Asiago Monografie

vol. 1

An introduction to analysis of single dispersion spectra with IRAF[1]

Tomaž Zwitter
Department of Physics, University of Ljubljana, Slovenia

Ulisse Munari
Padova and Asiago Astronomical Observatories, Italy

[1]IRAF (*Image Reduction and Analysis Facility*) is distributed by the National Optical Astronomy Observatories (USA), which are operated by the Association of Universities for Research in Astronomy Inc. (AURA), under cooperative agreement with the National Science Foundation.

Contents

1 Introduction

This is merely a transcript of a set of notes collected during our teaching to students in Padova and Ljubljana. By no means these notes can be taken as a systematic and/or exhaustive manual. The latter has already been written by Massey, Valdes & Barnes: "A User's Guide to Reducing Slit Spectra with IRAF". Your library or IRAF guru should have a copy of it. Check there for more infos and further references to literature. Another way to collect essential information on IRAF is to go to the IRAF Web site at

http://iraf.noao.edu/

where you find all the latest news, tutorials and manuals to download (included the Massey, Valdes & Barnes' one).

IRAF commands will be typed in these notes like *rfits @list0 * @list1*, while operating system commands will look like `ls –la *.fts`.

IRAF is designed not to depend on the kind of computer you are using (it is available under Unix, VMS and Linux operating systems, not Windows95/98/NT[2]). These notes are written with a user of a Linux–based system in mind (Linux is an Unix-type operating system originally developed for PCs, now ported to all major workstations). This should be fine for a vast majority of the astronomical community. For more detail on Linux, see

http://www.linux.org/

1.1 Demo spectra

These notes come with some real B&C spectra on which you can practice and follow step by step the procedures described here (we will refer directly to such real spectra when dealing with specific examples). The files have been secured in 1996 with the B&C+CCD spectrograph on the ESO 1.52 m telescope at La Silla (Chile) during observations of cataclysmic and symbiotic binaries. For information on the telescope and spectrograph see

http://www.ls.eso.org/lasilla/Telescopes/2p2T/E1p5M/E1p5M.html

The spectra can be downloaded from any of the following two sites:

http://ulisse.pd.astro.it/Astro/AsMon/Vol_1/

http://www.fiz.uni-lj.si/astro/learn.html

A listing of the files is given in Table 1. We start with some general remarks about IRAF. Next comes a description of general commands useful in evaluation and reduction of stellar spectra obtained with Boller & Chivens spectrograph (i.e. single order dispersion, long-slit modes). Then comes a description of specific procedures used in

[2]The only differences you will note will be connected to the way you call a function of the operating system directly from the IRAF prompt

spectra reduction. We conclude with a quick summary of the commands used to get flux and wavelength calibrated spectra.

2 Philosophy and practice of IRAF

Nobody knows everything, and this is particularly true for IRAF. So these pages have a modest goal: giving you enough confidence to try it yourself and possibly to save you from drowning in the depths of IRAF realms. But looking at some experienced swimmer is always useful. If any expert allows you to sit by him for a few hours and take notes while he/she is reducing data, you will learn the essential thing: IRAF may be big and difficult to learn in whole, but it has a very clear philosophy that is consistently implemented in all of its commands.

We start with a description of some basics. If you know some basic facts about IRAF already skip directly to Section 5. The present section assumes your X-windows and image display are already set-up. This is usually true in most classrooms. If you sit at home and wonder where to start IRAF from, move to Section 3 first and learn how to set up an user-friendly working environment.

IRAF is more than a software package just aimed to extract and calibrate astronomical CCD frames. Exploring it in some more detail (much more than really needed in these introductory notes), you will find that inside IRAF you have - for example - excellent plotting packages (no more need to export the files to ASCII tables and use SMONGO or WIP mathematics/graphics packages), a lot of tools for general astronomical computing, etc. For example under the package *noao.astutil* you have the tasks listed in Table 2.

2.1 Commands, packages and tasks

IRAF contains a large number of commands. For reasons of space and transparent use they are grouped together in chunks which are called packages. For example all commands dealing with manipulation of extracted slit spectra are in the package *onedspec* (the name comes from *one-d*imensional *spec*troscopy). The packages are further connected together in a tree-like structure. Only a limited number of commands is available when IRAF starts. Others can be enabled by typing the name of the package they belong to.

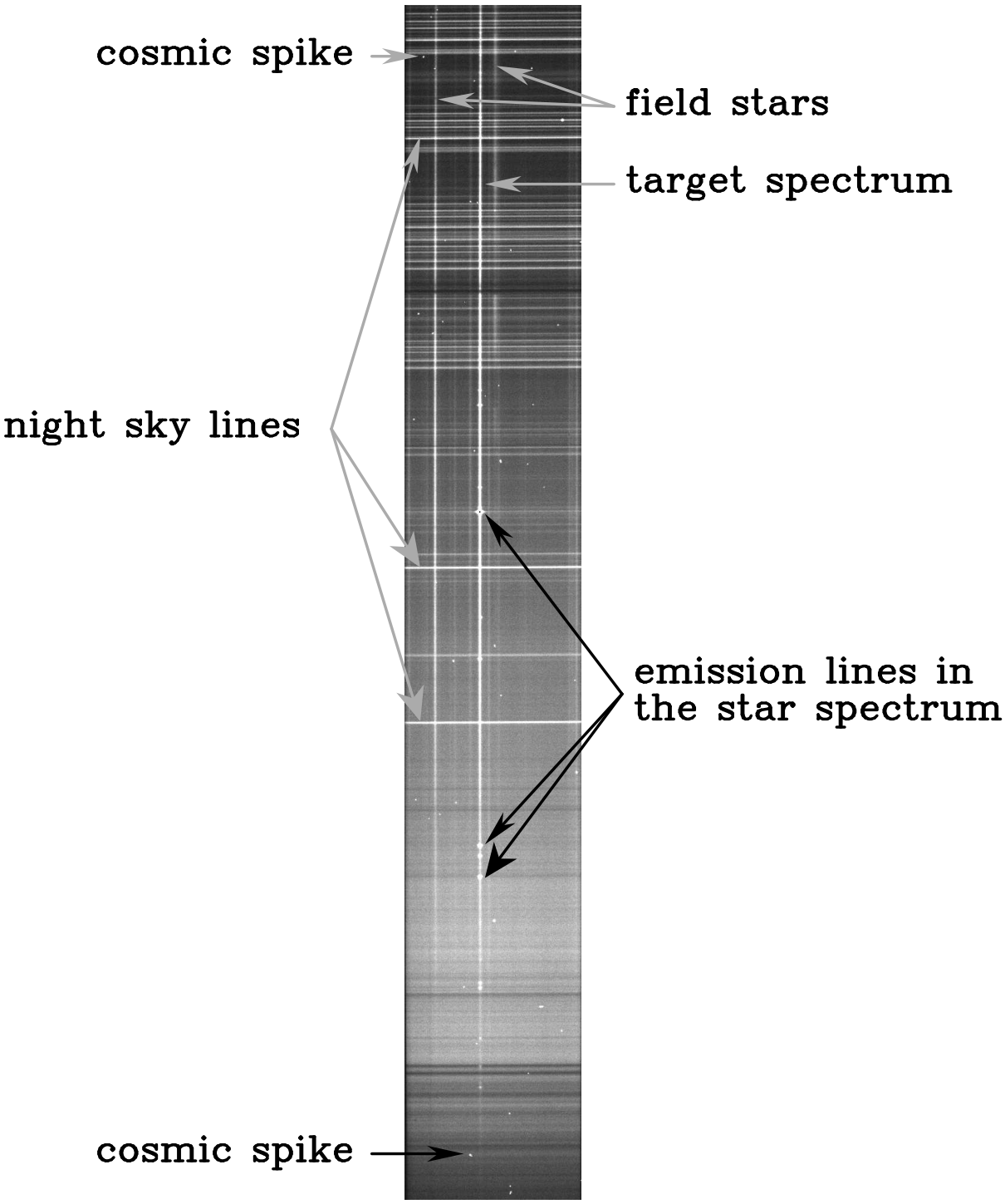
Commands can be thought of as instructions known to your operating system. They are used to perform a specific operation non-interactively (e.g. subtraction of pixel values in two images). But IRAF is in large part an interactive software that allows you to intervene and influence the result of a specific manipulation while it is in the works. We will call such manipulations tasks. When you start a task you actually enter a subprogram, or even a specific graphic environment, that would typically show graphs of results of consecutive reduction steps and allow you to intervene and improve the result interactively.
While you are executing the task you may enter commands in two ways:

(i) by hitting single keyboard keys (do *not* press the ENTER key to execute them): for example q usually leaves the task and ? shows a list of available commands;

Table 1: *List of the demo files with notes as in the original ESO 1.52 m B&C log book. For each object a long exposure with Hα possibly saturated has been obtained to have a better exposed stellar continuum. P.A. is the parallactic angle. All observations were obtained during the night of 28/29 May 1996.*

file	name	size (pixels)	image type	UT start	expt	zenith dist. (°)	P.A. (°)	notes
a0001	FLAT 4SEC	317 x 2060						
a0002	FLAT 4SEC	317 x 2060						
a0003	FLAT 4SEC	317 x 2060						
a0004	FLAT 4SEC	317 x 2060						
a0005	FLAT 4SEC	317 x 2060						
a0006	FLAT 4SEC	317 x 2060						
a0007	BIAS	317 x 2060						
a0008	BIAS	317 x 2060						
a0009	BIAS	317 x 2060						
a0010	V704 CEN 1MIN	317 x 2060	objc	23 37 43	1m	39.4	+50	
a0011	V704 CEN 10MIN	317 x 2060		23 43 37	10m	38.6	+50	
a0012	V704 CEN 20MIN	317 x 2060		23 58 13	20m	37.0	+50	
a0013	V704 CEN CAL	317 x 2060						
a0014	FEIGE 67 STD	317 x 2060	stand	00 47 26	15s	46.8	+00	
a0015	FEIGE 67 STD CAL	317 x 2060						
a0016	RS OPH 1MIN	317 x 2060	objc	05 03 24	1m	26.9	–30	
a0017	RS OPH 8MIN	317 x 2060		05 05 08	8m	26.2	–30	Hα sat.
a0018	RS OPH CAL	317 x 2060						
a0019	AS338	317 x 2060	objc	07 01 24	1m	45.9	–10	
a0020	AS338 10MIN	317 x 2060		07 04 49	10m	45.8	–10	Hα sat.
a0021	AS338 CAL	317 x 2060						
a0022	AS296 30SEC	317 x 2060	objc	08 17 00	30s	38.5	+40	
a0023	AS296 15SEC	317 x 2060		08 21 17	15s	39.4	+40	
a0024	AS296 3MIN	317 x 2060		08 23 49	3m	39.9	+40	
a0025	AS296 CAL	317 x 2060						
a0026	V1329 CYG 2MIN	317 x 2060	objc	08 49 25	2m	64.9	+00	
a0027	V1329 CYG 30SE	317 x 2060		08 54 00	30s	64.9	+00	
a0028	V1329 CYG 7MIN	317 x 2060		08 56 54	7m	64.9	+00	Hα sat.
a0029	V1329 CYG CAL	317 x 2060						
a0030	AG PEG 30SEC	317 x 2060	objc	09 15 10	30s	43.6	–20	saturated
a0031	AG PEG 10SEC	317 x 2060		09 18 24	10s	43.4	–20	Hα sat.
a0032	AG PEG 3SEC	317 x 2060		09 21 52	3s	43.4	–20	
a0033	AG PEG CAL	317 x 2060						
a0034	BD28 4211 STD	317 x 2060	stand	09 33 10	15m	58.6	+00	saturated
a0035	BD284211STD 5MIN	317 x 2060		09 50 40	5m	58.4	+00	
a0036	BD284211STD CAL	317 x 2060						
a0037	LTT9239 STD	317 x 2060	stand	?	10m	16.6	+00	
a0038	LTT9239 STD CAL	317 x 2060						
a0039	DARK	317 x 2060		18 00 10	20m			
a0040	DARK	317 x 2060		18 22 00	20m			
a0041	DARK	317 x 2060		18 44 30	20m			
a0042	DARK	317 x 2060		19 06 10	20m			
a0043	DARK	317 x 2060		19 30 15	20m			

Figure 1: *One of the symbiotic star spectra included in the set of demo images (this is image* **a0012** *of Table 1). Principal features are identified (WIP graphics courtesy T.Tomov).*



(ii) by entering double colon [:] followed by some more letters (i.e. text of instruction) that *are* finished by [ENTER] key. For example to change to 5 the order of a fitting function you have to press [:] [o] [r] [d] [e] [r] [SPACE] [5] [ENTER]

Do not press keys randomly! If in trouble sit back and exit as described in the next section.

IRAF is an open software that allows you to write your own commands, tasks and packages. An interpreter similar to a structured basic is ready at your disposal. This can be quite powerful as you can use any IRAF command or task as part of your code. If you are not satisfied with the interpreter speed the program can be translated and even included in any Fortran type code. It is useful to learn all this at some point. But certainly not now. If interested, check "An Introductory User's Guide to IRAF Scripts" by Ed Anderson and Rob Seaman.

At the level of these introductory notes it makes little difference if you have access to Iraf V2.10 or V2.11. One major difference will be connected with the short script *eso.set* described in Section 8 and Table 7, which is necessary in Iraf V2.10, but can be replaced by a sequence of appropriate commands in Iraf 2.11. We maintained the script because it makes the operations it is meant to perform more transparent to the reader.

2.2 Starting and leaving IRAF

The IRAF should be started from an **xgterm** window by typing

```
cl
```

There is nothing more nasty than to forget how to leave the program. The magic word in IRAF is *logout*. If your machine hangs (not an exception if you pressed two wrong keystrokes) try with [Ctrl]+[C] or [Ctrl]+[Y]. If desperate indeed, closing the window always helps. After such a "hardware" solution you may want to type

```
flpr
```

from the IRAF prompt. This command flushes (removes) any surviving accidental changes of parameters.

airmass:	Compute the airmass at a given elevation above the horizon
asttimes:	Compute UT, Julian day, epoch, and siderial time
ccdtime:	Compute time, magnitude, and signal-to-noise for CCDs
galactic:	Convert RA, DEC to galactic coordinates
pdm:	Find periods in light curves by Phase Dispersion Minimization
precess:	Precess a list of astronomical coordinates
rvcorrect:	Compute heliocentric correction to radial velocities

Table 2: *Some of the tasks available in the package* **astutil**

You leave individual tasks by hitting q. Within a command do not press keystrokes randomly, not even the ENTER key or mouse buttons. IRAF tries to understand them as instructions, so it is easy to get lost in the deepness of IRAF subtasks this way. The only keys which should do no harm are ? (to get help) and possibly q (to end the command).

2.3 Getting help

From within IRAF, type

help command-name

and you will be served with an extensive help (provided that the command exists). If lines scroll too far use

stty nlines=24

to define a 24-line display window. Pressing SPACE scrolls one screen further, pressing d rolls only half screen, and q quits from help display.

Help can be written to a file using

help command-name > file-name

The last part of the command tells IRAF to write the output to a file and not to the screen. In fact ANY command followed by > *file-name* writes its output to a NEW file with the name *file-name*. To APPEND to an existing file use >> *file-name* instead.

To see a one-line description of each command in the package type

help package-name

Try with *help noao* which shows meaning of various sub-packages within **noao**: you will get something like Table 3.

