right and the left shutters close and open respectively (HLimage). These features correspond to isolated arc lines over all field of the detector. A value of the best focus in the positions of these emission lines is obtained too.

The H_file is a file which contains in each row the following columns: focus, HRimage and HLimage:

-10000.0	HR1.imh	HL1.imh
-6000.0	HR2.imh	HL2.imh
-2000.0	HR3.imh	HL3.imh
2000.0	HR4.imh	HL4.imh
4000.0	HR5.imh	HL5.imh
6000.0	HR6.imh	HL6.imh
10000.0	HR7.imh	HL7.imh

The P_file is a file of two columns containing the aproximate (with a preciseness of around 1 pixel) coordinates X and Y over the detector of the emission features which correspond to the arc image. An example of P_file:

20	20
520	25
1002	22
23	527
542	532
985	520
24	1003
541	1012
1003	1007

The dispaxis parameter (0: horizontal axis, 1: vertical axis) corresponds to the dispersion axis on the CCD. And the outfile1 is an output file which contains the files names with the values of the shifts between the centres (obtained using the *imcntr* task) of the emission lines of the HRimage and the HLimage for all emission lines whose approximate coordinates (X,Y) are in P_file.

The outfile2 is the other output file containing the values of the best focus in each point (X,Y) of the P_file file. The outfile2 is typed on display. The best focus in each point is obtained from the offsets (in this point) for the different focus positions. These offsets are fitted to a straight line. The best focus position correspond with the value for which the offset is zero.

3.2. int_fwhm

Function: FWHM of a emission lines array for a list of 2D images

Syntax: int_fwhm A_file P_file dispaxis nx ny outfile

Parameters:

- A_file: File with the focus and the names of the ARC images.
- P_file: File with the coordinates (X,Y) of several emission features.
- dispaxis (= 0): Dispersion axis (0: horizontal axis, 1: vertical axis).
- nx number of plots in x direction.
- ny number of plots in y direction.
- outfile: Output file with the names of the output files.

Description: Firstly, you must to have opened a graphics display (e.g. Saoimage).

With this task we obtain for a list of arc images the FWHM (from a gaussian fit) for several emission features. For each one of the arc image, a mean value of the FWHM and the standard desviation is obtained from the different FWHM values. The A_file file contains two columns (focus, arc image) in each row:

```
-3000.0   I1.imh
-2000.0   I2.imh
-1000.0   I3.imh
0.0       I4.imh
1000.0    I5.imh
2000.0    I6.imh
3000.0    I7.imh
```

We have to select features as isolated as possible in order obtain the centers (with the *imcntr* task) of these ones with a good precision (around a 1 pixel).

P_file is a input file of two columns containing the approximate (around 1 pixel) coordinates X and Y of the emission features, which are distributed over all detector. An example of P_file:

20	20
520	25
1002	22
23	527
542	532
985	520
24	1003
541	1012
1003	1007

The dispaxis parameter (0: horizontal axis, 1: vertical axis) corresponds to the dispersion axis on the CCD.

The outfile is the file with the different output files names. These outfiles contain the FWHM values of the gaussian fits for the different emission features whose coordinates are in P_file.

On graphics display, int_fwhm shows the selected emission features: these emission features are marked with *tvmark*. And the values corresponding to the centers of the emission lines are determined from gaussian fits done with imexamine. Likewise, int_fwhm show the fits with *gkimosaic* task. A mean value of the FWHM and the standard desviation for each one of the arc image is showed too.

3.3. int_stat

Function: Statistical data for a list of 2D images

Syntax: int_stat imagelist region outfile

Parameters:

- imagelist: The file with the images names to determine the statistical parameters
- region (= "[3:17,10:1000]"): The region of the image to do the statistics
- outfile: The output image file with the statistic data

Description: This task determines the statistical parameters (mean, stsdev, min, max) of a section of all images includes in the imageslist file. The output file shows this statistical data and the Universal Time and the name of the image. If the name of the output file already exists, the task restores the old file by its name followed by 0. So, you can compare these two files.

The name of the images list (imagelist, with a image name per line) must be preceded by the @ symbol.

The syntax for defining the region is: "[x1:x2,y1:y2]", where x1 and x2 are the first and the end pixels in the x-axis and y1 and y2 correspond to the y-axis coordinates. This region is showed on display.

3.4. ms2ascii

Function: Converts a multispec format 2D image spectra to an ASCII text spectra.

Syntax: ms2ascii msimage asciiname

Parameters:

- msimage: Name (without extension ".ms") of the input 2D image in the multispec format.
- asciiname: Root name of the output files with the ASCII text spectra.
- invert (=no): Root name of the output files with the ASCII text spectra.

Description: With this task is possible convert (see help of *wspectext* task) a multispec format 2D image to ASCII text spectra.

All output files (ASCII text spectra) are called by the root name followed by _ symbol and the number corresponding to the fiber. This number must correspond to the aperture. If you have extracted the apertures with the parameter order=decreasing in the case of bundles SB1 and SB3, you have to put invert = yes. And invert = no in the case of SB2 (for which the parameter order=increasing).

