# IRAF notes for Observational Astrophysics I

Søren S. Larsen

March 31, 2011

# Contents

# Chapter 1

# Introduction to IRAF

The handling of astronomical datasets requires specialised software. Two general-purpose packages in common use are MIDAS (the Munich Image Data Analysis System), developed by the European Southern Observatory, and IRAF (the Image Reduction and Analysis Facility), developed by the National Optical Astronomy Observatories in the US. In addition, many instruments are now so complex that dedicated *pipelines* are available for the reduction of the observations. However, once the initial pipeline processing has been done, one often returns to general-purpose packages like IRAF or MIDAS for further analysis. Both run in the UNIX environment, and have similar functionality. In this course we will give an introduction to the use of IRAF.

As you might expect, IRAF is a very large package with many different functions. We will only scratch the surface in this course. You may find the user interface a bit old-fashioned, which is not surprising given that the first versions of IRAF date back to the late 1980s. However, it still remains the analysis package of choice for many astronomers across the world, and it has been updated over the years with many new tasks. For example, there is a dedicated set of tasks for the reduction and analysis of data from the Hubble Space Telescope. Similarly, the GEMINI observatory has developed its own IRAF tasks to handle data from the specific instruments available at that observatory. Later in the course we will use packages for photometry and spectroscopy. If you are going to work with astronomical data, it is very likely that you will encounter IRAF sooner or later.

## 1.1 Getting started

We suggest you start by browsing through the *Beginner's Guide to Using IRAF* by Jeanette Barnes, available via the web at: `http://iraf.net/irafdocs/beguide.pdf`. In the rest of these notes we will simply refer to this document as the Beginner's Guide. It is a general introduction to IRAF which covers the basic concepts, although parts of it are now largely obsolete (e.g. the detailed discussion of how to read tapes).

These notes cover a few additional issues specific to running IRAF as part of the OA1 course. They are designed such that you can work through them while sitting in front of a computer. Throughout these notes we will use the notation

```
> command
```

to denote a command typed at the UNIX prompt, and

```
--> command
```

for commands typed at the IRAF prompt (whether `cl` or `pyraf`).

- We will use the host `uranus`, which is a powerful Linux box owned by the Astronomical Institute. You should be able to log in on `uranus` with the same account as for `venus`. Your home directory will be the same on the two machines.
- The IRAF version on `uranus` is installed as part of the `scisoft` distribution of astronomical software, compiled by ESO. If you like, you can download `scisoft` free of charge for your own computer at `http://www.eso.org/sci/data-processing/software/scisoft/`. `Scisoft` includes many other astronomical software packages than IRAF (for example, also MIDAS).
- IRAF itself, along with lots of documentation, is available at `http://iraf.noao.edu` and `http://iraf.net`. Those are good places to look for more information. You can also download IRAF and install it from there, but getting it as part of `scisoft` is *much* easier.

You need to configure your UNIX environment so that it recognises the `scisoft` package. This is described in the `README_USERS` file in the `/scisoft` directory on `uranus`.

- Basically, you need to first find out which login shell you are using:

  ```
  > echo $SHELL
  ```

  If you are using the `bash` shell then type

  ```
  > . /scisoft/bin/Setup.bash
  ```

  Alternatively, if you are using the c-shell or one of its variants (e.g. tcsh), type

  ```
  >  source /scisoft/bin/Setup.csh
  ```

  It may be convenient to add these commands to your startup files (.cshrc or .tcshrc for csh and tcsh and .profile for bash) - otherwise you have to remember to type them every time you open a new window:

  ```
  if (-d "/scisoft") then
    source /scisoft/bin/Setup.csh
  endif
  ```

for the csh/tcsh or

```
if [ -e "/scisoft" ]; then
  . /scisoft/bin/Setup.bash
fi
```

for the bash.

- Now, make a directory called `iraf` in your home directory and go to this directory:

  ```
  > mkdir iraf
  > cd iraf
  ```

  and execute the `mkiraf` command as described in the Beginner's Guide.

- You can now start IRAF with the command

  ```
  > pyraf
  ```

  If you need to run IRAF on a system where `pyraf` is not available you will use the command `cl` (= command language). Note that you always have to start the `cl` from the iraf directory, while pyraf can be started from anywhere.

## 1.2 CL/ECL versus PYRAF

As noted above, the original user interface of IRAF, the *command language*, that you start with the command `cl`, is a bit old-fashioned. It is basically a command prompt similar to the one you may be familiar with from the UNIX/Linux environment, although the syntax differs somewhat (this is described in the Beginner's Guide). Remember that IRAF has been around for 20 years – when it was first developed, many astronomers just had a simple text-based VT100 terminal in their office without fancy graphics capabilities. So it made perfect sense to have a low-tech text-based user interface. There have been some attempts to "embed" the `cl` in a graphical interface but this wasn't really successful.