2.4 Remembering, finding and enabling commands

If you can't remember a command name, try with

?

which displays names of commands in the current command package. Similarly, a double question mark

??

shows command names in all open packages.

It may also happen you do know the name of command but you forgot in which package it sits. If this package is not one of the open packages you are working with, IRAF does not recognize the command. So you must know the name of the package you are supposed to open before you can use the commands in it. The simplest trick is to use

artdata:	Artificial data generation package
astrometry:	Astrometry package
astutil:	Astronomical utilities package
digiphot:	Digital stellar photometry package
focas:	Faint object classification and analysis package
imred:	Image reductions package
mtlocal:	Magtape i/o for special NOAO format tapes
nobsolete:	Obsolete tasks to be phased out in a future release
nproto:	Prototype (temporary, contributed) tasks
observatory:	Examine and define observatory parameters
onedspec:	One dimensional spectral red. and analysis package
rv:	Radial velocity analysis package
surfphot:	Galaxy isophotal analysis package
twodspec:	Two dimensional spectral red. and analysis package

Table 3: *Some of the sub-packages available under the package noao*

help command-name

The name of the package will appear at the top of the help screen. So *help display* reveals that *display* is part of the *images.tv* package. If *display* command were not understood, typing

images.tv

would open its package and so make the command accessible. Note that help is always available, even for commands that are not in opened packages.
If you have no idea of the name of command you are looking for, there are two ways out:

(i) search title lines of all help files for a specific word. For example, to list commands that display something type

*help * | match display*

(ii) do a more complete but longer search with

references display

2.5 Using commands of the operating system

To execute a command of the operating system put an exclamation mark (!) in front of it. If using IRAF under Linux, you print the contents of the file *file-name* to the default printer by

!lpr file-name

Many system commands are known to IRAF. A list is written in the login.cl file that appears in the directory you are starting IRAF from. Look for the so-called tasks "foreign". You'll find among others the system command time. Use it directly from the IRAF prompt to display hour/date/year/...

time

2.6 Editing commands

The last command **cannot** be accessed with the arrow key but can be invoked by pressing [e]. Commands further back can be invoked by repeatedly pressing the up-arrow key. You may edit the invoked command before execution by deleting and inserting letters in the usual way. Under Linux it is nice to know that [Ctrl]+[F] deletes one character to the right of cursor, while [Delete] key deletes one character to the left. [ENTER] key executes the command.

The last command starting with *hel* is invoked by typing

e hel

The command *history* shows a list of last typed commands. Command number 15 is repeated by typing

^ 15

2.7 Parameters of commands

Each command in IRAF has its own set of parameters that determine its execution. Parameters are of two types: *required* and *hidden*. Values of required parameters should be supplied each time you use the command. If you forget to do so you will be asked for values before the command is executed. IRAF does not query the values of hidden parameters but uses their current values. You may change the values of command parameters in two ways:

(a) **temporary change**: you add the desired values for the parameters on the command line immediately following the name of the command, as in the following example:

display image–name zrange=no zscale– z1=100 z2=300

This will display the image with name *image–name.imh* (this is a required parameter of the *display* task) coloring the pixels with values between 100 and 300 and disabling the automatic ranging and scaling algorithm ("=no" is the same as "–", and "=yes" is the same as "+");

(b) **permanent changes**: you change the values of parameters in the parameter file. The parameter file is changed typing *epar command-name*, e.g.

epar display

The hidden parameters are printed in brackets. After typing the changes use Ctrl+D to leave the editor and save the changes.

Values of the parameters can be displayed with *lpar* command, e.g.

lpar display

If you screwed the values hopelessly you may return back to defaults with the *unlearn* command, e.g.

unlearn display

Values of parameters for each command are stored in a special file. All these files are saved in a directory (**uparm**) that is specified in the login.cl file in the directory you are starting IRAF from (look for something like set uparm = "home$uparm/"). IRAF performs the following sequence to determine which values should be used:

⟶ read the default values of parameters (stored in the IRAF system)

⟶ override them with the values from the user's parameter file (in uparm directory)

⟶ override them with values supplied on the command line

⟶ query the values of required parameters that were not supplied on the command line.

This philosophy gives plenty of room for customization of the system you are using. And all the changes to parameter files will be remembered even after you leave IRAF. So set the command parameters as you wish. You may even want to start IRAF from a different directory for each type of reductions and specify different uparm directories in the corresponding login.cl files. However beware: too much freedom can be tricky.

2.8 Calculator

IRAF has its own calculator. Simply type

= 180/3.14159

to calculate how many degrees is one radian. You can use parentheses, trigonometric functions[3] and refer to variables. For example

= display.z2 – display.z1

gives the range of grey levels in displaying images with the *display* command.

[3]Trigonometric functions have the argument in radians. This is different than in the *hedit* command where the expression should have arguments of trigonometric functions in degrees. A quite complete set of defined functions is mentioned in the help to *hedit* command.

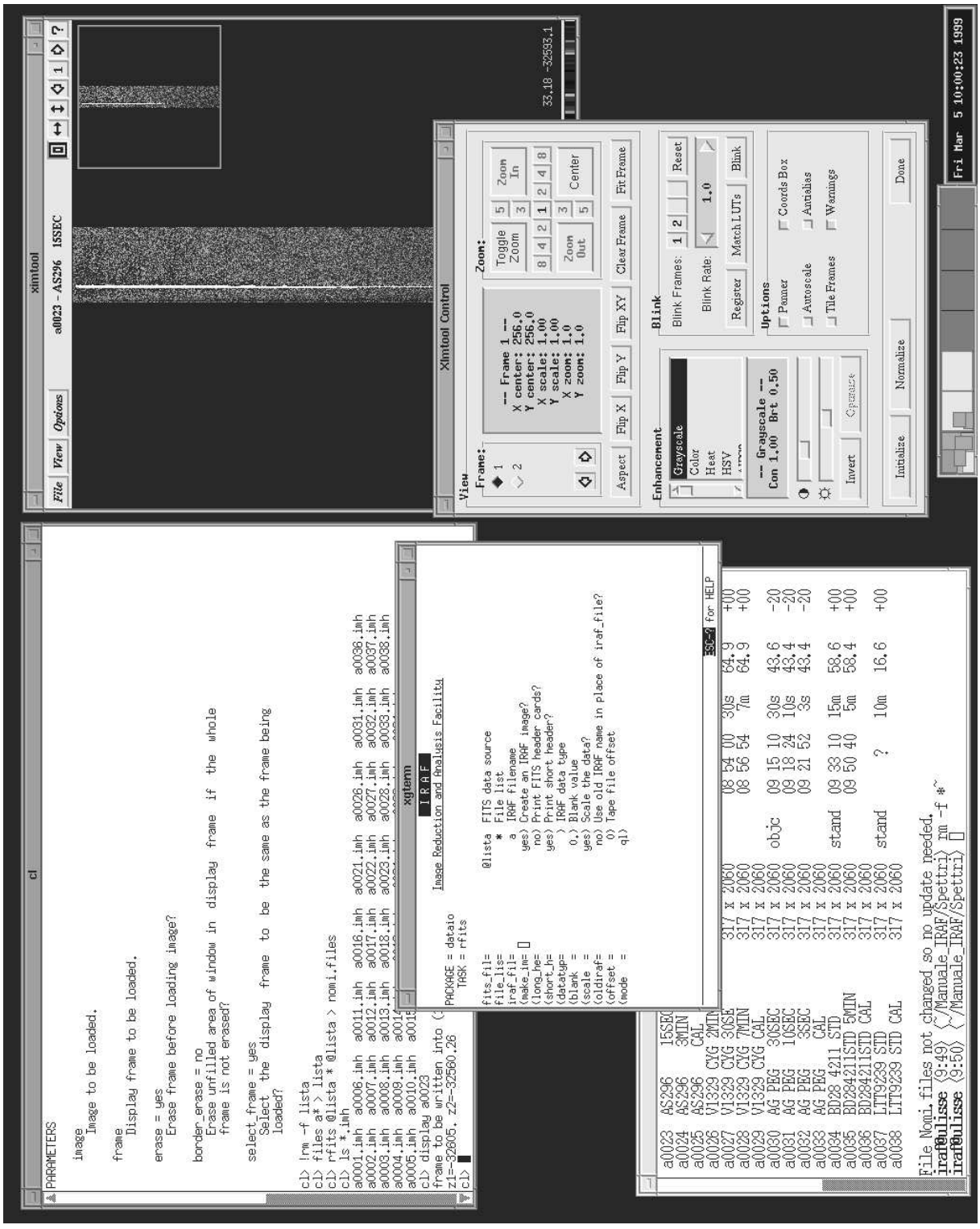


Figure 2: *A typical windows set-up when working in IRAF. Left windows are two xgterm terminals with IRAF prompt and one xterm under Linux. At top right there is the ximtool window to display 2-Dim images, and at bottom right the control panel of ximtool.*

3 IRAF set-up

IRAF works in X-windows, so you should be able to run windows on your display before starting IRAF. [4]

Set the display to 256 colors only (on public IRAF sessions generally this is a default setting). If not, start a 256 color X–environment from a console (text screen) by typing

```
startx -- -bpp 8
```

The reason for this is that the current versions of programs used to display 2-D images (*ximtool* and *saoimage*) do not work properly in high color environments. This problem is being solved, expect good news by end of 1999. But if you do not plan to use a 2-D image display any color depth should work fine.

IRAF interacts with the user through a terminal window. It is very important that you run IRAF from a **xgterm** window. This is a custom dialect of the convenient **xterm**, but much more stable, with better colors and with much safer handling of user input. Xgterm comes with IRAF, so you should have it. You can start a xgterm from a xterm by typing

```
xgterm &
```

Before invoking IRAF by typing cl from a xgterm, check that in the local directory there is the file login.cl and the directory uparm (login.cl is the IRAF configuration file, uparm is the directory where the parameter files of all commands are stored). If they are missing, you can create them by typing

```
mkiraf
```

Edit the login.cl (only if this is really necessary) to change anything in the set-up (for example where to store the pixel files). When IRAF is started it first learns what is written in the login.cl file. Therefore once started you may change directories at will, because IRAF will never look in the login.cl file again during this session. Note also that you can run mkiraf in different directories and adapt the contents of the login.cl file and the uparm directory to optimize the configuration for different kinds of reductions (for example when you work on Echelle spectra you perhaps want to have an IRAF configuration different from the optimal one for B&C spectra: to avoid to

[4] IRAF is very easy to install, just get the files from the net or from the CD, but be careful to follow instructions in the installation guide closely. You'll need to know the root password even though just for one of instructions during IRAF setup. About 100MB of space is needed for IRAF source files and executables, and any PC from a 486 upwards with some 40MB of RAM should work fine. Space is needed for the astronomical images too [as a guideline consider that a 1024×1024 pixel image requires 2.5 MB and a 512x512 pixel about 600 KB. During an observing night with the Asiago Echelle+CCD spectrograph (1024×1024 pixel CCD), you can collect up to 100 frames (divided among flats, bias, dark, comparison spectra and object spectra) – equivalent to 250 MB – and you'll probably want to have all these files accessible at the same time on your computer]. The wide spread use of CD-ROM writers to store data is now alleviating the heavy demands in terms of disk space. Also large disks are already very reasonably priced. Take in mind that on many public workstations available for image reduction under IRAF, you should be able to log in as a user named *iraf* and you will be immediately presented with the appropriate X-environment and the *cl>* prompt.

reconfigure your IRAF environment every time it is better to use different directories [and therefore different login.cl and uparm] for Echelle and B&C spectra reductions).

3.1 Windows

Open two xgterm windows and start IRAF with cl in both of them. The first window will be used for display, the second for editing command parameters. Another xterm can be useful for text operations. To get the scroll-bar in each window press together the Ctrl key and the right mouse button and select 'scrollbar'. A good configuration is presented in Figure 2.

3.2 Image display

IRAF uses the program *ximtool* to display 2-dimensional false-color plots of CCD frames (as an alternative you may use the program *saoimage*). Please note that by default ONLY ONE ximtool may be opened at a time. Four different images (frames) can be displayed within the ximtool (and make to blink). The frame number (1–4) is a parameter of the display task. To start ximtool type

!ximtool &

This opens a 512x512 window. If your picture has a larger number of pixels you have two possibilities to show it in full: the window of ximtool is enlarged by dragging its corners, or you view a small portion of the image at a time. Anyway you should tell IRAF that the image is actually larger than 512x512, or ximtool will not be able to show it correctly. The size of the image is controlled by the **stdimage** IRAF variable. To see a long rectangular display of 976x3040 pixels (useful for B&C spectroscopy) type:

reset stdimage = imt28

A large square picture (of 1600x1600 pixels) is obtained instead with

reset stdimage = imt4

This is suitable for Echelle or large direct imaging CCD frames. More choices can be found by listing the appropriate file

page iraf$/dev/graphcap

In any case the data are displayed using the

display image-name

command. The small rectangle on the upper right corner of the ximtool window is the panner: to move around a frame too big to be shown by ximtool as a whole, click on the corresponding rectangle and drag it around. To disable *zrange* and *zscale* parameters when you want to input grey-scale limits *z1* and *z2* explicitly, type e.g.

display image-name zrange– zscale– z1=1000 z2=10000

To zoom a portion of the image define the zoomed window holding the left mouse key and then (with the cursor in window) zoom it with the right mouse key. The picture is un-zoomed by choosing Unzoom from the View pull-down menu. To hardcopy the image choose "print" option in ximtool. The save option in ximtool allows you to save (the displayed portion of) the image in GIF or FITS format. A graphic control panel can be accessed from the *Options* pop-up menu on the ximtool window. Typing *man ximtool* from a terminal prompt gives a full help on ximtool capabilities.

There are some other interesting commands to display and examine 2-D images:

imexamine image-name

enters an interactive mode after making sure your image-name is displayed. Pressing [s] in the ximtool window gives you a perspective plot of the region around the cursor. This may be useful to decide if a certain strike is a cosmic ray or not. There are many other tricks in *imexamine*. Press [?] to see a short help (you exit help with [q] and [ENTER]). As always you exit the imexamine task by hitting [q].

Another useful command to plot 2-D images is

implot image-name

It plots cross-sections across the image. Pressing [l] or [c] gives a slice along the line or column with the number equal to the y-position of your cursor as marked on the right side of the plot. Again use [?] within the task or type *help implot* to learn more.

4 Reading, writing and printing data

IRAF has its own internal image format. To be appropriately handled in IRAF, each FITS file is separated in two parts: image header and pixel file.

- The **header** contains log-book infos, like how, where and when the CCD image was recorded. IRAF stores it with *.imh* extension. It is an ASCII file with 80 characters per line.

- The **pixel** file is a binary file with extension *.pix* that is stored in a special directory (specified as *imdir* in the login.cl file)[5]

The usual way to store and transfer image files is to have them in Flexible Image Transfer System, or FITS format. So you should know how to read and write FITS files. It may be a good idea to type

help dataio

to see the available commands for reading and writing files in IRAF, and proceed further with *help command-name*

[5]This directory is usually the same for all users of the computer. The reason for this is that pixel files are large and such philosophy makes purging of old files by system manager easier. Nowadays you will be perhaps the only user, still the idea of a separate pixel directory is useful: it keeps large files out of sight.