3.5. readfxcor

Function: Exports to a file the values of radial velocities, FWHM or pixel shifts of *fxcor* task.

Syntax: readfxcor fxcorfile rvfile pattern bundle

Parameters:

- fxcorfile: Text file generated by fxcor
- rvfile: Output file with the values of velocities
- pattern: Parameter to extract (velocities (vel); FWHM (fwhm) or shifts in pixels (pixel))
- bundle: Bundle: SB1 (1); SB2 (2) or SB3 (3)

Description:

With this task we can extract the velocities, FWHM or shifts in pixels from the ".txt" suffix text output file, which contains these parameters in the standard IRAF 'list' format.

The output file has the same structure of an input file of *int_map* task (see next section): nf, x, y, parameter and status, where nf is the number of the fibre, x and y are the x-coordinate and the y-coordinate of the fibre at the focal plane of the telescope, parameter is the extracted value (vel/fwhm/pixel) corresponding to the fibre number nf and status defines if the fibre is broken (0) or non-broken (1).

3.6. int_map

Function: 2D reconstruction from a ASCII file with fiber number, x, y, parameter to be mapped, and fibre status columns

Syntax: int_map inputfile interp bundle invert outimage

Parameters:

- inputfile: The name of the file to interpolate.
- interp (= 1): Interpolation: linear(1) o logarithmic(0).
- bundle: Bundle: SB1(0), SB2(1), SB3(2).
- invert (= no): Change the entry values for the interpolation of a fiber by its complementary one.
- outimage: Name of the output image (reconstructed image).

Description: With this task we can obtain the reconstruction of an image from the values of the different optical fiber array of a bundle.

The inputfile is a input file, which contains five columns: nf x y parameter status, where nf is the number of the fibre, x and y are the x-coordinate and the y-coordinate of the fibre at the focal plane of the telescope, parameter to interpolate corresponding to the nf fibre and status defines if the fibre is broken (0) or non-broken (1).

The interpolation may be linear (interp=1) or logarithmic (interp=0).

For each bundle there is a grid. For the SB1 bundle the reconstructed image (outimage) have 135 by 118 pixels with 0.05 arcsec per pixel. For the SB2 bundle the outimage have 153 by 111 pixels with 0.1 arcsec per pixel. And for the SB3 bundle the outimage have 97 by 185 pixels with 0.3 arcsec per pixel.

If the order of the fibers along the slit of the spectrograph is changed respect to the input values of the fibers, you can put invert = yes in order to change the entry values for the interpolation of each one of the fiber by its complementary one fiber number + complementary fiber number = fiber total number). If you have conveniently extracted the apertures, invert = yes in the case of the bundles SB1 and SB3 (parameter order=decreasing), and invert = no in the case of SB2 (parameter order=increasing).

3.7. int_rec

Function: 2D reconstruction from a non-reduced 2D image.

Syntax: int_rec object flat pix1 pix2 bundle bias outimage device

Parameters:

- object: Object image
- flat: Flat image
- pix1: First pixel along the dispersion axis
- pix2: End pixel along the dispersion axis
- bundle: SB1 (1), SB2 (2) and SB3 (3)
- bias: bias level
- outimage: Outimage
- device: Output device (stdgraph, eps, stdplot)

Description: This task allows us perform a 2D reconstruction. Using this task an intensity map between pix1 and pix2 pixels in the dispersion axis is obtained. For do it, *int_rec* requires to define, trace and extract the apertures of the object and correct the integrated values between pix1 and pix2 of the spectra corresponding to those apertures of the transmission of the fibres. A flat image is necessary for this correction and, optionally, for defining, tracing and extraction of the object image (this is convenient for faint objects).

Firstly, *int_rec* defines and traces the apertures (using *apall* task) of the flat image. So, a file with the parameters required to extracting is created in the *database* directory. The syntax of this file is 'ap' followed by the name of the image (in this case, apflatp1_p2). If this file is already in existence, the operations will not be repeated.

If the apflatp1_p2 no exists, you have to define the apertures interactively. In principle, *int_rec* marks all the apertures according to the bundle (SB1, SB2 or SB3). A or more intensive cosmic rays (which can be presented along the spatial profile of the fibres taken like reference for defining the apertures) and be marked instead of the fibres (fainter than the cosmic ray/s). In this case, the user has to delete (typing *d*) the aperture corresponding to the cosmic rays and mark (typing *m*) near the center of the aperture corresponding to the faint fibre. The broken fibres (in the case of SB1 and SB2 bundles) have to be marked (typing *n*) too. Probably you have to order the apertures. For do it, type *a* for select all the apertures and type *o* for order them. Help is available on these commands by typing *?* (see help of *apall* for more details). You have not change the aperture widths because these are conveniently selected by *int_rec*.

After defining of the aperture positions and widths, they are spectrally traced. In principle, the user can do it operation interactively (for some or all the fibres). Although, the tracing parameters (like the order of the polynomial) are conveniently selected and probably you don't need change them.