One annoying shortcoming of the `cl` is that it has no arrow-key history function. A somewhat extended version of the `cl`, called `ecl`, is included with `scisoft`. However, there is also a more modern command interpreter available, called `pyraf`. `pyraf` has most of the same command-line editing functions as a UNIX shell, and you don't need to be in your iraf directory when you start it.

However, the differences between `cl/ecl` and `pyraf` go far beyond the user interface itself. This becomes apparent once you realise that the `cl` is, in fact, a whole programming language with its own syntax and control structures. You can write and execute fairly sophisticated scripts in order to carry out complicated data reduction procedures with the `cl`. The `pyraf` interpreter can also execute these scripts, but in addition it understands programmes written in the `python` language (hence the name pyraf) which is much more versatile than the `cl`. This is however beyond the scope of this course.

For the purpose of this course, it will not make much difference whether you use `cl` (or `ecl`) or `pyraf`. There are a couple of differences between `cl` and `pyraf` to be aware of:

- In `pyraf` you cannot "unload" a package – the `bye` command has no function (see the Beginner's Guide and Sec 1.5)
- The command to quit `pyraf` is `.exit`, for the `cl` it is `logout`
- In `pyraf`, the `eparam` command (see below) brings up a separate graphics window and therefore requires a graphics-capable display. In most modern environments this is hardly an issue, but it can be a problem e.g. if you are connected via a slow network.
- The command prompts look different. In the `cl`, the prompt is `cl>` (or the name of the most recently loaded package) while in `pyraf` the prompt is `-->`
- A few of the `cl` features do not work in `pyraf`, e.g. the `=gcur` command for entering interactive graphics mode (see the Beginner's Guide). This may be one reason to use `cl` occasionally.

We recommend that you use `pyraf`. However, it is good to be aware that `pyraf` is an extra layer on top of the core IRAF system. On some systems, typically older installations, it may not be available.

## 1.3 Image file formats

IRAF recognises many different astronomical file formats. The Beginner's Guide describes the 'OIF' file format, in which an image is split into two parts: a header file with the extension"`.imh`" and a pixel file with the extension "`.pix`" containing the actual image data. Although this remains the default file format in IRAF, most astronomical images are nowadays stored in the FITS format (Flexible Image Transport System). IRAF's internal handling of the OIF is supposed to be more efficient than for FITS images, but in practice this is hardly ever noticeable. If you always specify the "`.fits`" extension explicitly when storing an image, you can be sure that it will automatically be stored in FITS format. This will make it easier to exchange data between IRAF and other software.

- IRAF has a "built-in" test image with the somewhat cryptic name `dev$pix`. You can copy this to a FITS file in your own directory using the `imcopy` task:

  ```
  --> imcopy dev$pix test.fits
  ```

  This will be useful in the following section.

## 1.4 Displaying images

You may be surprised to learn that an image display is not an integrated part of IRAF! But remember, back in those "good old days" with the VT100 terminals, graphics required special (expensive) hardware. So in IRAF the display of images remains delegated to external software, although the `cl` (and of course `pyraf`) is designed to interact with such image display software.

The Beginner's Guide describes the IMTOOL and SAOImage programmes. These are now outdated, and although SAOImage remains available in `scisoft` a more modern alternative is the `ds9` utility. Actually, `ds9` is quite a useful tool all by itself and allows, among other things, to overlay catalogs on an image and even download images from various web services.

Start `ds9` by typing

```
> ds9 &
```

at the UNIX prompt (or `--> !ds9 &` within IRAF). Be careful to start only one `ds9` at the time, otherwise IRAF will get confused!

There are two ways to display an image in `ds9`:

- Within IRAF (cl or pyraf) you can use the `display` command:

  ```
  --> display test.fits 1
  ```

  As you can see, the `display` task takes two parameters: the name of the image to displayed, and a number. The number indicates a buffer number in which the image should be displayed. `ds9` has 16 such buffers, and allows you to blink one image against another or show them next to each other. IRAF will convert the actual image data into an 8-bit bitmap and send this to `ds9`. Since astronomical data are generally stored with 16-bit or higher precision, this entails some loss of information. In fact, upon calling the `display` command, you will see IRAF producing a message like:

  ```
  --> display test.fits 1
  z1=35. z2=346.0218
  ```

  where the `z1` and `z2` numbers indicate the data values in the image that will be mapped to the lowest and highest intensity (black/white). Information outside this range will be lost. You can override this automatic scaling by specifying additional parameters when calling `display`:

  ```
  --> displ test.fits 2 zr- zs- z1=0 z2=2000
  ```

  In this case, the image will appear darker overall, but you can see details in the high intensity regions near the nucleus better. These will appear "burned out" with the default scaling. Note that we have now displayed the image in buffer number two, so you can use the "blink" function in `ds9` to compare the two scalings (under the "frame" menu).