4.1 Using lists of files

Most IRAF commands can work on single files or they can execute a specific task on many files in succession. So it is very useful to produce a one column file with a list of names. The simplest way is to use the *files* command. For example

files a > list_a*

writes to file *list_a* a list of all files starting with *a*. The same command can be made more sophisticated:

files cat.%fts%imh% > log.txt*

will write the list of all files starting with "cat" and extension ".fts" to file log.txt but their ".fts" extensions rewritten as ".imh" . The *files* command never overwrites an existing output file. A message like *"ERROR: cannot open log.txt for writing"* means that an existing file *log.txt* should be deleted first with the *delete log.txt* command. Almost any IRAF command accepts 3 sorts of input:

1. single filenames, e.g. *a0001.imh*
2. coma-separated sequences of filenames, e.g. *a0001.imh,a0002.imh,a0003.imh*
3. lists, e.g. *@list_a*. Here the @-character tells IRAF to open file list_a and use its first, second, third,... line for input.

4.2 Reading FITS files to IRAF format

The command is *rfits*. On a single file it is used as follows:

*rfits original-name * new-name*

On a list of files it can be used in several ways:

- *rfits @input_list * @output_list*
  reads the file name from the first line of the file input_list and stores it under the file name from the first line in file output_list, then continues with the second line etc.

- *rfits @input-list * c*
  will give as output the IRAF files c0001, c0002, ... This may look easier but it gives you less control over filenames. And knowing what is meant by a certain filename is crucial.

- *rfits @input-list * " old+*
  restore the original file names (the ones used by IRAF before creating FITS files). An alternative form is *rfits a* " " old+*

The names and dimensions of files are written to the screen as they are read from FITS to IRAF format (or redirected to a file using the > Unix command).

One may be interested in checking the content of a magnetic tape or DAT without the FITS files actually being read to the disk in their *.imh* and *.pix* parts. To do

this change the parameter *make_image* of the *rfits* command to "no". This can be done permanently by modifying the parameter file with *epar rfits*, or temporarily by inserting *make_ im=no* in the command line:

rfits @input-list make_im–

4.3 Writing IRAF files in FITS format

Use *wfits* command. To write IRAF files (i.e. those split into header and pixel parts) to the disk as FITS files use the syntax

wfits file-name new-name

or

wfits @list1 @list2

to write files from the list *list1* with names as stored in *list2*. To write directly onto a tape device (for example a DAT) all IRAF files which names start with *fy* use

wfits fy.imh mttk1*

In this example *mttk1* is the logical name of the DAT device. Check with your local IRAF guru for the appropriate name of your device (it could be perhaps something like *mtdat0*).

4.4 Changing filenames for a list of files

This is not strictly an IRAF operation, but it is worth to mention here because from time to time you may find convenient to change the names of a list of files too long to be performed manually file-by-file with the normal mv Unix command. It works with any type of file, not only FITS ones. First create a list of your files (for example all those which names star with the a character)

ls a > log.list*

Delete from the list those you don't want to change. Suppose the file *log.list* has the first lines as

```
a0001
a0006
...
```

Edit the ASCII file *log.list* with an editor and modify each line as in the example

```
mv  a0001  ESO-28may96-0001.original-2D.fts
mv  a0006  ESO-28may96-0006.original-2D.fts
...
```

(this is easily done in Unix editors that allow you to edit by columns, like our favourite *joe* editor, where you access the column editor by pressing Ctrl+t+x. A good alternative is the *nedit* editor). Now change the attribute of the *log.list* file to make it an executable

```
chmod +x log.list
```

and to enjoy the magic by typing

```
log.list
```

An alternative is to skip the *chmod* step and type simply

cl < log.list

which tells IRAF to read input from the file *log.list* and execute it line by line (the Unix command mv is understood by IRAF).

4.5 Deleting files and images

Normal files are removed with the *delete* command, e.g.:

delete log.lst*

But this will not work for IRAF images as they are protected against accidental erasing. To delete IRAF images use *imdelete*. It will delete both the *.imh* and *.pix* files. For example

imdelete a.imh*

deletes all images which names start with *a*.

4.6 Set bits appropriately

It may happen that your images look to be composed of pixels with negative counts (!). This is generally due to a bit switching and different representations of integer numbers while writing/reading files from one operating-system/software to another. For example, in the demo images (with actual counts extending from 0 to 65536, equivalent to 2^{16}) the counts range from -32768 to +32768 (they were written under MIDAS at ESO). To restore the original 0 to 65536 scale, add 32768 counts to all pixels in the images (listed in the input filename list *list_a*)

imarith @list_a + 32768 @list_b calctype=real pixtype=real

The bit-corrected images will have names as in file *list_b*.

4.7 Printing text, graphics or image

This section could appear later as we did not encounter any graphical or image data yet. Still, it is essential to put results of intermediate reduction steps on paper for later

reference. So we summarize printing in IRAF before starting with the real reductions. A brief guide is as follows:

- **Text** can be printed directly

  lprint file-name

  or with redirection

  help astutil | lprint

- **Line graphics** (i.e. graphic windows of *apall*, *splot*, *identify*, etc.) can be copied to a printer by pressing the [=] key. These screens can be also saved to a postscript file by typing

  :.snap epsfl [*ENTER*]

  This creates an encapsulated postscript file with the name *sgi???.eps* where *???* is a three-digit number. Of course you can rename it at will. There are two possible orientations: *epsfl* gives landscape and *epsf* portrait plots. In the task *imexamine* you'll have to press [g] to enable graphic cursor before typing the *:.snap* command and [i] to call back an image cursor.

- **Halftone graphics** printing in *ximtool* is self-explanatory: see the *File* drop-down menu.

5 Flat, Bias and Dark frames

The flat-fielding of CCD direct imaging frames is easy because all pixels are illuminated by light of the same range of wavelengths. In spectroscopy the situation is more complicated: for example each row of a Boller& Chives spectrum is illuminated by light of a different wavelength and from bottom to the top of a low resolution spectrum the wavelength can change for as much as 6000 Å.

Flat-fielding is used in direct imaging to correct *both* for pixel-to-pixel high spatial frequency variations *and* for low spatial frequency response over the whole frame.

In spectroscopy, flat-fielding is used *only* to correct the pixel-to-pixel high spatial frequency variation in the direction perpendicular to dispersion. Standard stars are used both to correct for the broad response function at different wavelengths (low spatial frequency), as well as to set the zero of the flux scale.

The goal of flat-fielding in spectroscopy is therefore to obtain a flat with an average value of 1.000 over the whole frame. Intensity of each individual pixel gives its sensitivity compared to adjacent pixels (which are illuminated by radiation of similar wavelength). Correction for flat will therefore be the division of the science frame by this *masterflat* in order to remove the effect of varying sensitivity of adjacent pixels.

In the following we will use numerical values of frame dimensions, bias values, RON, ADU gain, etc. appropriate for the real CCD images that come as a demo with these notes (cf. Sect. 1.1).

Before performing the flat-field correction, appropriate bias– and dark–frames have to be removed from the images. Let's start with them.

5.1 Bias subtraction

First the bias should be subtracted from all images. If *list_b* contains the input filename list and *list_c* is the output filename list (in our case output files will be named as c0001 etc.)

imarith @list_b – 176.3 @list_c calctype=real pixtype=real

The last two parameters guarantee that the internal calculation and the output are done with real numbers. You can of course write that to the parameter file with the *epar imarith* command. We assumed that bias has the value 176.3 (a constant value over the whole frame is characteristic of good CCD cameras and associated electronics. In case of BAD bias characteristics of your camera, you have to subtract bias images. They are filtered median images of several bias exposures or they are artificial bias images built from the overscan region – when available). To check some simple statistics of an image (for example a bias) type

imstat file-name

5.2 Dark subtraction

All electrons counted during CCD readout are not due to incoming light. Atomic thermal motion of atoms in the silicon lattice causes generation of additional free electrons. We have to subtract these dark–current counts from science frames. On modern, high quality CCDs the number of dark electrons produced per pixel per hour in liquid N_2 cooled CCD is very low, generally similar to the read-out noise. On exposures lasting up to 10-15 minutes the effect of dark current is almost undetectable. In case of demo images (a0039 to a0043 in Table 1) the dark current amounts to only 10 electrons per pixel per hour.

The dark counts quite consistently increase linearly with time. Therefore if in a night observing session all science exposures have been shorter than – say – 20 minutes, it will be enough to secure ∼5 dark frames of 20 minutes integration (with dome lights switched off, dome slit and dome blinds closed, shutter closed). The bias level estimated from the overscan region would be subtracted from each of them. Finally they would be combined into a median frame which represents the pixel-by-pixel dark current liberated in 20 minutes. This median frame, which we call **masterdark**, will then be subtracted from science 20-minute exposures to correct for the dark current. Note that the median takes care of cosmic strikes and read-out noise in individual dark frames. To correct a 5-min science frame the 20-min *masterdark* would be divided by 4 before subtracting it[6].

[6]See also a discussion of dark frame correction in the *help* file of the command *ccdproc*

5.3 Trim the images

After bias subtraction the overscan region has now a mean value of 0.0000 and it is not necessary any more. In the enclosed spectra the bias overscan region extends between rows 2049 to 2060. Moreover the first few rows seem to have some problems and there is little loss of information deleting them. We trim the first and last few rows of these images by the command

imcopy @list_d @list_e

where *list_d* is the input filename list (let's assume file names are c0001, c0002, ...) with lines written as

c0001[*,10:2040]
c0002[*,10:2040]
...

and *list_e* is the output filename list

e0001
e0002
...

Note that all images, i.e. scientific exposures, calibration lamps, as well as flat field exposures, should be trimmed in the same way. It is generally a good idea to trim also the first and last columns in case they are of little scientific interest or are affected by edge-reading problems.

5.4 Preparing the master flat

Now we take all flat frames (with equal exposure times) and produce a median image **medflat** (the use of median will filter out sporadic cosmic ray hits and deviating measurements) with

imcombine e0001,e0002,...,e0006 medflat combine=median

An example of the *medflat* is in Figure 3a. Next we make the master flat which reflects only local sensitivity differences between pixels[7]. This means we should filter out the strong and general wavelength dependence of CCD's sensitivity and spectrograph+telescope+... response. For the 317×2060 pixel images now trimmed to 317×2031 pixels use:

blkavg medflat[20:310,] avcol_in 291 1*

[7]Strictly speaking a dark frame of corresponding exposure time should be subtracted from flats too. However in the case of enclosed test images the exposure on flats has been so short (4 sec, cf. Table 1) that no contribution from dark current is expected.

Table 4: Example parameters for **ccdproc** command

overscan =	yes	flatcor =	yes
trim =	yes	column direction =	(right value)
zero level =	no	overscan strip =	[*,2049:2060]
darkcor =	yes	trim section =	[*,10:2040]
fix bad lines =	no	flat =	masterflat
illum =	no	dark =	masterdark
interactive =	yes		

We have averaged over columns 20 to 310 (i.e. 291 columns) and created the single line spectrum **avcol_in** that has dimensions 1x2031 pixels. The columns on the edges have been ignored in preparing *avcol_in* to avoid any possible CCD reading problems of the very first or very last columns. Now, typing

blkrep avcol_in avcol_out 317 1

the *avcol_ in* single column is expanded back to a 317 × 2031 pixel image by putting 317 avcol_in images side-by-side and saving it as **avcol_out**. The result is plotted in Fig. 3b. Finally, by dividing the medflat with this avcol_out we obtain the **masterflat**

imarith medflat / avcol_out masterflat calctype=real pixtype=real

Each row of masterflat has a mean value of 1.000, with local deviations being due to pixel-to-pixel differences in sensitivity.

5.5 Flat field correction of science frames

Create a list *list_e* of all bias- and dark–corrected as well as trimmed science frames and flat-field them with

imarith @list_e / masterflat @list_g calctype=real pixtype=real

where *list_g* is as usual the output filename list (suppose they are of the type g0001, ...).

Alternatively use **ccdproc** command (from imred.ccdred package) to make bias + dark + flat field + trimming manipulations at the same time. If the overscan region is in rows 2049 to 2060, use the parameters in Table 4. You may use a spline3 function to fit the overscan.

5.6 Removing cosmic rays

To remove cosmic rays in the regions far from the spectrum tracing you may use the task *imedit*. However we prefer not to remove cosmic rays this way. The task *apall* does it well (and fast! see next section) if its RON and e-/ADU parameters as well as wide-enough median-type extraction of the background are set properly. Suppose

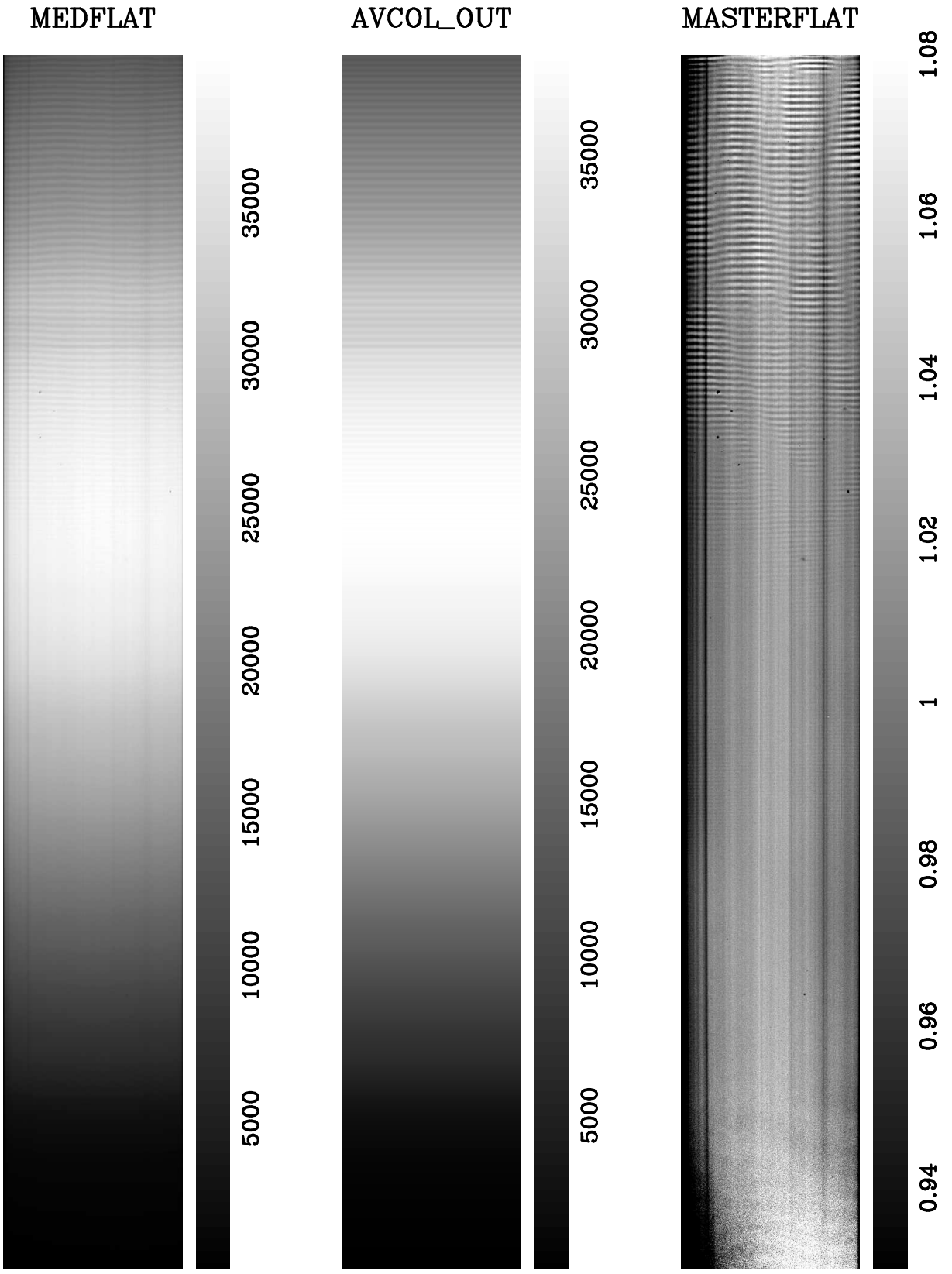


Figure 3: *From left to right: [a] the* **medflat**, *[b] the* **avcol_out**, *and [c] the* **masterflat** *(WIP graphics courtesy T.Tomov).*