Once done the defining and tracing of the apertures, these will be extracted.

Obviously, the apobjectp1_p2 file wasn't already in existence. So, you need define, trace and extract the apertures of the object. Like the previous operations have been carried out on the flat field image with the same instrument configuration (same central wavelength, grating and bundle), it is safe to assume that the same extraction parameters will apply. The INTEGRAL/WYFFOS system is very stable, and therefore you can use (like good approximation) the same extraction (we principally refer to tracing parameters) parameters for object than those for the flat. This will be unavoidable in the case of very faint objects.

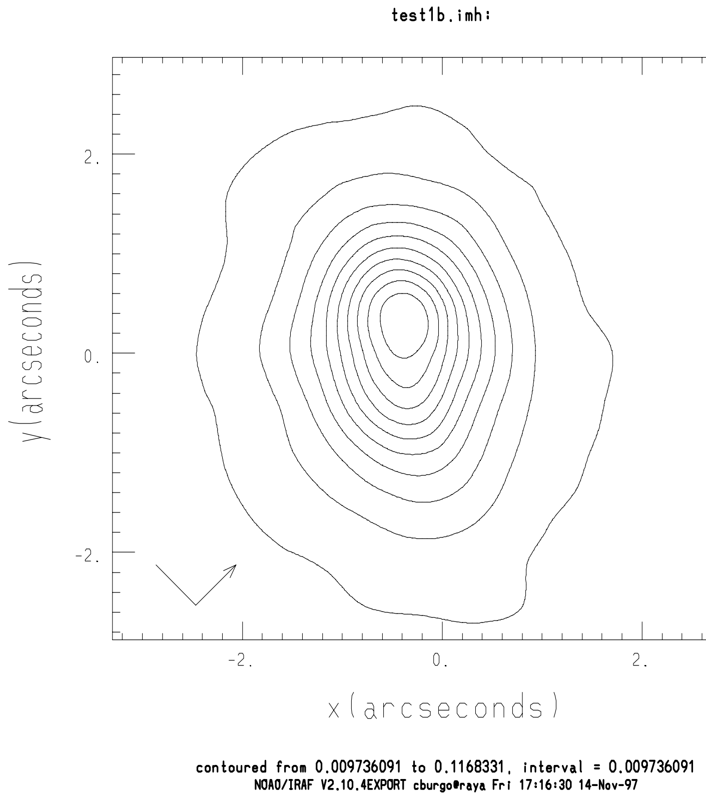

Figure 1. Postscript copy of M31 generated by int_rec.

After extraction of the object, a reconstruction of the image (according to the NAG routines *e01saf* and *e01sbf*) will be done.

The reconstruction of the image will be usually printed on display (stdgraph). But you can use other output devices: stdplot or eps (to get a postscript copy).

Note that an estimation of the bias level is required to *int_rec*.

Example:

If we are observing M31 and get a image of this object ("m31object") and a flat ("refflat") with the SB1 bundle, and we are interested in get a postscript copy of an intensity map between pixels 300 and 500 in the dispersion axis, we type *epar int_rec* and put:

PACKAGE=integral
TASK=int_rec

object	=	"m31object"	Object image
flat	=	"refflat"	Flat image
pix1	=	300	1st pixel along the dispersion axis
pix2	=	500	2nd pixel in the dispersion axis
bundle	=	1	SB1(1), SB2(2), SB3(3)
bias	=	1000	BIAS
outimage	=	"test1"	Outimage
(device	=	eps)	Output device (stdgraph, eps, stdplot)
(mode	=	"q")	

The output image (test1) is showed in the Figure 1.

3.8. int_spd1f

Function: Spatial shift between two images for a fiber

Syntax: int_spd1f image1 image2 fiber outfile outimage

Parameters:

- image1: First image name
- image2: Second image name
- fiber: Number of fiber
- outfile: Output file
- outimage: Output image

Description: This task allows us to obtain the spatial shift of a fiber (whose number is indicated with fiber parameter) between two images (FLAT SPECTRAs, generally). These images (image1 and image2) has been previously traced in order to get the fiber centers (which are saved at the *database*).

At the output file (outfile), the fiber spatial shift is saved. And an image (outimage) with the differences between polynomial parameters for do the spectra extraction is created too.

3.9. int_spdXf

Function: Spatial shift between two images for a fiber list

Syntax: int_spdXf image1 image2 fib_file outfile

Parameters:

- image1: First image name
- image2: Second image name
- fib_file: List with the fibers numbers (one per column)
- outfile: Output file with the data

Description: This task allows us to obtain the spatial shift of a fiber list (whose numbers are indicated with fiber parameter) between two images (FLAT SPECTRAs, generally). These images (image1 and image2) has been previously traced in order to get the fiber centers (whose are saved at the *database*).

At the output file (outfile), the spatial shifts of each one of fiber list are saved.

3.10. int_nspec

Function: Calibrated and extracted spectrum of a fiber.