- You can load the image directly into `ds9` via the `file` menu. In this case, there is no loss of information. You can use the `scale` menu in `ds9` to edit the display options. This way of displaying the image has the additional advantage that information about "World Coordinate Systems" in the image header will be read by `ds9` so that coordinates can be displayed in physical units (right ascension, declination). You can still load images into multiple buffers.

Independently of how the image is displayed, you can adjust the contrast by moving the mouse across the `ds9` display area while holding down the right mouse button. Moving the cursor up or down will decrease/increase the contrast, while moving the cursor left/right will increase/decrease the brightness. While this is useful for a quick-look inspection, it is in general better to adjust the scaling via the `scale` menu in `ds9` or via the `z1/z2` parameters in IRAF.

## 1.5 Tasks and packages

Similar to the UNIX command prompt, the IRAF command interpreter itself doesn't do much. Most of the functionality in IRAF is contained in various *tasks*, which are organised into *packages*. We have already encountered one such task, `display`. Like most other tasks, it has required parameters (in this case, image name and buffer number) as well as optional ones.

### 1.5.1 Loading packages

In order to use a task, the package in which it belongs must first be loaded. Some packages are automatically loaded when starting IRAF (this can be defined in the `login.cl` or `loginuser.cl` files in the iraf start-up directory). Others can be loaded simply by typing the name of the package, e.g.

```
--> images
```

will load the `images` package. Upon loading a package, the `cl` prompt will change to the two first letters of the name of the most recently loaded package (`ecl` prints the whole name). The `pyraf` prompt will remain the same. In either case, a list of the tasks contained within this package is displayed:

```
--> images
      imcoords.   imfit.      immatch.    tv.
      imfilter.   imgeom.     imutil.
-->
```

A package can contain other packages as well as tasks. Packages can be recognised since their names end with '.', so in the example above all the new items are packages. We can then load, for example, the `tv` package:

```
--> tv
      display     iis.        imedit      imexamine   tvmark      wcslab
-->
```

and we find that it contains the tasks `display`, `imedit`, etc., in addition to yet another package, `iis`. Actually, the `tv` package is among those that are automatically loaded, which is why the `display` task was already accessible before we manually (re-)loaded the package.

It is not necessary to type the full name of a task or package. Any unique abbreviation can be used. Here, `im` will not be unique, but we could load the `imutil` package, for example, by just typing `imut`.

In `cl/ecl`, the most recently loaded package can be unloaded by typing

```
cl> bye
```

The tasks in the package will then no longer be available. Note, however, that packages are different in this sense than UNIX directories. In the example above we could have loaded the `immatch` package after loading the `tv` package without any need to say `bye` to the `tv` package. In the early days of IRAF usage, loading only a minimum of packages could help optimize the use of computer memory, but on modern systems there is no real need to unload packages once they have been loaded. Indeed, in `pyraf` the `bye` command has no function.

### 1.5.2 More about tasks

To find out more about how the `display` task works, we can use the `help` function:

```
--> help display
```

and IRAF responds by displaying the help text for `display`:

```
DISPLAY (Mar97)                images.tv                DISPLAY (Mar97)



NAME
    display -- Load and display images in an image display


USAGE
    display image frame


PARAMETERS

    image
        Image to be loaded.

    frame
        Display frame to be loaded.
    bpmask = "BPM"
        Bad pixel mask.  The bad pixel  mask  is  used  to  exclude  bad
[q=quit,d=downhalf,f|sp=downfull,j|cr=downline,N=next]
```

Among other things, the top line tells us that the `display` task is found in the `tv` package, which is a sub-package of the `images` package. Of course, we knew this already, but in general this is useful information, since the `help` function works also for tasks in packages that haven't been loaded. By browsing through the help text, we find out which parameters the task accepts and what their function is.