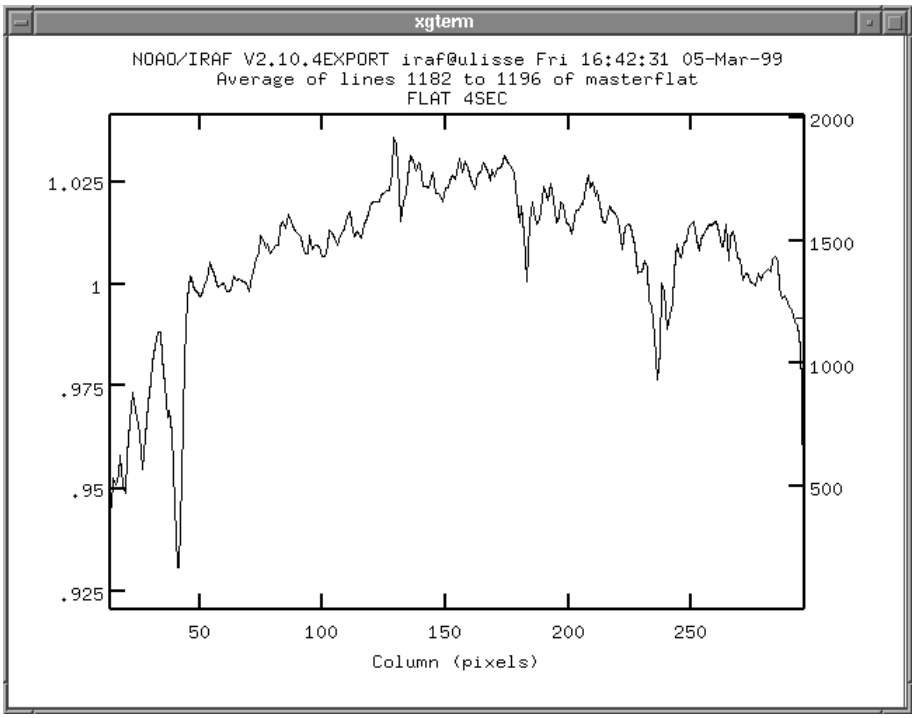


Figure 4: *A tracing across the* **masterflat** *(cf. Figure 3c) perpendicular to the dispersion direction. The columns with reduced sensitivity are evident (darker ones in Figure 3c).*

you are estimating the sky background from 20 columns on one side of the spectrum. Taking the median value for them automatically rejects any cosmic that may affect one or two of the sky background pixels. For the particularly nasty cases when a cosmic ray hit directly on the spectrum we prefer to remove it at the very end of the extraction by hand, editing the final extracted spectrum.

In some cases (particularly when the sky background changes over a few pixels like in the case of a supernova superimposed on a galaxy) it is preferable to work with sky backgrounds free from cosmic rays. To do it use the command

imedit file-name radius=2

Place the image cursor on the cosmic ray and press [b], which replaces it with a properly averaged background over a radius of 2 pixels. In several cases it is preferable to work pixel-by-pixel and in this case a replacement *radius=1* is more appropriate. Useful keys are also [c] and [l] that replace a cosmic hit by average of pixels in the column/row direction. Use them when working on the slope of stellar tracing or close to sky lines.

6 Aperture extraction

Use *apall* (noao.twodspec.apextract package) which is actually a collection of many tasks. After determining the aperture (object's part of the tracing) and background (which sky regions will be used for subtraction) it traces the spectrum (finds its path in the direction of dispersion) and sums it up in a mono-dimensional line.

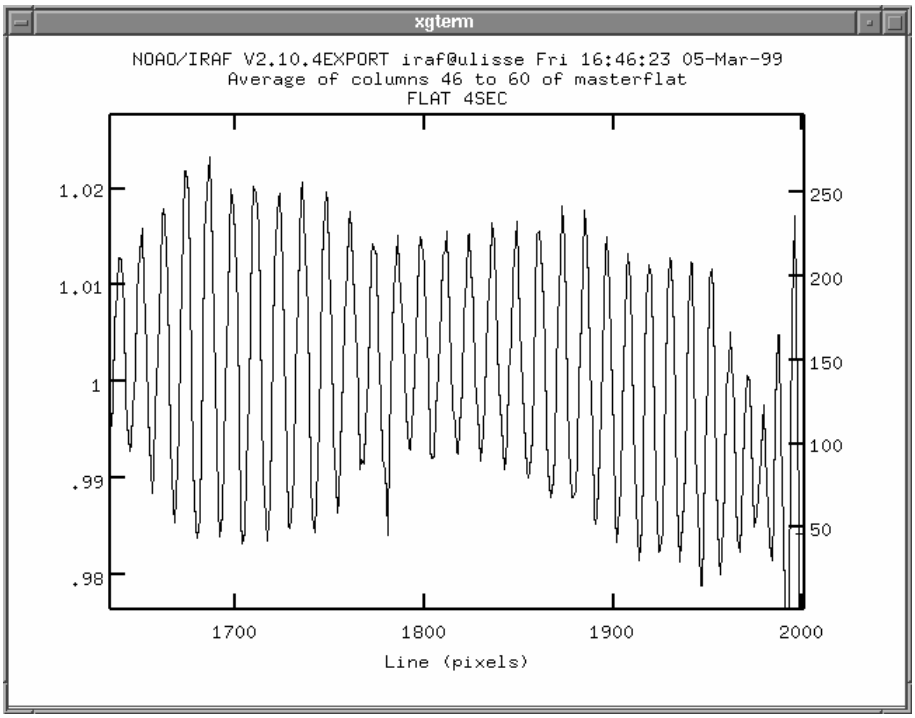


Figure 5: *A zoomed-in tracing across the* **masterflat** *(cf. Figure 3c) parallel to the dispersion direction. These pixels are illuminated with near-IR light. The interference fringes evident in this plot are due to internal reflections of the thinned CCD.*

Apall does so many operations that a vast number of options should come as no surprise. Some highlights are given below. Refer also to a commented listing of its parameter file.

6.1 The *apall* task

Understanding the meaning of all *apall* parameters means to fully understand the basics of CCD spectral handling. It requires some work, much beyond the scopes and limits of the present brief introduction. We strongly recommend the reader to carefully read the help file that comes with *apall*

help apall

and to take time and to experiment with its various parameters. What follows is just a quick overview. Remember that with

unlearn apall

you reset the values of all parameters to their defaults values (which are in several cases quite good choices).

All interactive options (except review extractions) should be set to *yes*. Background subtraction should be yes, with appropriate fitting function (a b_order=1 is normally adequate). The background region is set relative to aperture center: a range –30:–10,10:30 indicates that the background is estimated over the 21 pixels extending

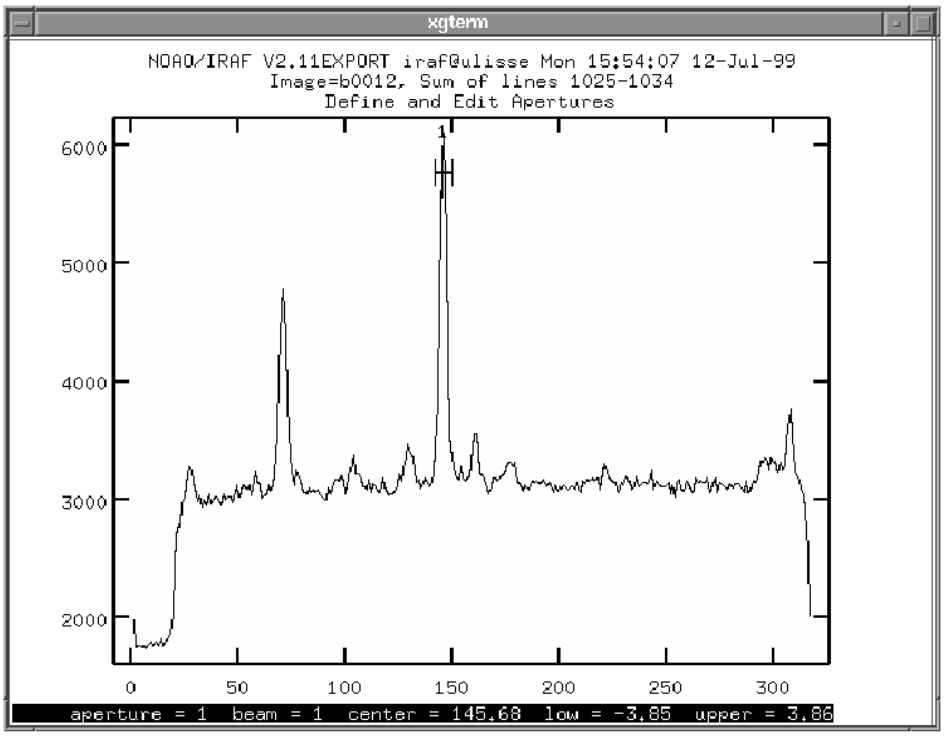


Figure 6: *Identification of aperture #1 with the star of interest.*

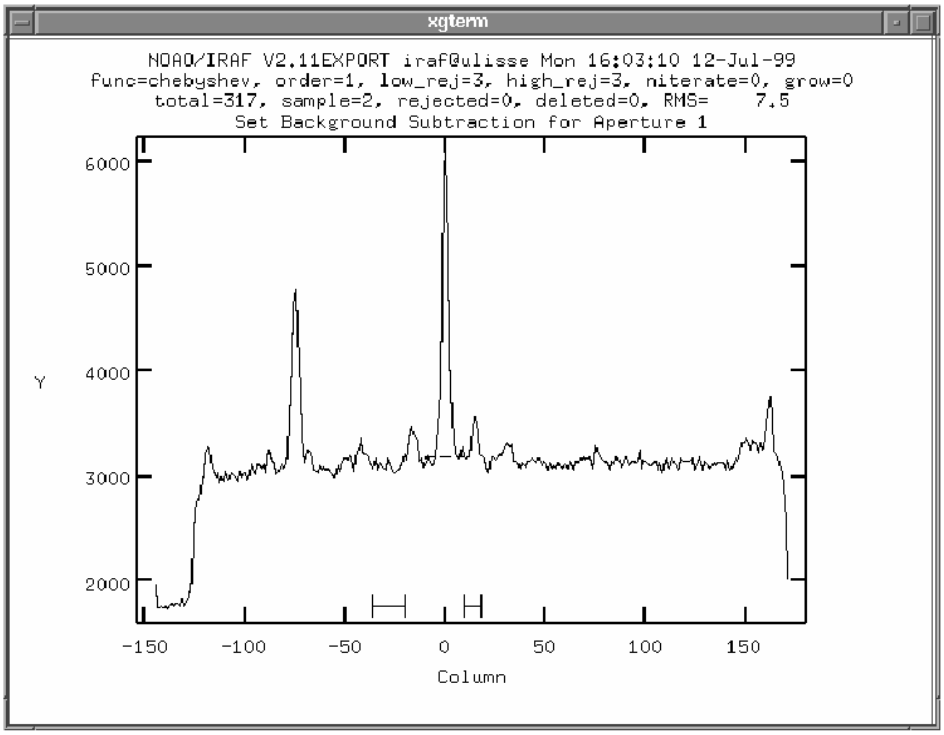


Figure 7: *Determination of sky background (the region to the right is not a good choice because it include a background faint star, and it must be re-selected).*

from pixel 10 to pixel 30 on the left of the spectrum and similarly for the right side. Use some moderate averaging of the background, best using the median of 3 points: so use *b_ naver = –3*. The other important parameters are (use the values appropriate for your spectrum):
lower= −4.0
upper=4.0 (or something reasonable)
llimit= −5.0 (approx. half width at zero intensity,)
ulimit=5.0 (n.b.: do NOT leave it INDEF!)
backgro=fit (we subtract the sky background)
weights=variance[8].

Do not forget to set RON and e/ADU values (4.2 and 1.2 respectively in the case of demo images). Control extraction of every single spectrum.

The task starts by asking what are the filenames and how many stellar tracings you want to extract from each image. Answer *yes* to all queries if the reduction should be interactive. Now comes the first interactive step: deciding on the *position of the star(s)*. You are presented with a graph similar to Fig. 6 [if the crossection turns out to be in the wrong direction (parallel instead of perpendicular to the dispersion direction) set the dispaxis parameter of the noao.twodspec.apextract package appropriately, using *epar apextract*.] If the position of the star is wrong you can delete computer's choice by hitting [d]. New aperture (i.e. star) on the cursor's position can be added pressing [n], or [m] that chooses exact position of a nearby peak. If the left and right limits of the star tracing (small vertical bars on the edges of a horizontal line above the graph) are not good change them by pressing [l] and [u] on left and right.

IRAF should now know the *position of the background used for this star*. This background will be used to determine intensity of the sky emissions (i.e. what would be observed on the position of your star if your star would vanish) that will be subtracted from pixels belonging to the star. This is very important so *always* control the background for each star you extract.

Enter background fitting with [b]. Often the graph does not cover enough pixels on left and right from the spectrum so use [w][m] to enlarge it. The graph should now be similar to Fig. 7. The situation in Fig. 7 is not satisfactory. Peaks in the background are faint stars that can spoil the background level. First delete both background ranges by pressing [z] twice. Then define a new background interval by pressing [s] with cursor on its left and its right edge. You may define as many background intervals as you wish. In Figure 7 a reasonable choice would be to press [s] with cursor at x= –40, –30, 35, and 70. Finally fit the new background with [f] and exit the background subtask by [q].

It is very important to control the quality and correctness of the aperture tracing, so do this for every spectrum. Here [d] deletes the point nearest to the cursor, [u] undeletes it, and [a] adds new point at cursor (after asking for how many points it should count). [f] remakes the fit, [r] redraws the graph and *:order 3* changes fitting

[8]Variance extraction does the almost the same as a simple sum of intensities (*weights=none*) for bright stars, but performs better on weakly exposed spectra.