Syntax: int_nspec image, reference, fibre_n, bundle, bias, spectrum

Parameters:

- image: Object image
- reference: Flat image
- fibre_n: Number of fiber
- spectrum: Output image of the n-spectrum
- bundle: SB1 (1), SB2 (2), SB3 (3)
- bias: bias level
- calibration (=yes)

Description: This task allows us obtain the calibrated (or non-calibrated) spectrum of the a selected fibre.

Firstly, *int_nspec* defines and traces the apertures (using *apall* task) of the flat image (reference image). So, a file with the parameters required to extracting is created in the *database* directory. The syntax of this file is 'ap' followed by the name of the image (in this case, apflat). If this file is already in existence, the operations will not be repeated.

If the apflat no exists, you have to define the apertures interactively. In principle, *int_rec* marks all the apertures according to the bundle (SB1, SB2 or SB3). One or more intensive cosmic rays (which can be presented along the spatial profile of the fibres taken like reference for defining the apertures) and be marked instead of the fibres (fainter than the cosmic ray/s). In this case, the user has to delete (typing *d*) the aperture corresponding to the cosmic rays and mark (typing *m*) near the center of the aperture corresponding to the faint fibre. The broken fibres (in the case of SB1 and SB2 bundles) have to be marked (typing *n*) too. Probably you have to order the apertures. For do it, type *a* for select all the apertures and type *o* for order them. Help is available on these commands by typing *?* (see help of *apall* for more details). You have not to change the aperture widths because these are conveniently selected by *int_rec*.

After defining of the aperture positions and widths, they are spectrally traced. In principle, the user can do it operation interactively (for some or all the fibres). Although, the tracing parameters (like the order of the polynomial) are conveniently selected and probably you don't need change them.

Once done the defining and tracing of the apertures, these will be extracted.

Obviously, the apobject file wasn't already in existence. So, you need define, trace and extract the apertures of the object. Like the previous operations have been carried out on the flat field image with the same instrument configuration (same central wavelength, grating and bundle), it is safe to assume that the same extraction parameters will apply. The INTEGRAL/WYFFOS system is very stable, and therefore you can use (like good approximation) the same extraction parameters (we principally refer to tracing parameters) for object than those for the flat. This will be unavoidable (and commendable) in the case of very faint objects.

Finally, a calibrated (or non-calibrated) spectrum of the selected fibre is showed on display. The calibration is done according to the object header parameters. Therefore, the calibration is approximative.

Note that an estimation of the bias level is required to *int_nspec*.

3.11. int_xyfib

Function: Shows on display the spatial coordinate of a fiber

Syntax: int_xyfib image fiber

Parameters:

- image: The image name
- fiber: The fiber number

Description: With this task xcenter (fixed for all fiber when you define the apertures) and ycenter (spatial coordinate) parameters of a selected fiber are found at the apertures *database* (*database*/apimage) and shows on the display. Of course, you had previously traced (with e.g. *apall*) the fibers/apertures.

3.12. int_daref

Function: r.a. and dec. shifts produced by differential atmospheric refraction

Syntax: int_daref infile outfile

Parameters:

- infile: Input file with the data
- outfile: Output file

Description: With this task we can calculate right ascension and declination shifts produced by differential atmospheric refraction.

An example of input file (example.ref):

```
*************** Temperature ********************************
15.9
************** Pressure ************************************
575.0
************** Object **************************************
test
************** Number of images ****************************
2
************** Number of wavelength ************************
3
************** Wavelength in microns ***********************
0.4000
0.7000
0.8000
** Object right ascension in hours,minutes y seconds**************
0.0
10.0
20.0
** Object declination in grades, minutes and arc seconds**********
30.0
40.0
50.0
*** La Palma latitude in grades, minutes and arc seconds*********
28.0
45.0
34.0
******* TSLstart(h,m,s) and texposed (seconds)*****************
01.0 09.0 08.69 1800.0
02.0 53.0 22.30 1800.0
****************** Output files****************************
ref1.test
ref2.test
****************** Output file*****************************
refdata.out
```

You can edit and change the temperature and pressure (corresponding to the observing night), the name of the object, the wavelength/s (and the number of wavelength/s), the r.a. and dec. of the object, the TSLstart and texposed (and the number of images), and the names (in this case, ref1.test and ref2.test) of the output files, which contains r.a.*cos(dec.) and dec. shifts for the object. The output file (refdata.out, according to our example) contains the above parameters and other parameters like the paralactic angle, zenith distance (z), Local Sidereal Time (LST) and hour angle (h).

Editing and running the *integral special tasks*

Like a wichever IRAF task, these tasks can be edited with *epar*, e.g.:

cl> epar int_hart

Then you can edit/change the parameters of the task. Once done this, you can go out (typing *:q*) or run the task (typing *:go*). For run the task you can type its name too, e.g.:

cl> int_hart

Using *integral* tasks

Previous observing, you can be interested in determine the WYFFOS focus (see *Ref.:* INT-IAC-TRP-009 for details). In order to do it, you can use **int_hart**. And you can obtain arc lines FWHM with **int_fwhm** for the best focused image.