To see all the parameters accepted by the `display` task, we can use the `lparam` command:

```
--> lpar display
        image = "dev$pix"       image to be displayed
        frame = 1               frame to be written into
      (bpmask = "BPM")          bad pixel mask
   (bpdisplay = "none")         bad pixel display (none|overlay|interpolate)
    (bpcolors = "red")          bad pixel colors
     (overlay = "")             overlay mask
     (ocolors = "green")        overlay colors
       (erase = yes)            erase frame
(border_erase = no)             erase unfilled area of window
(select_frame = yes)            display frame being loaded
      (repeat = no)             repeat previous display parameters
        (fill = no)             scale image to fit display window
      (zscale = yes)            display range of greylevels near median
    (contrast = 0.25)           contrast adjustment for zscale algorithm
      (zrange = yes)            display full image intensity range
       (zmask = "")             sample mask
     (nsample = 1000)           maximum number of sample pixels to use
     (xcenter = 0.5)            display window horizontal center
     (ycenter = 0.5)            display window vertical center
       (xsize = 1.)             display window horizontal size
       (ysize = 1.)             display window vertical size
        (xmag = 1.)             display window horizontal magnification
        (ymag = 1.)             display window vertical magnification
       (order = 0)              spatial interpolator order (0=replicate, 1=line
          (z1 = )               minimum greylevel to be displayed
          (z2 = )               maximum greylevel to be displayed
      (ztrans = "linear")       greylevel transformation (linear|log|none|user)
     (lutfile = "")             file containing user defined look up table
        (mode = "al")
-->
```

Note that all except the two first parameters appear in parentheses - this means that the first parameters are required (IRAF will ask for them if they are not specified), while the rest do not need to be specified. If they aren't specified, the default values listed by lpar will be used.

To change the default parameter values, we can use the eparam command. In pyraf this brings up a separate window. This window has a help button for the task and radio buttons/menus for boolean/multi-choice variables. You exit this window by using the save button, or execute the task directly with the execute button.

In the cl, you get a text-based interface for editing the task parameters:

```
cl> epar display
                                     I R A F
                    Image Reduction and Analysis Facility
PACKAGE = tv
   TASK = display

image   =             dev$pix  image to be displayed
frame   =                   1  frame to be written into
(bpmask =                 BPM) bad pixel mask
(bpdispl=                none) bad pixel display (none|overlay|interpolate)
(bpcolor=                 red) bad pixel colors
(overlay=                    ) overlay mask
(ocolors=               green) overlay colors
(erase  =                 yes) erase frame
(border_=                  no) erase unfilled area of window
(select_=                 yes) display frame being loaded
(repeat =                  no) repeat previous display parameters
(fill   =                  no) scale image to fit display window
(zscale =                 yes) display range of greylevels near median
(contras=                0.25) contrast adjustment for zscale algorithm
(zrange =                 yes) display full image intensity range
(zmask  =                    ) sample mask
(nsample=                1000) maximum number of sample pixels to use
More
                                                            ESC-? for HELP
```

You can then move the cursor up and down with the arrow keys and enter other values. When you are done, exit by typing ":wq" (same syntax as the vi editor).

Like task names, parameter names can also be abbreviated on the command line - the zrange and zscale parameters correspond to the zr and zs arguments in the example in Sec. 1.4. These are examples of boolean parameters, that can take the values yes or no. The notation zr- on the command line is shorthand for zrange=no.

- Try setting the ztrans parameter in display to log. Then redisplay the image.

## 1.6 Interacting with the image display

Many tasks in IRAF can interact graphically with the user. As an example, let us take a look at the `imexamine` task which is useful for visually inspecting an image and carrying out a few basic measurements. Here we just give a few examples of things you can do with `imexamine`. You can read more about this task in the Beginner's Guide and in the built-in help.

If called without any parameters, `imexamine` will work on the image that is currently displayed in `ds9`. The task can also be called with an image name (or a list of image names) as parameter; in this case the (first) image will be loaded into the display. Hence, you may examine the image `dev$pix` like this:

```
--> display dev$pix 1
z1=35. z2=346.0218
--> imexam
```

or like this:

```
--> imexam dev$pix
```

Note that, in both cases, we have abbreviated the command name to `imexam`, which is still unique.

You will see that the cursor now has moved from the terminal window to `ds9`. `imexam` is now waiting for further commands (depending on how your window manager is configured, you may still have to click on the `ds9` window to make it active).

- If you move the cursor to a feature in the image and hit the "a" key, `imexamine` will measure the counts in a circular aperture centered on the feature and calculate a few statistics including the centroid, counts, ellipticity and the FWHM (full width at half maximum).
- You can get a plot of the radial profile of the source by using the "r" key.
- "s" will produce a surface plot of the region of the image near the cursor
- To get a list of all the functions, hit the "?" key. Note that the cursor now moves back to the terminal window (again, you may have to click on it). After having read the help message, type "q" to get back to the `ds9` window.
- You can play around with the different commands. When you are done, hit "q" in the `ds9` window and you will get the IRAF command prompt back.

These examples illustrate a common feature of many tasks in IRAF: there is an *image cursor* that allows you to interact with the image display and a *graphics cursor* that allows you to interact with the graphics window.