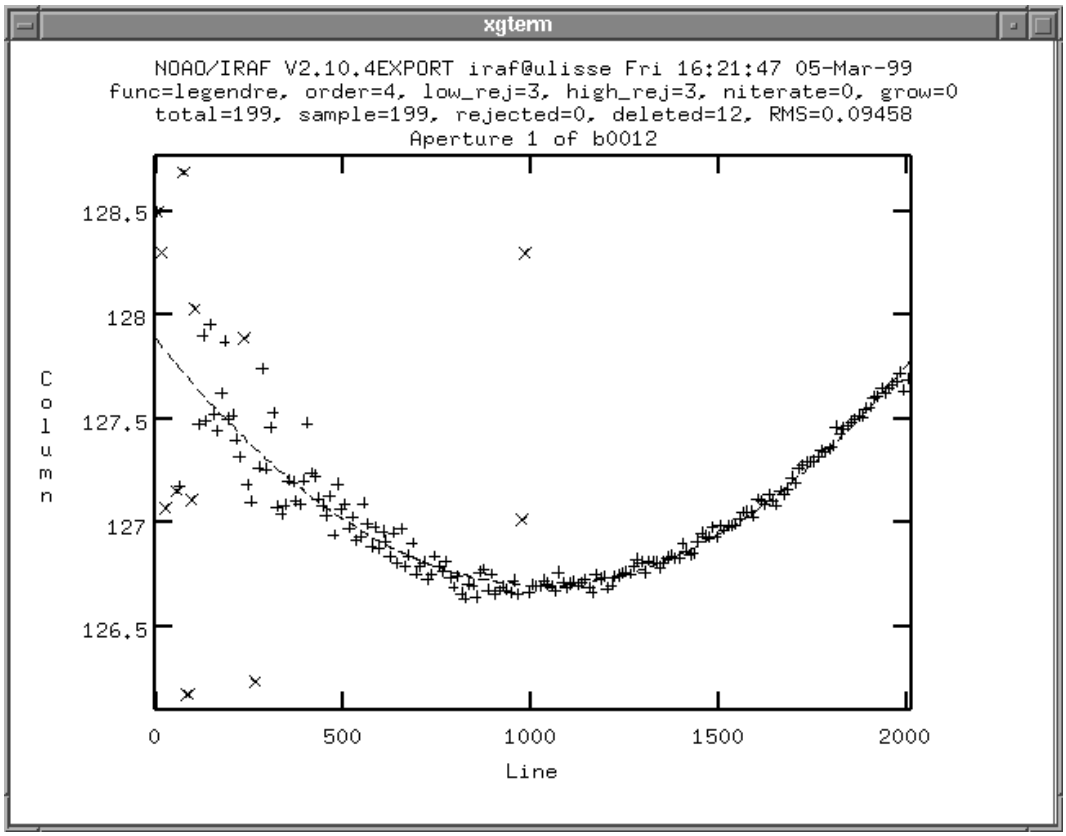


Figure 8: *Fitting the tracing to the spectrum.*

order to 3 (typing *:o 3* will suffice). Exit tracing by hitting [q]. If you can't get a good tracing:

(i) control the direction of the dispersion;

(ii) control the width parameter which should be close to the FWHM of the spectrum;

(iii) control position of the initial aperture (use *epar apall* and change the value of *line* to the pixel position of e.g. Hα if you are dealing with an emission type object);

(iv) increase the number of lines IRAF sums up to get each tracing point by setting *t_nsum* (and *t_step*) accordingly;

(v) increase the number misses (e.g. *t_nlost*=5) before the tracing is aborted;

(vi) get a cup of tea and meditate on what you did: is filename right? where exactly it crashes? etc...;

(vii) if desperate indeed read the manual, i.e. open Massey et al. on page 20.

The result of *apall* is a monodimensional file named *e0001.0001* (if the parameters from Table 5 are used) or e0001.ms (if *apall* is run from imred.kpnocoude package). Note

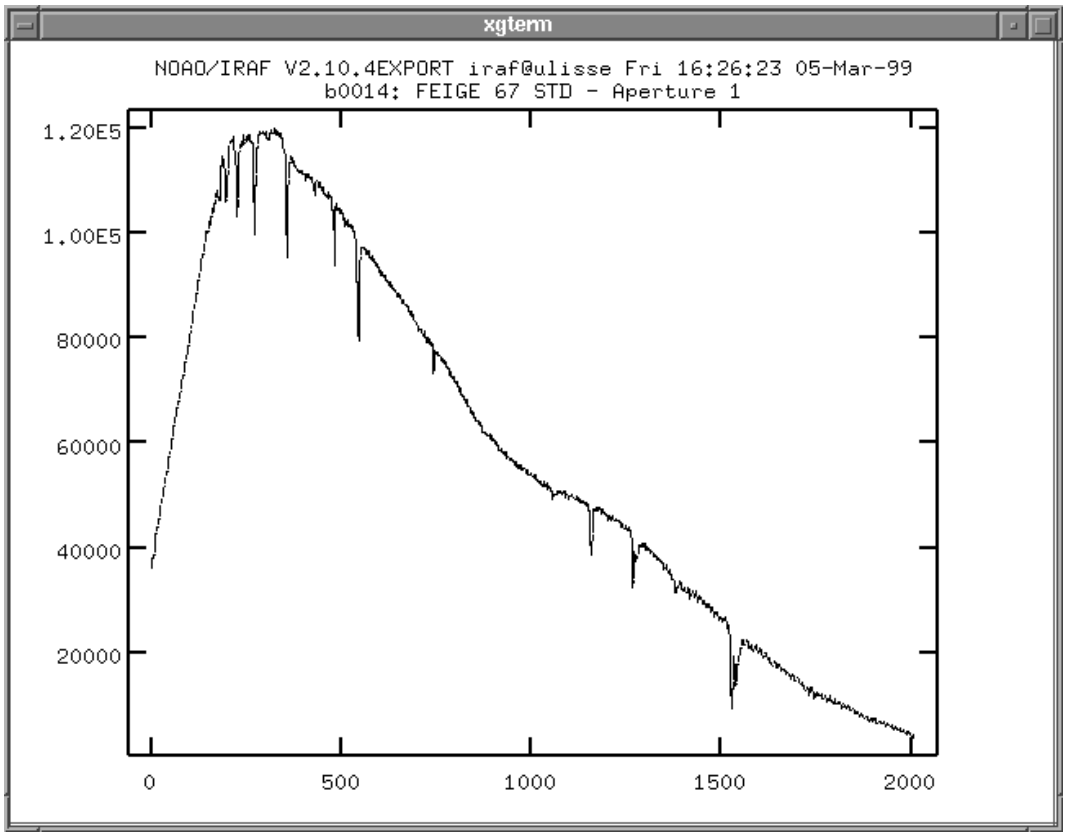


Figure 9: *Reviewing the extracted spectrum (abscissae still in pixels and ordinates in counts).*

also that *apall* writes the critical parameters that were used for extraction to a text file in the database directory[9]. For *e0001.imh* file the file would be *database/ape0001* where 'ap' stands for 'aperture definition'. This remark may come handy if you forget values of parameters that were used for a particular spectrum extraction.

People noticed that *apall* sometimes does not obey the parameters from its parameter file. This is due to the fact that *after* it reads the parameter file it checks if the *apall* task was already run on the image and if so adopt the saved parameters. This is usually useful. If you do not agree delete the corresponding file in the database (cf. previous paragraph).

[9]It seems appropriate to clarify what is a *database*: it is a normal subdirectory in which IRAF stores the results of aperture extractions and wavelength solutions. These results are stored as normal text files. Their names are image names beginning with *ap* or *id* for aperture and wavelength solutions. This comes handy sometimes if you want to copy such solutions between images or delete a solution you accidentally wrote.

Table 5: *Quick outlook of the parameters of the* **apall** *task (noao.twodspec.apextract).*

input	=		List of input images	
(output	=	””)	List of output spectra	
(apertures	=	””)	Apertures	
(format	=	”multispec”)	Extracted spectra format	use onedspec
(references	=	””)	List of aperture reference images	(a)
(profiles	=	””)	List of aperture profile images	(b)
(interactive	=	yes)	Run task interactively?	yes
(find	=	yes)	Find apertures?	yes
(recenter	=	yes)	Recenter apertures?	yes
(resize	=	yes)	Resize apertures?	yes
(edit	=	yes)	Edit apertures?	yes
(trace	=	yes)	Trace apertures?	yes
(fittrace	=	yes)	Fit the traced points interactively?	yes
(extract	=	yes)	Extract spectra?	yes
(extras	=	yes)	Extract sky, sigma, etc.?	yes
(review	=	yes)	Review extractions?	no
(line	=	INDEF)	Dispersion line	(c)
(nsum	=	10)	Number of dispersion lines to sum or median	(d)
(lower	=	–5.)	Lower aperture limit relative to center	(e)
(upper	=	5.)	Upper aperture limit relative to center	(e)
(apidtable	=	””)	Aperture ID table (optional)	
(b_function	=	”chebyshev”)	Background function	
(b_order	=	1)	Background function order	(f)
(b_sample	=	”-10:-6,6:10”)	Background sample regions	(g)
(b_naverage	=	-3)	Background average or median	(h)
(b_niterate	=	0)	Background rejection iterations	2
(b_low_reject	=	3.)	Background lower rejection sigma	
(b_high_rejec	=	3.)	Background upper rejection sigma	
(b_grow	=	0.)	Background rejection growing radius	
(width	=	5.)	Profile centering width	(i)
(radius	=	10.)	Profile centering radius	(i)
(threshold	=	0.)	Detection threshold for profile centering	
nfind	=		Number of apertures to be found automatically	usually 1
(minsep	=	5.)	Minimum separation between spectra	obsolete, if nfind=1
(maxsep	=	1000.)	Maximum separation between spectra	obsolete, if nfind=1

a: Empty (but object image if tracing the lamp)
b: Empty (but object image if tracing the lamp)
c: If continuum is weak put pixel number of some strong emission lines here
d: If spectrum noisy increase this to e.g. 30
e: Examine the spectrum with display first and decide about this
f: 1 means subtraction of a costant. This is usually OK.
g: Never trust what you put here, but always examine it interactively with b
h: –3 means median of 3 points in the same line: should be OK
i: Should be compatible with profile width

Table 5: *(continued)*

(order	=	”increasing”)	Order of apertures	
(aprecenter	=	””)	Apertures for recentering calculation	
(npeaks	=	INDEF)	Select brightest peaks	
(shift	=	yes)	Use average shift instead of recentering?	
(llimit	=	INDEF)	Lower aperture limit relative to center	(l)
(ulimit	=	INDEF)	Upper aperture limit relative to center	(m)
(ylevel	=	0.1)	Fraction of peak or intensity for automatic width	
(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	=	no)	Average limits over all apertures?	
(t_nsum	=	10)	Number of dispersion lines to sum	(n)
(t_step	=	10)	Tracing step	(p)
(t_nlost	=	3)	Number of consecutive times profile may be lost	(q)
(t_function	=	”legendre”)	Trace fitting function	
(t_order	=	2)	Trace fitting function order	(r)
(t_sample	=	”*”)	Trace sample regions	(s)
(t_naverage	=	1)	Trace average or median	
(t_niterate	=	0)	Trace rejection iterations	1
(t_low_reject	=	3.)	Trace lower rejection sigma	
(t_high_rejec	=	3.)	Trace upper rejection sigma	
(t_grow	=	0.)	Trace rejection growing radius	
(background	=	”median”)	Background to subtract	(t)
(skybox	=	1)	Box car smoothing length for sky	
(weights	=	”none”)	Extraction weights (none—variance)	variance (u)
(pfit	=	”fit1d”)	Profile fitting type (fit1d—fit2d)	
(clean	=	no)	Detect and replace bad pixels?	
(saturation	=	INDEF)	Saturation level	32657 (v)
(readnoise	=	”0.”)	Read out noise sigma (photons)	4.2 (z)
(gain	=	”1.”)	Photon gain (photons/data number)	1.2 (z)
(lsigma	=	4.)	Lower rejection threshold	
(usigma	=	4.)	Upper rejection threshold	
(nsubaps	=	1)	Number of subapertures per aperture	
(mode	=	”ql”)		

l: Always examine interactively, hit [l] key to change
m: Always examine interactively, hit [u] key to change
n: Increase to e.g. 30 if noisy tracing
p: Keep similar value to the one of t_nsum
q: Increasing this does not help with noisy spectra; increase t_nsum and t_nstep instead
r: 2 means straight line, if this is too low try :order 4 [ENTER] + [f] + [r]
s: Put e.g. 1:1000 if pixels above 1000 are overscan
t: *none* for comparison spectra and *median* for stars
u: Use *none* for lamp spectra and very bright stars, and *variance* for all other
v: Put proper value here!
z: Put proper value here (important if using variance weights)!

7 Wavelength calibration

Now it's time to calibrate the pixel scale of our spectra into a wavelength scale. The results will be written by IRAF as a polynomial function in the spectrum header (and copied to the database to a text file called like *database/ide0001.0001*). Such a polynomial function will be read and applied to the spectrum when a wavelength scale will be requested by an IRAF command (like *splot* to show the spectrum or *listpix* to write the spectrum to an ASCII file).

7.1 Finding wavelength solution of the first calibration spectrum

First add up the columns of the comparison spectrum image corresponding to those of the stellar spectrum on the science image. If the science image is *g0020*, the corresponding comparison spectrum is *g0021*, and the stellar spectrum runs over columns 128 to 132, the sum will be

blkavg g0021[128:132,] g0021_1D*

with *g0021_1D* being the name of the output file. An easier and better alternative is to use the aperture tracing of the corresponding scientific frame and apply it to the calibration spectrum using *apall* once again. Type the name of scientific frame for *references* and *profiles* parameters, turn off parameters from *interactive* to *review* (the only exception is *extract=yes*). Finally turn off the background subtraction with *background=no*. Using apall with these parameters will produce a 1-D comparison spectrum by summing exactly the same pixels as it was done for the corresponding exposure of the star (= scientific frame). This is more accurate than the *blkavg* command above because it takes into account for any curvature or distortions of the stellar spectrum.