During observing you can use **int_rec** to perform a reconstruction image of your object. And get some of their spectra with **int_nspec**. These are very useful in order to verify the observed field, the characteristic of this field, the emission and absorption features, or the data quality (signal to noise).

Likewise, with **int_daref** you can determine the different wavelenght continuum maxima shifts produced by the differencial atmospheric refraction.

And can verify the stability of the system with **int_spd1f** and/or **int_spdXf**.

4. INTEGRAL Data Reduction

4.1. General Considerations

Before INTEGRAL Data Reduction (see next section) you have to prepare the files required to reconstruction of the images. In order to do it, run *int_status* (see **chapter 3**).

Description of the files required and created by int_status

The *int_status* (includes in *integral* package) allows us to obtain the coordinates files required to the reconstruction of the images. For do it, you have to run *int_status* inside IRAF (see **chapter 3**).

The files required for int_status are:

- The file databundles.txt, which basically contains the names of the input files required for int_status.
- The files SB1.status, SB2.status and SB3.status (one for each bundle). These files contain two columns: #fibre and FS, where FS is the current fibre status (broken: 0, no broken:1).
- The files SB1.data, SB2.data and SB3.data. These files contain four columns: #fibre, x(arcsec), y(arcsec) and fibre type. This parameter defines object fibres (distributed over the rectangular central region of the bundle) and sky fibres (distributed over the external ring of the bundle).

You must no change the files with 'data' extension, but you have to know the current status of each fibre. In principle, the support astronomer should edit the files with extension 'status' in order to change the fibre status (non-broken ⇒ broken) if this was necessary.

The files generated by int_status are:

- SB1.coor, SB2.coor and SB3.coor. These files contain four columns: #fibre (number of fibre), x(arcsec), y(arcsec) and fibre status. The parameters x and y are the fibre coordinates in arcsec at the focal plane, fibre status defines the current status of each fibre (broken: 0, non-broken:1), and #fibre is the fibre number. The number of rows of these files corresponds to the number of fibres of each bundle which are object type (those of the rectangular central region of each bundle). These files are necessary for perform a image reconstruction.
- SB1.pix, SB2.pix and SB3.pix. These files are very useful in order to display with *tvmark* task some of the INTEGRAL bundles (circumferences with center at the fibre position and the fibre radius for each one of the observing fibres) over an image.

4.2. Step by step Data Reduction

The principal tasks used by *integral* package (*integral*) to perform INTEGRAL Data Reduction are in *specred* package. The INTEGRAL user can use this package to perform a lot of steps of the data reduction. But, in order to do: i) a first analysis of the INTEGRAL data quality during observing, ii) a on-line data reduction, and iii) 2D-image reconstruction, he would have to use the specific *integral* tasks (see **chapter 3**).

The steps required to the INTEGRAL Data Reduction are defined on the Table II and successively explained. At the Appendix I you can see the parameters that are recommended in the INTEGRAL data reduction. The user can use these or another (similar) parameters.

Table II

Step	Goal	Package	Task/s
1	Bias Subtraction	*ccdred*	*ccdproc*
2	Aperture Definition and Trace	*specred*	*apall*
3	Scattered Light Correction	*specred*	*apscatter*
4	Flat-spectra Correction	*specred*	*apflatten*
5	Extraction	*specred*	*apall*
6	Thoughtput Correction	*specred*	*msresp1d*
7	Arc Line Identification and Dispersion Correction	*specred*	*identify*, *reidentify* *refspectra*, *dispcor* *setjd*
8	Sky Subtraction	*specred*	*skysub*
9	Image combination	*specred*	*scombine*
7	Cosmic rays Rejection	*images*	*lineclean*
8	Image reconstruction	*integral*	*int_map*

Bias subtraction

The subtraction of the bias level is the first step in the data reduction. This bias level is introduced onto the CCD chip for ensuring that the chip is working in a linear regime.

You can subtract the bias level according to one of the following proceedings:

i) Get the mean value on the overscan region of each frame and subtract this constant off the whole rest of the frame. The task to see the profiles along the vertical and horizontal axis is *implot* (from the *plot* package) and determine the overscan region, Once done this, you can use *ccdproc* (load *noao*, *imred* and *ccdred*) for sustracting the bias level.

ii) Average a sample (around 10 or more) of bias frames (frames taken with zero exposure time) pixel by pixel in order to get a mean bias frame, and subtract this frame from each image. The task to do it is *zerocombine* (previously, from *noao*, *imred* and *ccdred* packages). Then, with *ccdproc* you can sustract the bias level.

In practice, it can be used the first way for subtracting bias level to INTEGRAL images (see Appendix I).

Aperture definition and trace

The apertures corresponding to each one of the fibres are distributed onto the CCD detector. We have to define and trace these apertures before than to extract them.

The apertures are commonly affected by the field distortions 0 by the optics ot the system and so each spectrum aren't line up along the rows/columns in the dispersion direction. The apertures will follow distorted paths which can be fitted by a polynomial.