## 1.7 Summary

In this lesson you have learned:

- Basic steps to get get started using IRAF
- The concepts of *packages* and *tasks*
- How to load a package
- How to view and edit the parameters of a task
- How to use the *help* function in IRAF
- How to display an image and do some simple analysis

# Chapter 2

# Basic reduction of CCD images

The raw images delivered by a CCD camera generally contain various instrumental artefacts that need to be removed before further analysis can be done. To understand which steps are required, it is useful to recall a few basic facts about CCD data.

## 2.1 Removal of detector artefacts

A CCD detector is divided into a number of pixels, each of which is capable of holding a number of electrons (on order $10^5$). CCD detectors used for astronomy now typically have $2048 \times 2048$ or $2048 \times 4096$ pixels, but can be mosaiced together to larger formats. When a new exposure is initiated, the whole detector is first cleared and electrons are then released by incoming photons during the exposure and accumulated in each pixel. The number of electrons $N_e$ is, to a very high degree of accuracy, proportional to the number of incoming photons $N_p$:

$$N_e = \text{Q.E.} \times N_p \tag{2.1}$$

where the proportionality factor is called the *Quantum Efficiency*. For modern CCD detectors this can be as high as 90%.

When the exposure time (typically a few minutes) has elapsed, the pixels are "read out" one by one. The charge in each pixel is converted to a voltage, which in turn is amplified and converted to a 16-bit number by an analogue-to-digital (A/D) converter. The resulting "Digital Number" is again linearly related to the number of electrons accumulated during the exposure:

$$\text{D.N.} = N_e/\text{Gain} + \text{Bias} \tag{2.2}$$

Note that a *Bias* level is added during the read-out process. The use of the term "Gain" in Eq. (2.2) is unfortunately somewhat inconsistent with the standard usage e.g. in electronics, and a more proper term would be "inverse Gain". However, we are using the term Gain here in the same way it is used in many IRAF tasks, i.e. number of electrons per D.N.

It should be mentioned that electrons may also be released due to *dark current* in the CCD. The dark current is very strongly temperature sensitive, and can be reduced to negligible levels by cooling the detector with liquid nitrogen. However, for completeness we note that the processing of a CCD image may also involve the subtraction of this dark signal. Since the dark current can vary quite strongly from one pixel to another, the best way to correct for it is to obtain a "dark exposure" with the same duration as the science exposure, but with the camera shutter closed. As long as the CCD temperature is maintained at a constant level, this can be conveniently done during the day in order to not waste observing time at night.

In reality, the sensitivity of the CCD can vary somewhat across the chip. For the purpose of the present discussion we may view this as due to variations in the quantum efficiency, although there can also be other reasons (e.g. dust particles on the CCD or on optical surfaces in front of it, or different parts of the chip being read out through different amplifiers with slightly different gains). This variation can be accounted for by observing a uniformly illuminated surface, such as the twilight sky or the inside of the telescope dome. Such an image is called a *flat-field* image. Once the Bias level has been subtracted from the science exposure, the sensitivity variations can be corrected by division with the (also bias-subtracted) flat-field image. In order to keep the Gain of the processed CCD image close to that of the raw data, the flat-field image is typically normalised to a mean value of unity.

We can then summarise the main steps involved in the initial reduction of a CCD image as follows:

- Subtract the Bias level from the science image and the flat-field exposure
- A (bias-subtracted) dark frame may be subtracted if dark current is significant. The flat-field exposure will almost always be short, so for this dark current can be safely neglected.
- Normalise the bias-subtracted flat-field exposure to a mean value of 1
- Divide the bias-subtracted science image by the normalised flat-field.

The result of this process is then an image in which the digital numbers are directly proportional to the flux incident upon each pixel during the exposure.

## 2.2 Further processing

Because astronomical targets are faint, the exposure times required to obtain useful measurements are often long. Long integrations are often split into several shorter ones, for a number of reasons:

- CCDs are also sensitive to cosmic ray (CR) hits and during long exposures, a significant fraction of the pixels in a CCD image can be affected by CRs. This can prevent accurate measurements of the sources of interest. Since the CR hits are random events, they will affect different pixels if another exposure is made, and can therefore be eliminated by splitting long exposures into several shorter ones.
- Telescope tracking errors: Although best efforts are made to make the telescope track objects accurately as they move across the sky, this may occasionally fail. This might render a long exposure useless, or at least lead to significant degradation of image quality. A shorter exposure can simply be rejected and repeated.

- It is often the case that several pixels, or columns of pixels, in a CCD detector do not provide useful data. These pixels may simply be ”dead”, or suffer from much higher than average dark current (”hot” pixels) so that they saturate even in a relatively short exposure. By including a small offset of the telescope pointing between multiple exposures, such bad pixels can be eliminated in the same way as CR hits.