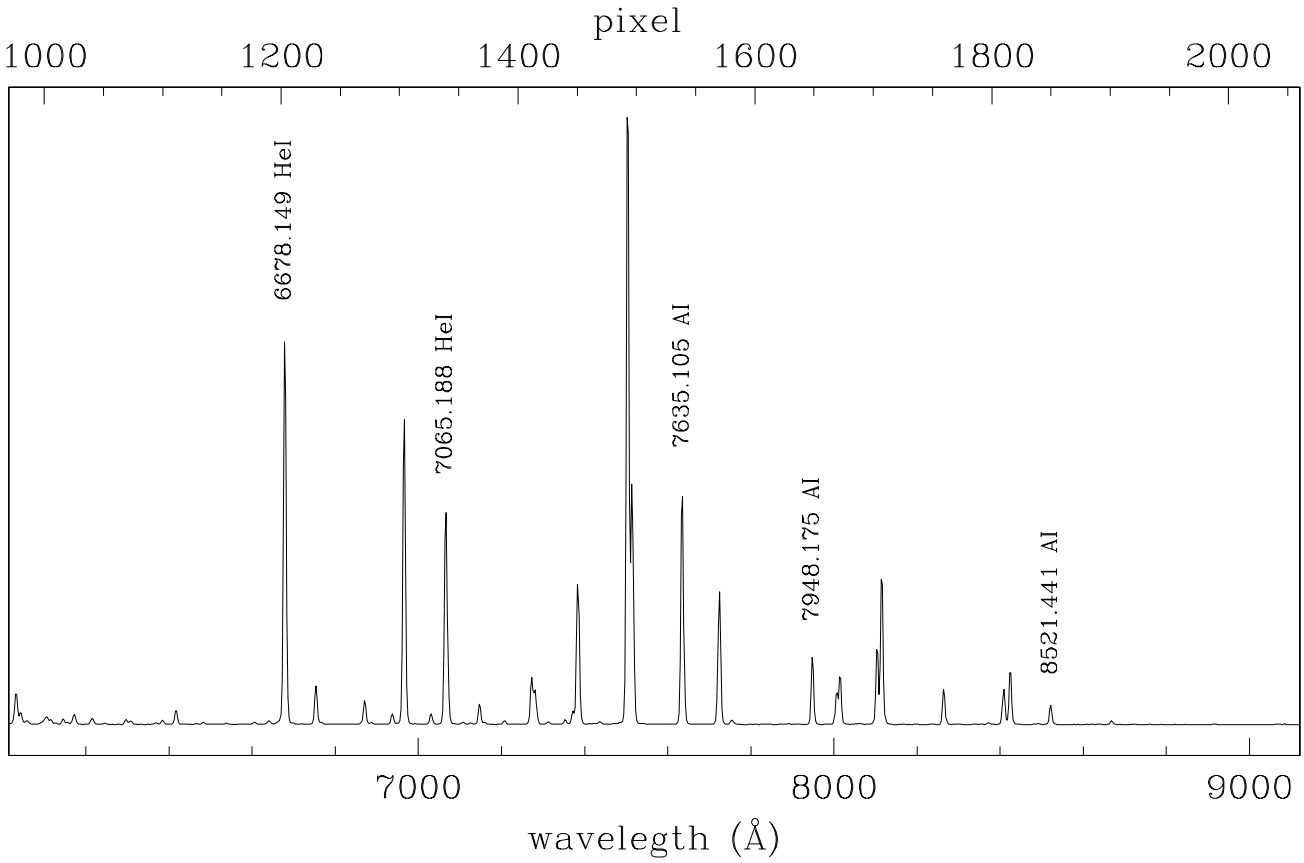
The idea is now to manually identify a few lines in the spectrum. This gives IRAF a first idea of the calibration function. Then you will tell IRAF which sort of lamp produced the comparison spectrum (for example an Iron or Helium-Neon-Argon lamp) and ask it to proceed automatically to identify - on the base of the preliminary calibration - all the other lines. You will check and perhaps modify these additional identifications and finally you will tell IRAF to proceed with the fitting of the wavelength solution using all validated comparison lines. Let's start with the identification

identify g0021_1D

Three parameters have to be set appropriately (use *epar identify*):

ftype (emission | absorption) For a standard comparison spectrum the lines are in emission, but there are cases in which you may be forced to use another spectrum with absorption lines (for example that of a nearby star, or the telluric absorptions)

fwidth the approximate total width (in pixels) of the lines (generally 2-5 pixels). This greatly helps IRAF to properly fit the desired lines, particularly in crowded spectra (estimate it by plotting the spectrum with *splot g0021_1D*)



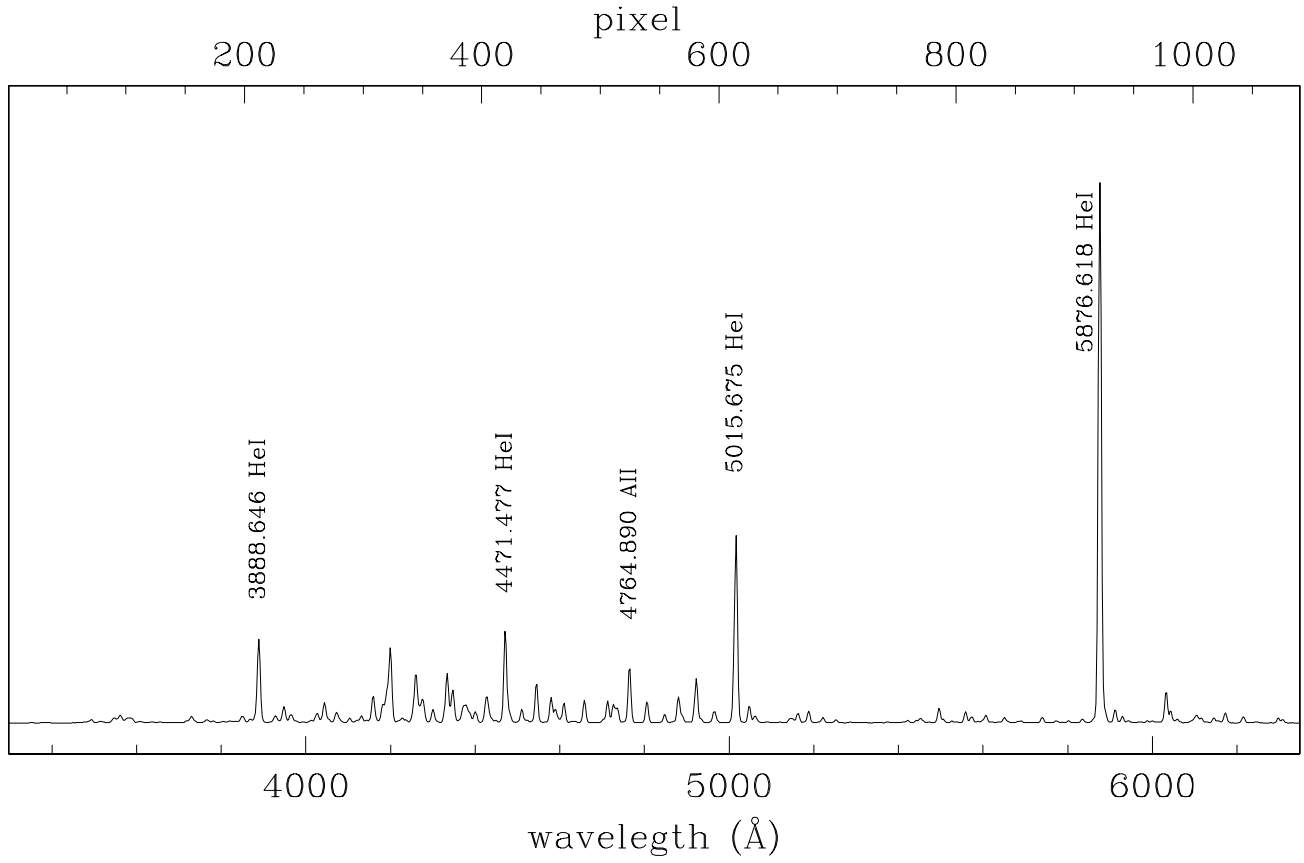


Figure 10: *Some identifications of the comparison spectrum that comes with these notes.*

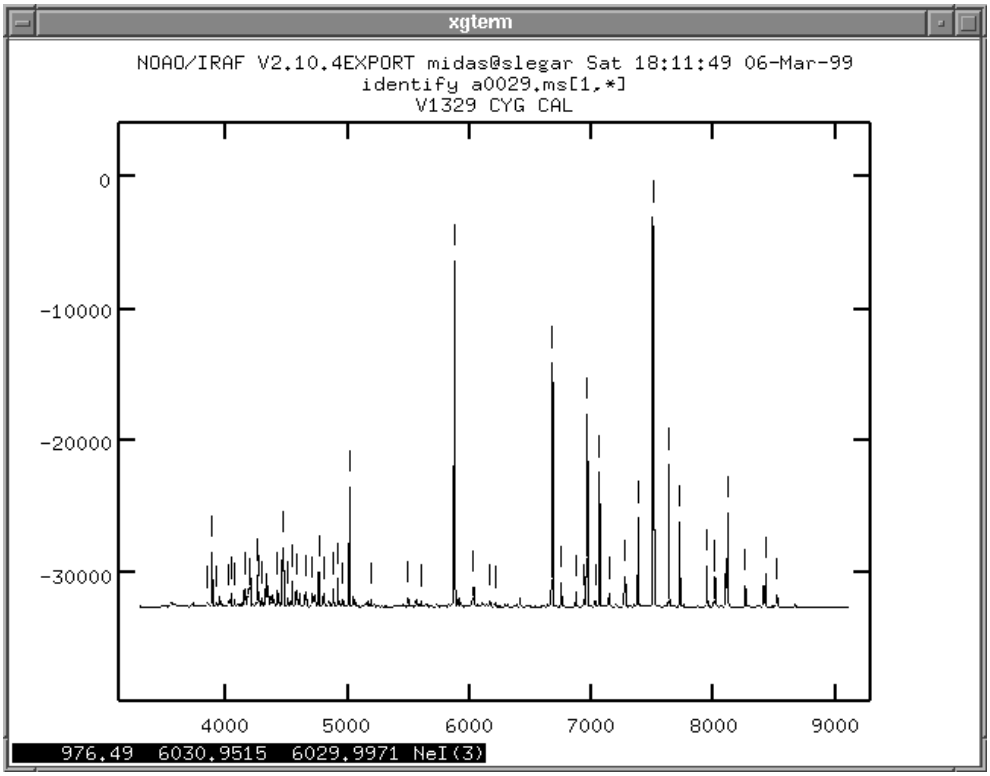


Figure 11: *Automatic identification of lines in the comparison spectrum by task* **identify**.

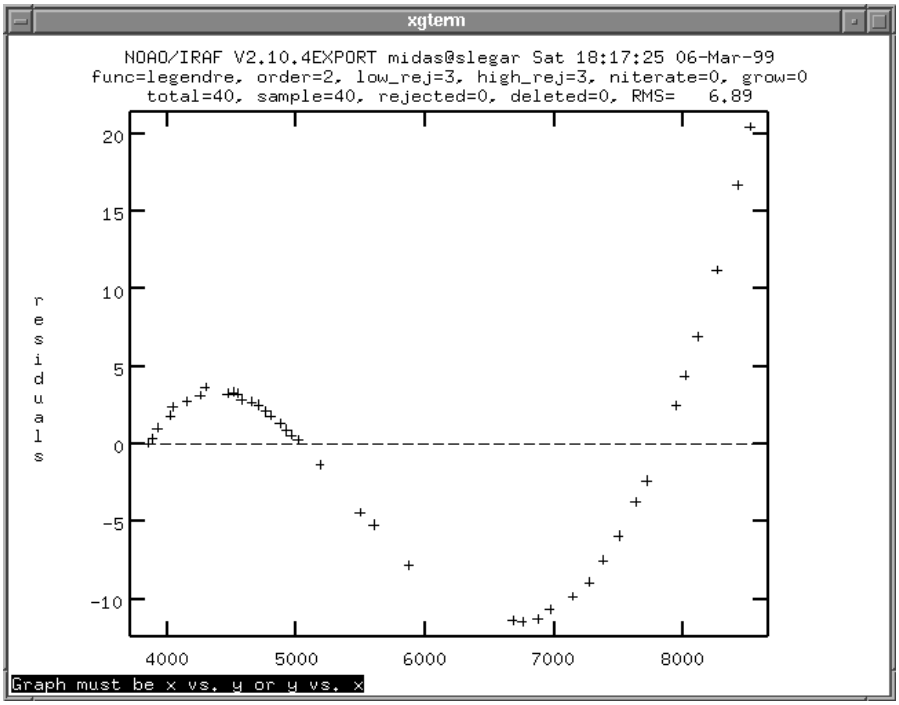


Figure 12: *Fitting the dispersion curve. In this case an order 2 is clearly too low for the Legendre fitting function and must be increased to appropriately fit the dispersion curve over the whole spectrum (ordinates and abscissae in Å).*

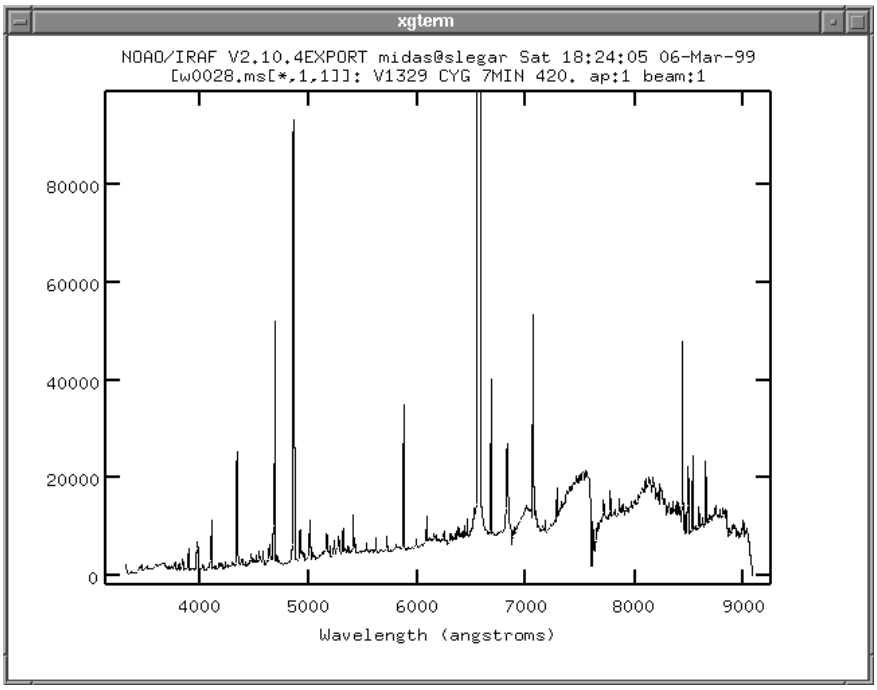


Figure 13: *The science spectrum calibrated in wavelength via the* **dispcor** *task. The spectrum has been plot by task* **splot** *and the Hα has been truncated (using the windowing sequence* w+e+e*) to emphasize visibility of the continuum and weak emission features (abscissae in Å and ordinates in counts).*

coordli the file where IRAF has to look for the whole list of lines for that type of comparison lamp. To see the available files type

page linelists$README

and you should get information on files like henear.dat, fear.dat, cuar.dat, skylines.dat, thar.dat ... Choose the most appropriate one[10] (*henear.dat* for the demo images coming with these notes).

Now by hitting m mark some lines and type their wavelengths (if it doesn't take it, position the cursor a bit to the right!). You should input several lines, e.g. 10, which span well the wavelength range. If needed zoom a portion of the spectrum by w e e (hit e at diagonal corners of desired viewing cut) before marking lines (use w m w n to de-zoom it back). Fig. 8 shows an expanded plot of a comparison spectrum from the set of demo images with identification of some of the strongest lines.

Next use f to fit a preliminary solution. If needed change the order of the fitting function typing e.g. *:order 4* *ENTER* . Delete any points by d. Return to marking by q and add more line identifications with m, or delete identification nearest the cursor by pressing d. You can also zoom on the feature nearest the cursor by pressing z, and move to the next (previous) one by pressing + (or −); p un-zooms you

[10]When observing at the telescope you should write down the type of lamp you are using and photocopy a sample identified lamp spectrum from the observatory manual.

back, and [r] redraws the graph. [a][c] re-centers all lines already identified. Avoid features in blended areas, or the ones with asymmetric bases!

When you are happy with the preliminary solution you may decide to include all other lines from the database. Do this by pressing [l], and re-fit the solution using all the identified features using [f].

When happy exit with [q]. It automatically write solution to the database. Actually it is a good idea to slowly build a solution. Each time you successfully identify few more lines you exit the task and so write the solution to the database; and the next time you start over the computer learns the work done so far from the database anyway. This philosophy saves you disappointments like painfully identifying many lines but then spoiling it all with an obsolete set of lines from a wrong line list.