For quite faint objects can be necessary a **reference image** (typically, a **flat image**, image taken illuminating the system with a uniform lamp -generally tungsten-), which parameters of the tracing polynomial are used to the **object image**. This is justifiable given the stability of the INTEGRAL/WYFFOS system.

Just before the scattered light subtraction, we define the apertures very narrow (around 0.5 pixels). After the scattered light subtraction, we redefine the apertures to the **standard widths** (2.8, 3.2 and 5.4 pixels for the SB1, SB2 and SB3 bundles).

This operation can be done with *apall* (see Appendix I) task (load *noao*, *imred* and *specred* packages).

Scattered light subtraction

WYFFOS has scattered light like whichever spectrograph. It is important to subtract the scattered light, principally in the case of faint objects. We recommend *apscatter* (load *noao*, *imred* and *specred* packages) to do this operation (see Appendix I). Firstly, if the definition and the tracing of the apertures was not done, this task allow us to do it. Secondly, a polynomial fit is done along the spatial direction: a polynomial fit is used to reproduce the enveloped of the intensity values from the spaces between fibres. Finally, a polynomial fit is done along the dispersion axis.

Flat-spectra Correction

This operation is done to correct of the pixel to pixel variations. A **flat image** with high signal to noise ratio is necessary. And the relative displacement between the apertures of this **flat image** and the **object image** have to be lower than 0.1 pixel.

Firstly, we redefine the apertures to the **standard widths** with *apall*.

Secondly, we use *apflatten* (see Appendix I) to create an image (hereafter, **flatresponse**) which is exactly equal to one outside the apertures width and around to one (depending on pixel to pixel relative sensibility variations) inside them.

Finally, we divide the **object image** (after bias subtraction and scattered light subtraction) by the **flatresponse**. This operation can be done with *imarith* (load *images* package).

Extraction

The extraction consists in the summing up of the spectral pixels along the spatial direction into a final onedimensional spectrum (one per aperture or optical fibre). We can do the extraction of the apertures with the **standard widths** and the previously determined tracing polynomial using *apall* (see Appendix I).

After the extraction, we have 2D-image of n×m dimensions, where n is the number of pixels along the dispersion axis and m the number of fibres.

Thoughtput Correction

The spectra of the **object images** have to be corrected of fibre thoughput. To do it, a **sky image** is used. In absence of a **sky image**, we can use a *flat image*.

This operation is done with the *msresp1d* task. A 2D output image (hereafter, **fibresens**) with a constant value per each fibre (corresponding to its sensibility; all the values are around the unity) is obtained (e.g.: msresp1d "" flatim "" fibresens).

Then, we divide the **object image** by the **fibresens** with the *imarith* task.

Arc Line Identification and Dispersion Correction

The identification of the emission lines of an image corresponding to a comparison lamp (hereafter, **arc image**) is done to perform the calibration in wavelength of the spectra.

Firstly, we use *identify* (load *noao*, *imred* and *specred* packages) to identify the emission lines (around 15 features) of the middle fibre of the **arc image**.

Secondly, we use *reidentify* (load *noao*, *imred* and *specred* packages) to identify the emission lines of the remaining fibres of the **arc image** taking the middle fibre like reference. So, we will have a **arc image** with the identification of the emission features for all fibres. We can use this **arc image** like reference to identify the emission lines of the whichever **arc image** using *reidentify*.

After that, we use *setjd* (load *noao*, *imred* and *specred* packages) to set geocentric Julian day, the heliocentric Julian day and the **ljd** parameter (which is a constant value for all images taking during the same observing night).

Then, we assign one or two reference **arc image** for each **object image** with *refspectra* (load *noao*, *imred* and *specred* packages).

And, finally, we can perform the dispersion correction using the *dispcor* task (load *noao*, *imred* and *specred* packages).

Image combination

If we have several **object images**, we can combine these images in order to improve the signal to noise ratio using *scombine* (load *noao*, *imred* and *specred* packages).

Cosmic rays Rejection

For the rejection of cosmic rays we recommend *lineclean* (load *images*).

Image reconstruction

The reconstruction of 2D images of whichever parameter can be done using *int_map* (from *integral* package).

Appendix I

Parameters of *ccdproc*:

images	=	“”	List of CCD images to correct
(ccdtype	=	“object”)	CCD image type to correct
(max_cache	=	0)	Maximum image caching memory (in Mbytes)
(noproc	=	no)	List processing steps only?
(fixpix	=	no)	Fix bad CCD lines and columns?
(overscan	=	yes)	Apply overscan strip correction?
(trim	=	yes)	Trim the image?
(zerocor	=	no)	Apply zero level correction?
(darkcor	=	no)	Apply dark count correction?
(flatcor	=	no)	Apply flat field correction?
(illumcor	=	no)	Apply illumination correction?
(fringecor	=	no)	Apply fringe correction?
(readcor	=	no)	Convert zero level image to readout correction?
(scancor	=	no)	Convert flat field image to scan correction?
(readaxis	=	“line”)	Read out axis (column—line)
(fixfile	=	“”)	File describing the bad lines and columns
(biassec	=	“[5:45,10:1100]”)	Overscan strip image section
(trimsec	=	“[49:1075,1:1024]”)	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
(minreplace	=	1.)	Minimum flat field value
(scantype	=	“shortscan”)	Scan type (shortscan—longscan)
(nscan	=	1)	Number of short scan lines
(interactive	=	no)	Fit overscan interactively?
(function	=	“legendre”)	Fitting function
(order	=	1)	Number of polynomial terms or spline pieces
(sample	=	“*”)	Sample points to fit
(naverage	=	1)	Number of sample points to combine
(niterate	=	1)	Number of rejection iterations
(low_reject	=	3.)	Low sigma rejection factor
(high_reject	=	3.)	High sigma rejection factor
(grow	=	0.)	Rejection growing radius
(mode	=	“ql”)	