- In long exposures the full-well capacity of a pixel may be exceeded for bright objects, leading to saturation and loss of information. Again, this can be avoided by splitting a long exposure into several shorter ones.

- Having several exposures of a field allows monitoring of the photometric stability. If the number of counts for objects in the field differ widely from one exposure to another, this may be an indication of extinction variations (for example due to passing clouds).

- For images where the spatial resolution is dominated by the pixel size (“undersampled” images), an improvement in the resolution may be achieved by combining several images shifted by sub-pixel offsets. This technique, known as *drizzling*, is commonly used for HST images.

It is then clear that one often needs to combine several exposures to a single equivalent long exposure. It may seem that the most effective way to reject outliers such as cosmic ray hits or bad pixels is to median combine the images, but as we have seen in a previous lecture this will lead to a reduction in S/N of about 25% compared to simple averaging of the images. Of course, for a simple average the bad pixels will not be eliminated, but this can be accomplished by using a sigma-clipping algorithm whereby pixels that deviate by more than a specified number of standard deviations from the mean are rejected.

If telescope offsets are performed, then each image will need to be shifted relative to a reference image before the combination is performed.

Flat-fields are also typically obtained by combining many short exposures. Here, the goals are (at least) two-fold:

- A flat-field image, like any other image, will be subject to Poissonian noise. The Poisson noise in the flat-field image should be small enough that it does not lead to a significant degradation of the science image. For a full-well capacity of $10^5$ electrons, the Poissonian noise will be about 0.3% per pixel if the image is exposed to near the saturation limit. In practice, the response of a CCD often becomes non-linear near saturation, so it is wise to avoid getting close to saturation. A more realistic noise level may then be $\sim$ 1% for a single flat-field image. For some applications this may be comparable to the photometric accuracy aimed at, so it is desirable to beat down the Poisson noise by combining several such exposures.

- Sky flats may contain stars, even if an effort is made to point the telescope at a ”blank” patch of sky. In this case, the stars represent an undesirable contamination that must be eliminated. Similar to cosmic rays or bad pixels, this can be achieved by combining several exposures, pointed at different regions of the sky.

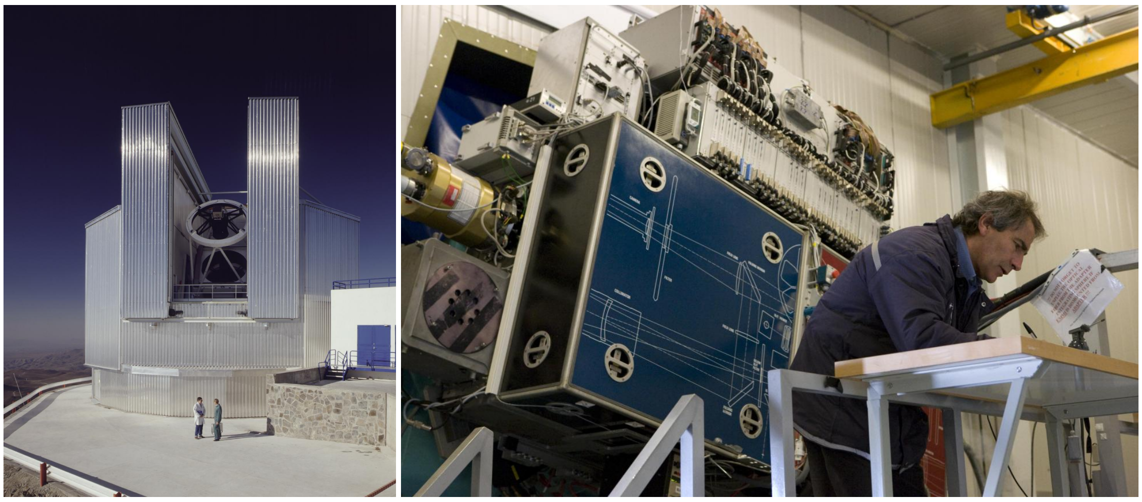


Figure 2.1: The ESO 3.5 m New Technology Telescope (left) and the EMMI instrument (right)

## 2.3 Reduction of CCD images in IRAF

The reduction steps described above can all be carried out within IRAF. In the following, we will reduce a set of CCD images taken with the *ESO Multi-Mode Instrument* (EMMI) on the *New Technology Telescope* at La Silla Observatory in Chile. The data are *B*, *V* and *I* images of the asteroid 4829 Sergestus, which belongs to a group of asteroids known as the Jupiter Trojans. These are found near the Sun-Jupiter L4 and L5 Lagrange points.

In this exercise we will carry out the basic reduction steps of the images of 4829 Sergestus itself and a Landolt standard star field. In a subsequent exercise, we will use the reduced images to carry out photometry of 4829 Sergestus.