And what if you do not have any calibration lamp spectra? There are two ways out though both provide only a very provisional wavelength calibration.

- You may use lines of some normal or emission–line star. A useful lists of stellar lines are Moore's tables that should be available at your observatory. You can prepare a small private line-list by simply typing the wavelengths one per line and then refer to this filename by *coordli* keyword of the *identify* task. Note that here we are neglecting any radial velocity of the calibration star (unless you get it from the literature and Doppler shift the typed wavelengths accordingly.)

- Emission of the Earth's atmosphere can be used. These lines have (almost by definition) zero radial velocity, but they are unevenly distributed in the optical and near-IR range. A nice identification of night sky lines appears in Osterbrock et al. (1996, PASP, 108, 277). In spectra extracted with *apall* the sky is in band 4 (if variance extraction was used) or band 2 (if it was not used). Sky lines can be useful to check the accuracy of your wavelength solution even if it seems already quite robust: prompt to use is the NaI D doublet (5889.951 and 5895.924 Å, from street lamps), the [OI] lines (5577.35, 6300.23 and 6363.88 Å) and Hg I 4358.343 Å again from street lamps).

7.2 Identify other calibration exposures

After you spent much time determining the wavelength solution of the first calibration image (*g0021_1D* in the example above) you may identify a second one (suppose the file to be *g0036_1D*) by typing

reidentify g0036_1D referenc=g0021_1D

This task uses your solution but allows for simple shifts in the spectrum (set the parameters accordingly with *epar reidentify*) that are due to e.g. spectrograph differential flexures when observing in different directions and/or small changes on the focal plane geometry induced by changes in the ambient temperature.

The correctness of the wavelength solution for g0036_1D can be checked by entering the *identify* task for that image and leaving the task *without* writing the results to the **database**.

7.3 Apply wavelength calibration to science data

Finally the wavelength solution has to be applied to a specific science spectrum. To tell IRAF which wavelength solution to use, you should edit the image header and insert the appropriate instruction. For the above *g0020* and *g0021* science and calibration spectra, it will be

hedit g0020.0001 refspec1 g0021_1D add+ ver−

which writes in the header of the image *g0020.0001* that its reference spectrum for wavelength calibration is the image *g0021_1D*[11]. Let's now calibrate the science spectrum in wavelength:

dispcor g0020.0001 w0020 linearize=no

where *w0020* is the wavelength calibrated science spectrum. Setting *linearize=no* means that you want each pixel to be associated with its exact wavelength. Alternatively *linearize=yes* produces a spectrum with pixels equally spaced in wavelength (easier to understand but generally offering a less accurate representation of the true spectrum).

8 Flux calibration

An accurate flux calibration of spectra is a difficult task as even tiny cirrus clouds can spoil the result. So remember to always secure spectra of standard stars at different zenith distances and to rotate the slit perpendicular to the horizon (avoiding differential refraction).

We will use the task *observatory* to determine observatory parameters, *standard* to flux calibrate each standard star, and *sensfunc* to finally determine the wavelength response. The solution will be applied to the spectra by the *calibrate* task.

To perform the calibration into fluxes IRAF needs to know all kinds of times and coordinates. And it wants them in a format that it can recognize. Images at the telescopes are written by different software packages, which means image headers written in different styles. For example, some ESO keywords are formatted to comply with the MIDAS reduction package and not with IRAF. In the majority of observatories the images are however written in strict accordance with IRAF prescripts. The demo images that come with these notes are taken at ESO, so we have to perform some manipulations of the header to make it understandable to IRAF and we have also to insert some missing information.

As this is quite complicated a simple procedure for data taken at ESO has been prepared (like the demo images coming with these notes. The procedure can be easily adapted to any other observatory). The procedure (named *eso.set*) is given in Table 6. Before starting make a list of all wavelength calibrated files and write it to a file named *a.lst*

files w.imh > a.lst*

[11]Instead of direct editing of image header the *refspectra* command can be used that includes also many options for spectra interpolation. See *help refspectra*.

```
                                                  I R A F
                                       Image Reduction and Analysis Facility
PACKAGE = onedspec
   TASK = standard

input   =                         Input image file root name
output  =                  std  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=  3.6800000000000E-20) Absolute flux zero point
(extinct= onedstds$/ctioextinct.dat) Extinction file
(caldir = onedstds$/spec50cal/) Directory containing calibration data
(observa=                  eso) Observatory for data
(interac=                  yes) Graphic interaction to define new bandpasses
(graphic=              stdgraph) Graphics output device
(cursor =                    ) Graphics cursor input
star_nam=              feige67  Star name in calibration list
airmass =                       Airmass
exptime =                       Exposure time (seconds)
answer  =                  yes  (no|yes|NO|YES|NO!|YES!)
(mode   =                   ql)
```

Table 6: *Parameters of* **standard** *task for the star Feige 67*

The procedure that set appropriately the image headers is invoked by

cl < *eso.set*

IRAF V2.11 includes tasks that can make the above calculation more elegant. First type

astutil

to load the noao.astutil package. If the wavelength calibrated files are listed in the file *a.lst*, the command to compute the airmass, exposure time, etc. is

asthedit *@a.lst* *eso.dat*

where *eso.dat* is a file of instructions given in Table 7.

8.1 The task standard

We are about to flux calibrate the spectrum. The reference will be spectra of some standard stars observed during the same night. The aim is to calibrate the CCD chip response, spectrograph+telescope throughput and allow for atmospheric extinction. The result is a spectrum as observed from outside the atmosphere with an ideal uniformly sensitive detector+telescope+spectrograph. In short, you (*a*) take from a tabular compilation the energy distribution of the standard star you observed, (*b*) correct this energy distribution for wavelength-dependent atmospheric extinction, (*c*)

Table 7: *Command procedure* **eso.set** *to calculate observatory parameters, JD and airmass from values stored in the image header of an ESO spectrum like those coming with these notes.*

```
# This is a task to calculate JD and airmass for observations at ESO 1.5-m
# File a.lst should contain the list of images.
hedit @a.lst date-obs '(@"DATE-OBS")' add+ ver-
hedit @a.lst UT '(@"TM-START"/3600)' add+ ver-
hedit @a.lst EPOCH '1994.8' add+ ver-
hedit @a.lst RA '(@"POSTN-RA"/15)' add+ ver-
hedit @a.lst DEC '(@"POSTN-DE")' add+ ver-
hedit @a.lst exptime '(@"TM-END"-@"TM-START")' add+ ver-
noao
observatory set eso
astutil
setjd @a.lst
# eso is 4.7153 hours west of Greenwich
hedit @a.lst st '(6.6974-(2451543.5-@"ljd")/15.21842447-4.7153+@"UT"*1.002738)' add+ ver-
# sidereal time may be bigger than 24 hours, so we subtract days first.
hedit @a.lst st '(@"st"-24.0*int(@"st"/24))' ver-
# if sidereal time is negative we add additional 24 hours
hedit @a.lst st '((@"st">0.0) ? (@"st") : (@"st"+24.0))' ver-
# now comes the hour angle
hedit @a.lst ha '(@"st"-@"RA")' add+ ver-
# eso is at fi=-29.2566 deg, so sin(fi)=-0.488721 and cos(fi)=0.872440
hedit @a.lst sinh '(-0.488721*sin(@"DEC")+0.872440*cos(@"DEC")*cos(@"ha"*15))' add+ ver-
# and finally the elevation above horizon
hedit @a.lst h '(asin(@"sinh"))' add+ ver-
# and the airmass.
hedit @a.lst airmass '(1/@"sinh")' add+ ver-
hedit @a.lst sinh . del+ ver-
```

compare it to the energy distribution of the observed spectrum, (*d*) derive from such a comparison the function that gives the response of your system for every wavelength.

The task *standard* determines calibration pass-bands and writes them (together with data on extinction and airmass) to a file called *std*. If this file exists before you use *standard* for the first time, delete it. You will run *standard* once for each exposure of a standard star. The results will be appended to the *std* file. You will have to know where the input extinction and flux calibration files are. To find them first change your directory typing

cd onedstds$

and look for the extinction file with *dir*. Usually for ESO you will use *ctioextinct.dat* (unless you prepared a custom file for it). Then find where the file with fluxes for your

Table 8: *Contents of the file eso.dat used by the asthedit command to manipulate demo images obtained at ESO.*

```
observat = "eso"
ut = sexstr ((@'tm-start'+0.1) / 3600.)
utend = sexstr ((@'tm-end'+0.1) / 3600.)
epoch = epoch (@'date-obs', ut)
st = mst (@'date-obs', ut, obsdb (observat, "longitude"))
exptime = (utend>ut)?(utend-ut)*3600.:(utend+24-ut)*3600.
ra = sexstr (@'postn-ra' / 15)
dec = sexstr (@'postn-dec')
airmass = airmass (ra, dec, st, obsdb (observat, "latitude"))
```

star is (cf. Figure 13). A good try for southern hemisphere may be

cd ctionewcal

Write down the paths and filenames and return to your home directory typing *cd home$*, then go to the sub-directory you are working in. Before running the *standard* command change its parameters using *epar standard* and write the paths and names of the extinction and flux files. Typically the extinction file would be written as *onedstds$/ctioextinct.dat* and the caldir and calib would be *onedstds$/ctionewcal/* and e.g. *l9239* (for the standard star LTT9239 in the example images that come with these notes). An example of *standard* parameters for the standard star Feige 67 in our demo images (cf Table 1) is given in Table 8. An example of the flux file for the standard star LTT 9239 is given in Figure 13.

Now it is time to finally run (w0037 is the extracted and wavelength calibrated file for standard star LTT 9239 as from Table 1)

standard w0037

Always review the passbands interactively. Use [d] to delete a pass-band under cursor, [a][a] to mark corners of a new passband, and [r] to redraw the graph. [w][e][e] expands graph between given corners and the commands [w][m] [w][n] de-window it back. It may happen that sensitivity of the CCD drops around a given wavelength. Be sure to 'paint' such minimum adding many [a][a] passbands or the flux calibration will be no good there!

Note that *std* is a usual text file so you may use the editor to view/change it. This is good to know if you already run standard on several good stars, but the last turns out to be particularly dumb: edit the *std* file and delete the part belonging to the star you don't want. An example of the beginning of a *std* file for the same flux standard as in Figure 13 is given in Figure 14.

8.2 The task sensfunc

Standard task recorded response of each standard star. Now it is time to put the results

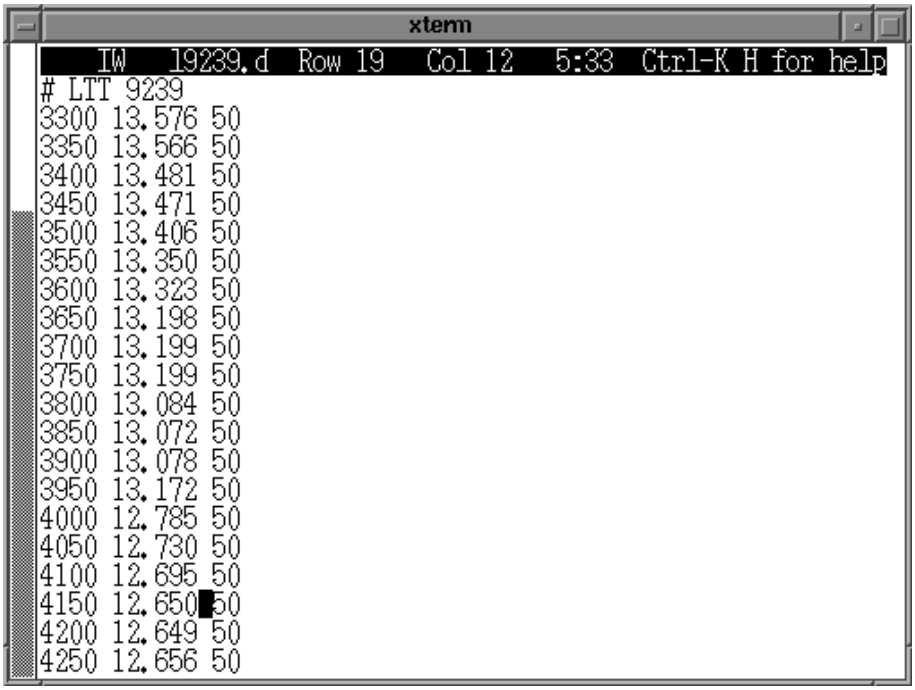


Figure 14: *The beginning of the flux file for the standard star LTT 9239 (resident in onedstds$/ctionewcal/). The first column is the central wavelength, the second is the flux (expressed as −log of the flux in erg cm^{-2} sec^{-1} Å^{-1}). The last column is the width of the λ−interval over which to sum around the central wavelength to get the given flux.*

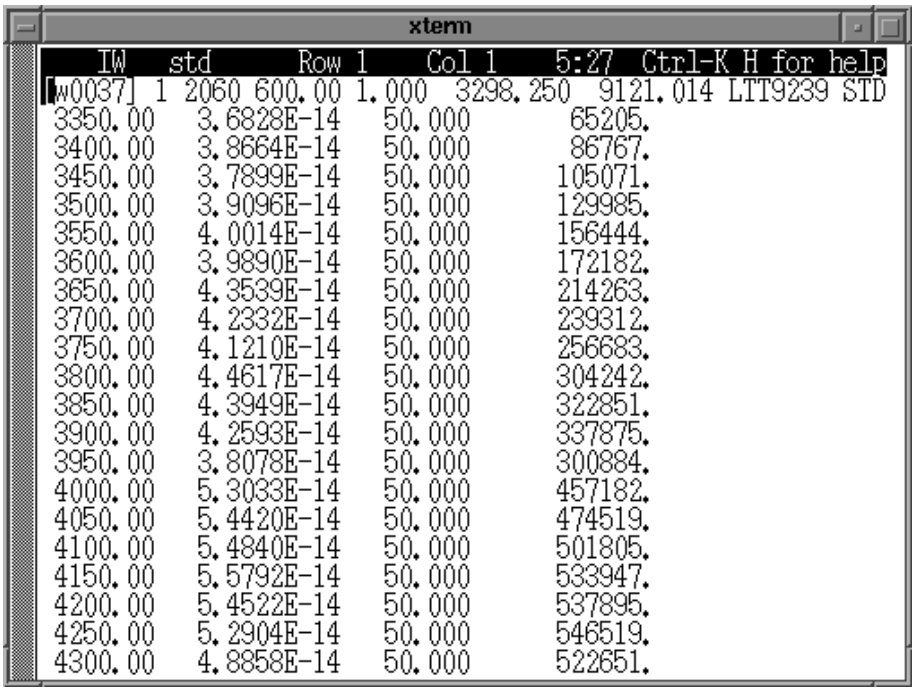


Figure 15: *The beginning of a std file for the standard star LTT 9239. The first three columns are the same as in Figure 13, the last column are counts summed over the 50 Å range centered at the λ from the first column.*