Parameters of *apscatter*:

input	=	“”	List of input images to subtract scattered ligh
output	=	“”	List of output corrected images
(scatter	=	“”)	List of scattered light images (optional)
(references	=	“”)	List of aperture reference images
(interactive	=	yes)	Run task interactively?
(find	=	*see Note 1*)	Find apertures?
(recenter	=	no)	Recenter apertures?
(resize	=	no)	Resize apertures?
(edit	=	no)	Edit apertures?
(trace	=	no)	Trace apertures?
(fittrace	=	yes)	Fit the traced points interactively?
(subtract	=	yes)	Subtract scattered light?
(smooth	=	yes)	Smooth scattered light along the dispersion?
(fitscatter	=	yes)	Fit scattered light interactively?
(fitsmooth	=	yes)	Smooth the scattered light interactively?
(line	=	INDEF)	Dispersion line
(nsum	=	10)	Number of dispersion lines to sum or median
(buffer	=	1.)	Buffer distance from apertures
(apscat1	=	“”)	Fitting parameters across the dispersion
(apscat2	=	“”)	Fitting parameters along the dispersion
(mode	=	“ql”)	

Parameters of *apscat1*:

(function	=	”legendre”)	Fitting function
(order	=	*see Note 2*)	Order of fitting function
(sample	=	”*”)	Sample points to use in fit
(naverage	=	1)	Number of points in sample averaging
(low_reject	=	1.)	Low rejection in sigma of fit
(high_reject	=	0.75)	High rejection in sigma of fit
(niterate	=	5)	Number of rejection iterations
(grow	=	0.)	Rejection growing radius in pixels
(mode	=	”ql”)	

Parameters of *apscat2*:

(function	=	”legendre”)	Fitting function
(order	=	*see Note 3*)	Order of fitting function
(sample	=	”*”)	Sample points to use in fit
(naverage	=	1)	Number of points in sample averaging
(low_reject	=	3.)	Low rejection in sigma of fit
(high_reject	=	3.)	High rejection in sigma of fit
(niterate	=	1)	Number of rejection iterations
(grow	=	0.)	Rejection growing radius in pixels
(mode	=	”ql”)	

Parameters of *apflatten*:

input	=		List of images to flatten
output	=		List of output flatten images
(references	=	"")	List of reference images
(interactive	=	yes)	Run task interactively?
(find	=	yes)	Find apertures?
(recenter	=	yes)	Recenter apertures?
(resize	=	yes)	Resize apertures?
(edit	=	yes)	Edit apertures?
(trace	=	yes)	Trace apertures?
(fittrace	=	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
(threshold	=	10.)	Threshold for flattening spectra
(background	=	"none")	Background to subtract
(pfit	=	"fit1d")	Profile fitting type (fit1d—fit2d)
(clean	=	no)	Detect and replace bad pixels?
(skybox	=	1)	Box car smoothing length for sky
(saturation	=	INDEF)	Saturation level
(readnoise	=	"0.")	Read out noise sigma (photons)
(gain	=	"1.")	Photon gain (photons/data number)
(lsigma	=	4.)	Lower rejection threshold
(usigma	=	4.)	Upper rejection threshold
(function	=	"legendre")	Fitting function for normalization spectra
(order	=	1)	Fitting function order
(sample	=	"*")	Sample regions
(naverage	=	1)	Average or median
(niterate	=	0)	Number of rejection iterations
(low_reject	=	3.)	Lower rejection sigma
(high_reject	=	3.)	High upper rejection sigma
(grow	=	0.)	Rejection growing radius
(mode	=	"q")	

Edit *fit1d* and select order between 7-9.