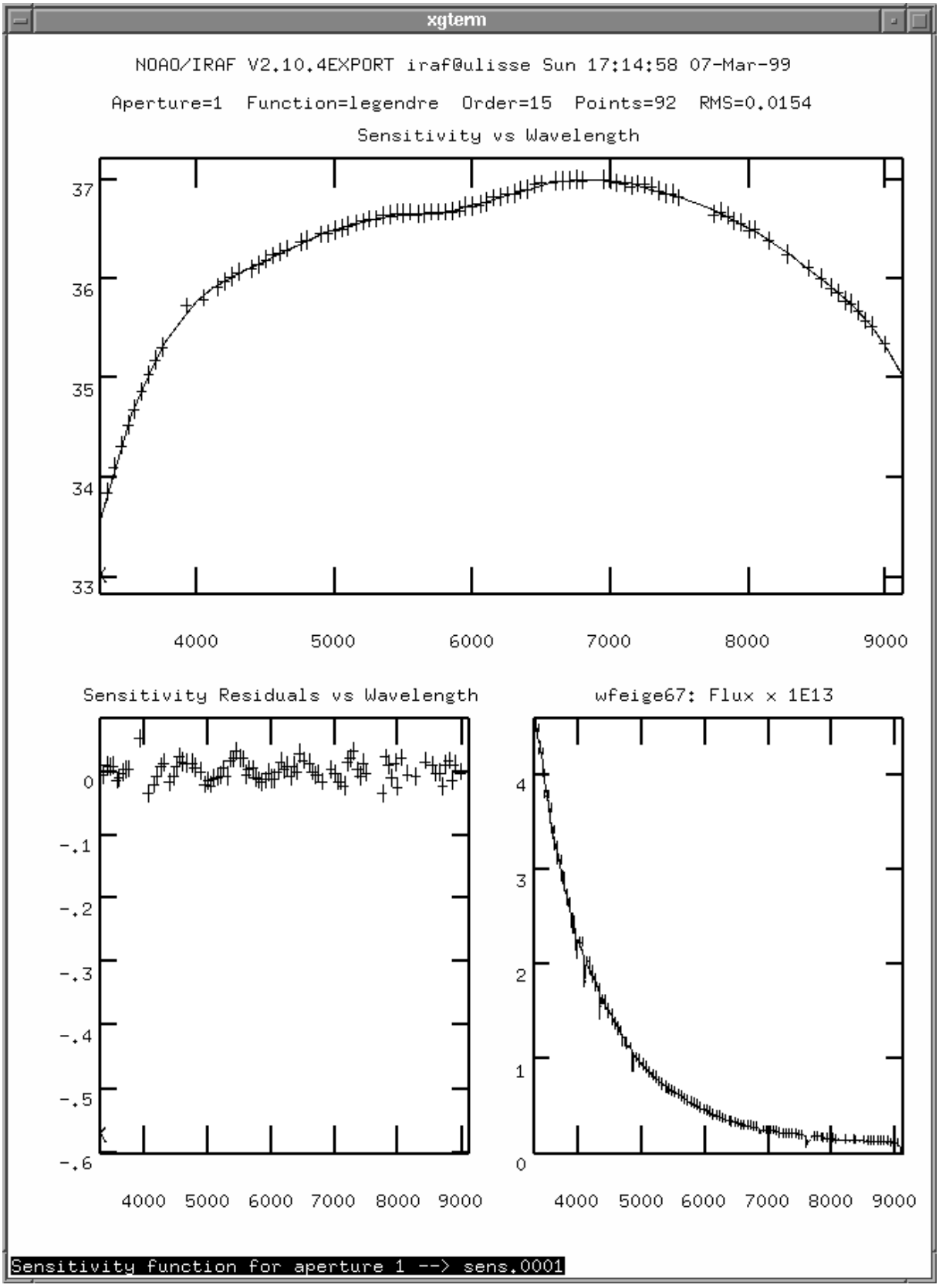


Figure 16: *Graphic output of the sensfunc task for the combination* **:g sri** *on a single standard star.*

together and find a proper wavelength dependence of instrumental sensitivity and atmosphere transparency. This is done with the *sensfunc* task. It creates an image with a default name *sens.0001* so if it exists delete it with *imdel sens.0001* before running *sensfunc*. You may use a single standard star and adopt an extinction function, or you may combine observations of several standard stars observed at different airmasses to solve simultaneously for the sensitivity function and atmospheric extinction.

IRAF needs to have some general idea of atmospheric extinction before to start. So use e.g. *extinct = onedstds$/ctioextinct.dat*. Putting *newexti = myextinct.dat* will store the revised extinction file there.

Always run the task interactively; for *graphs* option use the *srei* string. This plots 4 graphs: sensitivity function, residuals of the current fit, extinction function, and flux-calibrated spectrum. Vertical axes are in magnitudes (with an arbitrary zero point).

As usual you may add calibration points to the graph by pressing [a], re-fit the solution by [f], and re-draw it by [r]. Delete is, however, a bit more complicated: after you press [d] it asks you whether you want to delete a point nearest the cursor, all points at the nearest wavelength, or possibly all data of the star with a point nearest the cursor. Similarly [u] undeletes the corresponding point(s). Hitting [i] gives info on the point nearest the cursor.

More generally, [s] shifts all stars so that they have the same average sensitivity as the star with the highest sensitivity curve (an useful option if some stars were observed through thin clouds with a grey opacity law). Pressing [c] combines all pass-bands with the same wavelength and replaces them with a single point with average sensitivity.

Most importantly [o] brings you back to the original situation, therefore un-doing any delete, shift or combine operations you did so far.

If you feel the extinction function is not satisfactory, you may enter the extinction curve construction task by pressing [e]. Use [?] to get help there. Exit with [q]. Remember that extinction should be always a decreasing function of wavelength, i.e. red photons pass the atmosphere easier than the blue ones.

Note that the function used to fit the instrumental response will be usually of very high order. A good idea is to use *spline3* fitting (*:function spline3*) with some 20 pieces, i.e. (*:order 20*).

Finally [q] exists the *sensfunc* task and writes the *sens.0001* image.

8.3 The task calibrate

All calculations concerning flux calibration have been done at this point. What remains is to finally apply the solution to each star to be calibrated. Use *epar calibrate* to set the appropriate extinction table: *extinct= onedstds$/ctioextinct.dat* or *extinct= myextinct.dat*.

If *list_w* contains the list of wavelength calibrated spectra which had their airmass and other parameters appropriately set by the *eso.set* procedure (or the *asthedit* command in the more recent IRAF v2.11) and *list_f* is the same list with the names changed to the desired names of the output flux-calibrated files, you would just type

calibrate @list_w @list_f

8.4 Calibrating flux without standard star spectra

This should be avoided, as a good flux calibration is difficult to obtain even with observations of many standard stars. But it may still happen you end up with observations of a star for which you know its spectral type but no more.

A fast but very rough possibility is to use its tabulated V magnitude and assume it is a blackbody of a given effective temperature. The first step is to create an artificial spectrum of a blackbody with the *noao.artdata.mk1dspec* command:

*imarith w-star * 0.0 bbody*

mk1dspec bbody continu=1.0 tempera=10000. lines=0

where *w-star* is a wavelength calibrated spectrum of a 10.000 K star. Division of *w-star* with *bbody* now gives an approximation to a sensitivity function (*sens.0001*) mentioned above. The resulting spectrum will be calibrated in colour but not in absolute flux. The last step is to calculate the difference in V magnitude between tabulated data and the resulting spectrum and so calibrate the absolute flux. Note that here we neglected the departure of the true stellar energy distribution from that of a blackbody (not tremendous if dealing with O-type standard stars longward of the Balmer jump) or the influence of atmospheric extinction.

An alternative possibility would be to use standard procedure for flux calibration. You fake the *standard* task by using standard star with the same spectral type as your object. Still bear in mind that here you neglect any differences in line strength (so avoid regions with spectral lines in calibration intervals!) and reddening. And of course the flux should be corrected for the difference in V magnitude of observed star and the fake standard.

9 Additional corrections

Often flux calibrated spectrum is a final result that needs to be measured and published. But sometimes one would like to remove the influence of Earth motion around the Sun, of interstellar reddening or sky absorptions. There are three tasks in the *onedspec* package useful for this purpose. Here we give only a brief example of their use. More can be found in the corresponding help files.

9.1 Heliocentric correction

To remove Doppler shift due to Earth's motion from flux calibrated spectrum *f0020* (that was prepared with the script *eso.set* or with the *asthedit* command – see Section 8) type

rvcorrect ima=f0020 imupdate+ observa=eso

dopcor f0020 h0020 –vhelio isvel+

and obtain the spectrum *h0020* as would be observed from the Solar system barycenter. Note the minus sign before *vhelio*.

9.2 Dereddening

If the spectrum *h0020* was observed through an interstellar cloud with *E(B-V)*= 0.5 its intrinsic spectrum *i0020* would be obtained with

deredden h0020 i0020 0.5 type="E(B-V)"

A good reference for reddening vs. distance in the region close to the galactic plane is Neckel et al. (1980, A&AS, 42, 251), while galactic reddening far from the plane can be found from http://adc.gsfc.nasa.gov/adc/ (Burstein+ gif images).

9.3 Removing sky lines

The spectra were flux calibrated to remove broad continuum absorption due to Earth's atmosphere. However telluric (i.e. atmospheric) absorption bands such as the one between 7600 and 7750Å remain. Usually this does not bother you. Still sometimes the lines of interest (for example the near-IR K I doublet) fall within telluric bands. So you wish to cancel them out. To do that use the task *telluric* within the *onedspec* package. It is not an easy job, still the help file discusses all the details. By now you should have enough experience to learn it yourself. Enjoy!

10 Plotting, measuring and exporting reduced spectra

To plot and measure parameters (like equivalent width, Gaussian fitted wavelength, FWHM, etc.) of various line features in reduced spectra use the *splot* task (it is in *noao.onedspec* package)

splot i0020

Click k on the continuum at both sides of a line to get the equivalent width, width, flux and position of a Gaussian fit to the line. Pressing m on both edges of a wavelength range gives an average, sigma and S/N ratio there. Zooms work as usual (w e e expands graph between given corners and the commands w m w n de-window it back). Check help for many other interactive commands of the *splot* task.

Magnitudes and colors can be derived from your flux calibrated spectra by the *sbands* command and the continuum is normalized by the *continuum* task.

To plot the pixel wavelengths and intensities in an ASCII table use *listpix*. The *wcs=world* option prints wavelengths instead of pixel coordinates. For nicer output you may append a proper format string (optional, see *help listpix*). A simple example would be

listpix f0020 wcs=world format="%12.8e %3.1d %10.3d" > f0020.dat

Replacing the single file name *f0020* with a list of file names is not recommended as *all* the spectra from the list will be written to a single file. It is better to write all *listpix* commands to a file (let us call it *2ascii.txt*) and then execute them with

cl < 2ascii.txt

11 Summary of Most Common Commands

11.1 Options on the command line

- Options included on the command line are used instead of the ones written in the command's parameter file, e.g.
  display e025 zrange- zscale- z1=100 z2=800

- Permanently change the options with *epar* [Exit with CTRL-D], e.g.:
  epar display

- List parameters with *lpar*:
  lpar display

- Write any output to a file instead of the screen with "> filename" included on the end of the command line:
  lpar display > display.prs
  files e*.imh > listing.lst
  help display > display.hlp

- Add any screen output to the end of an existing file by including ">> filename" on the end of the command line:
  files f*.imh >> listing.lst

- Print any screen output to the printer by including "| lprint" on the end of the command line (Unix systems only):
  help display | lprint

- Execute a sequence of commands written in an ascii file myfile.com:
  cl < myfile.com

11.2 Commands in any graphic display window

? — help page ([space] to scroll, q and [return] to exit)

= — print graph to printer

:.snap epsfl — save graph to a postscript file sgi???.eps

r — redraw graph

I — interrupt task immediately (sometimes it works, often not...)

w e e — zoom graph with e-s as corners

w m — de-zoom taking the whole range for x-axis

w n — de-zoom taking the whole range for y-axis

q — exit task

11.3 Additional commands while fitting a function

f — re-fit function [possibly with "r" to re-draw graph]

d — delete point nearest the cursor

u — un-delete point nearest the cursor

a — add point at cursor [usually you are asked how many points it is worth: type number and RETURN]

:o 15 — changes function's number of coefficients to 15 [in the case of spline3 this is the number of pieces along the graph]

:f spline3 — change function type to cubic spline [other choices: chebyshev, legendre, spline1]

11.4 Command sequence after flat-fielding

Go to: noao.twodspec.apextract.

11.4.1 apall

- Aperture definition:

:show — show parameters

:low -3 — put left aperture edge 3 pixels left from center

:up 5 — put right aperture edge 5 pixels right from center

c — re-center aperture

b — review background range(s):

z — delete background range nearest the cursor

s s — define new background range with "s" positions for edges

f r — fit and redraw solution

q — quit background task

- Tracing (with usual commands for a fitting package)

11.4.2 identify

Go to: noao.onedspec.

m — mark center of a feature close to the cursor

z — zoom on the closest marked feature ["+", "-" to move zoom to features left and right, "p" to de-zoom back]

f — start the fitting package [use usual fitting commands there]

l — try to add many features from the list [don't use this before your solution with manually identified features is not good enough!]

11.4.3 reidentify

Reidentify other lamp spectra [use it non-interactively, but with refitting = yes option].

11.4.4 tell the reference spectrum

hedit e025.0001 refspec1 e026.0001 add+ ver–
tells that the lamp spectrum of the object e025.0001 is called e026.0001.

11.4.5 dispcor

Applies wavelength solution [use "linearize=no" option to preserve pixel identities]

11.4.6 airmass and exptime definition

IRAF needs airmass and exposure time for flux calibration: use
imhead e025.0001 l+
to type all header data, or
hedit e025.0001 airmass,exptime .
to type airmass and exposure time fields for your image(s). You can add these two fields manually by, e.g.
hedit e025.0001 airmass 1.52 add+ ver–
hedit e025.0001 exptime 1200 add+ ver–
or you may use an automatic procedure like *eso.set* described in the main text.

11.4.7 standard

Add all standard stars in turn. Use

a a — to add a wavelength bin

d — to delete a bin

The results are written in an ascii file "std": edit, paste, delete at will!

11.4.8 sensfunc

Use with "srei" option to describe which windows to plot.
Use the standard fitting commands, and

d — to delete something [it will ask you if you want to delete point, wavelength, or whole star]

i — to get info [which star a point belongs to]

s — to shift mean sensitivities of all stars to the same value [use for cloudy nights: effectively you assume a grey opacity law]

o — to return to the original (starting) situation

e — to enter extinction curve calculation [good luck: this is a standard fitting package, use "q" to quit.]

11.4.9 calibrate

Apply flux calibration.

11.4.10 splot

Use the ":histogram yes" option to plot histo-style plots.
Use usual display manipulation commands, plus:

k k — to fit a single Gaussian (mark wings)

d d — to fit blended Gaussians (mark edges, use "m" to mark guesses for centers of Gaussians, "q" to start fitting [answer questions, "q"s to exit])

m m — calculate statistics of a region (SN ratio, ...)

o g — overplot another spectrum

s — smooth the spectrum

f — to start arithmetic manipulation of the spectrum ("l" to change "y" axis to its logarithm, "q" to exit)

i — save present spectrum to a file

% — change beam (you are asked for its number, e.g. beam 3 is sky if you use variance weighting)

11.4.11 listpix

To write spectrum to an ascii file. you may forget about format option and manipulate the results with SuperMongo instead.
listpix f025.0001 > f025.txt
The form
listpix @f.lst > spectra.asc
writes *all* spectra from f.lst to a single ascii file *spectra.asc*.