Parameters of *apall*:

PACKAGE= specred
TASK=apall

input	=		List of input images
nfind	=	*see Table III*	Number of apertures to be found automatically
(output	=	"")	List of output spectra
(format	=	"multispec")	Extracted spectra format
(references	=	*see Note 1*	List of aperture reference images
(profiles	=	"")	List of aperture profile images
(interactive	=	yes)	Run task interactively?
(find	=	yes)	Find apertures?
(recenter	=	*see Note 1*)	Recenter apertures?
(resize	=	no)	Resize apertures?
(edit	=	yes)	Edit apertures?
(trace	=	yes)	Trace apertures?
(fittrace	=	yes)	Fit the traced points interactively?
(extract	=	*see Note 2*)	Extract spectra?
(extras	=	no)	Extract sky, sigma, etc.?
(review	=	yes)	Review extractions?
(line	=	INDEF)	Dispersion line
(nsum	=	10)	Number of dispersion lines to sum or median
(lower	=	*see Table III*)	Lower aperture limit relative to center
(upper	=	*see Table III*)	Upper aperture limit relative to center
(apidtable	=	"")	Aperture ID table (optional)

DEFAULT BACKGROUND

(b_function	=	"chebyshev")	Background function
(b_order	=	2)	Background function order
(b_sample	=	*see Table III*)	Background sample regions
(b_naverage	=	2)	Background average or median
(b_niterate	=	0)	Background rejection iterations
(b_low_reject	=	3.)	Background lower rejection sigma
(b_high_rejec	=	3.)	Background upper rejection sigma
(b_grow	=	0.)	Background rejection growing radius

APERTURES PARAMETERS

(width	=	*see Table III*)	Profile centering width
(radius	=	2.)	Profile centering radius
(threshold	=	0.)	Detection threshold for profile centering
(minsep	=	*see Table III*)	Minimum separation between spectra
(maxsep	=	*see Table III*)	Maximum separation between spectra
(order	=	*see Table III*)	Order of apertures

RECENTERING PARAMETERS

(apertures	=	“”)	Select apertures
(npeaks	=	(*see Table III*))	Select brightest peaks
(shift	=	yes)	Use average shift instead of recentering?

RECENTERING PARAMETERS

(llimit	=	*see Table III*)	Lower aperture limit relative to center
(ulimit	=	*see Table III*)	Upper aperture limit relative to center
(ylevel	=	0.1)	Fraction of peak or intensity for automatic wid
(peak	=	yes)	Is ylevel a fraction of the peak?
(bkg	=	yes)	Subtract background in automatic width?
(r_grow	=	0.)	Grow limits by this factor
(avglimits	=	yes)	Average limits over all apertures?

TRACING PARAMETERS

(t_nsum	=	*see Table III*)	Number of dispersion lines to sum
(t_step	=	*see Table III*)	Tracing step
(t_nlost	=	3)	Number of consecutive times profile is lost bef
(t_function	=	“legendre”)	Trace fitting function
(t_order	=	*see Table III*)	Trace fitting function order
(t_sample	=	“*”)	Trace sample regions
(t_naverage	=	1)	Trace average or median
(t_niterate	=	0)	Trace rejection iterations
(t_low_reject	=	3.)	Trace lower rejection sigma
(t_high_rejec	=	3.)	Trace upper rejection sigma
(t_grow	=	0.)	Trace rejection growing radius

EXTRACTION PARAMETERS

(background	=	*see Note 3*)	Background to subtract
(skybox	=	1)	Box car smoothing length for sky
(weights	=	*see Note 4*)	Extraction weights (none—variance)
(pfit	=	“fit1d”)	Profile fitting type (fit1d—fit2d)
(clean	=	no)	Detect and replace bad pixels?
(saturation	=	INDEF)	Saturation level
(readnoise	=	*see Note 5*)	Read out noise sigma (photons)
(gain	=	*see Note 6*)	Photon gain (photons/data number)
(lsigma	=	3.)	Lower rejection threshold
(usigma	=	3.)	Upper rejection threshold
(nsubaps	=	1)	Number of subapertures per aperture
(mode	=	“ql”)	

Table III

Parameter	Bundle	Reduction Step	Value/s
nfind	SB1	always	175
	SB2	always	189
	SB3	always	115
lower	SB1	for	-0.5
	SB2	scattered light	-0.5
	SB3	correction	-0.5
upper	SB1	for	0.5
	SB2	scattered light	0.5
	SB3	correction	0.5
lower	SB1	for	-1.8
	SB2	extraction	-2.0
	SB3		-3.0
upper	SB1	for	1.8
	SB2	extraction	2.0
	SB3		3.0
b_sample	SB1	always	"-4:-2,2:4"
	SB2	always	"-4:-1,1:4"
	SB3	always	"-5:-2,2:5"
width	SB1	always	4.0
	SB2	always	4.0
	SB3	always	6.0
minsep	SB1	always	3.0
	SB2	always	4.0
	SB3	always	6.5
maxsep	SB1	always	9.0
	SB2	always	5.0
	SB3	always	10.0
order	SB1	always	"decreasing"
	SB2	always	"increasing"
	SB3	always	"decreasing"
npeaks	SB1	always	174
	SB2	always	188
	SB3	always	115
llimit	SB1	always	-1.8
	SB2	always	-2.0
	SB3	always	-3.0
ulimit	SB1	always	1.8
	SB2	always	2.0
	SB3	always	3.0
t_nsum	SB1	always	20
	SB2	always	20
	SB3	always	15
t_step	SB1	always	10
	SB2	always	10
	SB3	always	15
t_order	SB1	for all the	7-15
	SB2	wavelenght	7-15
	SB3	range	7-15