What follows is an outline of the rudimentary steps required for the reduction of CCD images. In practice, procedures may be somewhat different, depending on the characteristics of the particular instrument you are working with. As usual, more extensive documentation is available at `http://www.iraf.net`, in particular *A User's Guide to CCD Reductions with IRAF* by Phil Massey (`http://iraf.net/irafdocs/ccduser3.pdf`).

### 2.3.1 EMMI

EMMI is a multi-purpose instrument designed for both imaging and spectroscopy. It was installed on the NTT in 1990, and until it was decommissioned in 2008 it was one of the main workhorses of the La Silla observatory. Fig. 2.1 shows the telescope in its enclosure (left) and EMMI itself (right), mounted at one of the Nasmyth foci of the NTT.

EMMI has two "arms" for observations from 300 to 500 nm ("blue arm") and 400 nm to 1000 nm ("red arm"). The data we will use here were taken with the red arm, which has two $2048 \times 4096$ pixel CCD detectors mounted in a mosaic, thus equivalent to a single $4096 \times 4096$ detector apart from a small gap between the two CCDs. The total field of view is $9.9' \times 9.1'$

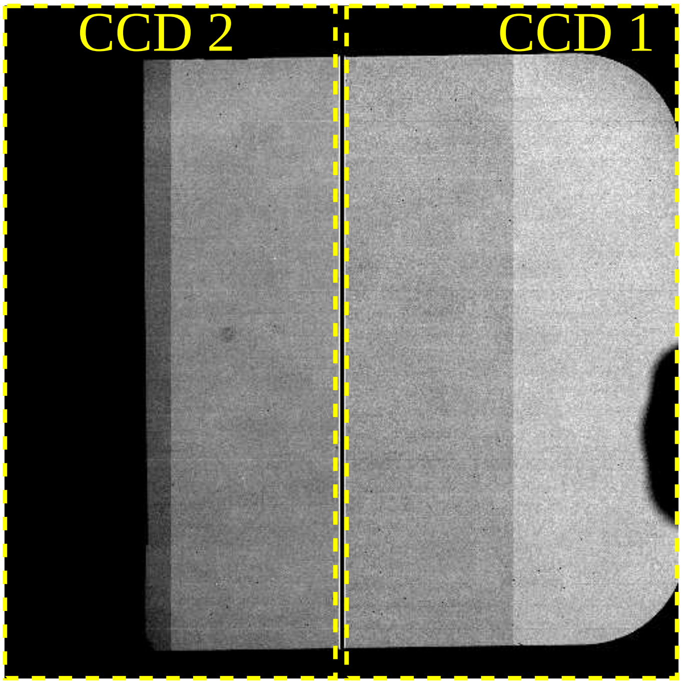


Figure 2.2: $V$-band flat-field image from EMMI

and the pixel scale is 0.166″ per pixel. Since such a small pixel scale is really only needed in the most excellent seeing conditions, the detectors are often *binned* by a factor of two during read-out so that each pixel in the final image really consists of $2 \times 2$ pixels on the detector. This saves time during the read-out process, and also makes the final image sizes smaller. The data used here are binned in this way.

The read-out noise is 9 $e^-$ per pixel and the (inverse) gain is 1.25 $e^-$ per D.N.

Fig. 2.2 shows a $V$-band flat-field image taken with EMMI. Data from the two CCD chips have been mosaiced together in a single FITS file so they can be viewed as a single image. Apart from the gap between the two detectors (the dark vertical strip in the centre of the image) you can also see that only about half of CCD #2 is illuminated, and that each chip is again divided into two halves with somewhat different mean count levels. This is because each CCD is read out through two separate amplifiers with slightly different gain and bias level.

### 2.3.2 Getting the data

First, make a directory for the data:

```
> mkdir OA1data
> cd OA1data
```

The data are available as a gzip'ed tar file from the course web page. It may be easier to get them with the UNIX `wget` command directly from `uranus`:

Table 2.1: Data files

| ARCFILE | FILE | OBJECT | FILTER | EXPTIME | AIRMASS |
|---|---|---|---|---|---|
| EMMI.2005-01-17T08:17:27.821.fits | ser_b.fits | 4829_Ser_17_0600 | Bb#605 | 39.9992 | 1.383 |
| EMMI.2005-01-17T08:19:53.706.fits | ser_i.fits | 4829_Ser_17_0600 | I#610 | 6.3987 | 1.391 |
| EMMI.2005-01-17T08:18:51.300.fits | ser_v.fits | 4829_Ser_17_0600 | V#606 | 9.0994 | 1.388 |
| EMMI.2005-01-18T00:35:39.349.fits | pg0231_b.fits | STD | Bb#605 | 29.9996 | 1.274 |
| EMMI.2005-01-18T00:34:33.082.fits | pg0231_i.fits | STD | I#610 | 9.9996 | 1.272 |
| EMMI.2005-01-18T00:32:33.249.fits | pg0231_v.fits | STD | V#606 | 9.9992 | 1.269 |
| EMMI.2005-01-16T19:32:59.672.fits | bias_1.fits | BIAS | Free | 0.0611 | 1.000 |
| EMMI.2005-01-16T19:33:18.374.fits | bias_2.fits | BIAS | Free | 0.0605 | 1.000 |
| EMMI.2005-01-16T19:33:34.996.fits | bias_3.fits | BIAS | Free | 0.0607 | 1.000 |
| EMMI.2005-01-16T19:33:52.798.fits | bias_4.fits | BIAS | Free | 0.0607 | 1.000 |
| EMMI.2005-01-16T19:34:12.850.fits | bias_5.fits | BIAS | Free | 0.0606 | 1.000 |
| EMMI.2005-01-16T19:34:32.843.fits | bias_6.fits | BIAS | Free | 0.0606 | 1.000 |
| EMMI.2005-01-16T19:34:50.294.fits | bias_7.fits | BIAS | Free | 0.0606 | 1.000 |
| EMMI.2005-01-16T19:35:08.656.fits | bias_8.fits | BIAS | Free | 0.0605 | 1.000 |
| EMMI.2005-01-16T19:35:29.009.fits | bias_9.fits | BIAS | Free | 0.0607 | 1.000 |
| EMMI.2005-01-16T19:36:38.076.fits | bias_10.fits | BIAS | Free | 0.0604 | 1.000 |
| EMMI.2005-01-17T09:28:41.547.fits | skyflat_b1.fits | SKYFLAT | Bb#605 | 4.6860 | 1.311 |
| EMMI.2005-01-17T09:29:04.589.fits | skyflat_b2.fits | SKYFLAT | Bb#605 | 4.3166 | 1.310 |
| EMMI.2005-01-17T09:29:27.302.fits | skyflat_b3.fits | SKYFLAT | Bb#605 | 3.9913 | 1.308 |
| EMMI.2005-01-17T09:23:48.226.fits | skyflat_i1.fits | SKYFLAT | I#610 | 4.6469 | 1.331 |
| EMMI.2005-01-17T09:24:11.438.fits | skyflat_i2.fits | SKYFLAT | I#610 | 4.2657 | 1.329 |
| EMMI.2005-01-17T09:24:34.250.fits | skyflat_i3.fits | SKYFLAT | I#610 | 3.9298 | 1.328 |
| EMMI.2005-01-17T09:26:16.581.fits | skyflat_v1.fits | SKYFLAT | V#606 | 5.0701 | 1.321 |
| EMMI.2005-01-17T09:26:40.804.fits | skyflat_v2.fits | SKYFLAT | V#606 | 4.6550 | 1.319 |
| EMMI.2005-01-17T09:27:05.876.fits | skyflat_v3.fits | SKYFLAT | V#606 | 4.2580 | 1.317 |

```
> wget http://www.astro.uu.nl/~larsen/teaching/OA1/data/Emmi.tar.gz
```

Then extract the data from the file:

```
> tar xvzf Emmi.tar.gz
```

Table 2.1 gives some basic information about the data. The first column lists the name under which each file is stored in the ESO data archive. These are in a standard format for ESO data with the first characters identifying the instrument and the rest of the filename indicating the time (in U.T.) when the image was first stored. Next follows the name of the actual file you'll get when unpacking the `Emmi.tar.gz` file. The ESO archive file names have been replaced by more informative ones, and the data for the two CCD detectors have been mosaiced together to a single image. Other than this, no further processing of the "raw" archive files has been done. The next columns list some of the header information: The object name, filter used, exposure time, and the airmass at the time the exposure was started.
You will see that the package includes four types of images:

- Images of the science field in the *B*, *V* and *I* filters

- Images of a Landolt standard field (PG0231+051) in the same filters
- Skyflats
- Bias frames.These are very short exposures taken with the shutter closed, which serve to indicate the bias levels of the CCD detectors (slightly different for each chip and A/D converter). Obviously, filter information is irrelevant for these frames.

- Use IRAF to look at some of the images. 4829 Ser is the object at $(x, y) = (1375, 1043)$ in the science frames. Can you identify the standard stars shown in the finding chart in Fig. 2.3? (*Hint:* if you are having trouble with this, try rotating the EMMI image 180 deg.)
- Use the `imexamine` task to measure the seeing on the images of 4829 Ser. Be careful to avoid bright objects, which might be saturated! Since these are 16-bit images, the maximum possible pixel value is 65535, so if this occurs then the dynamic range has been exceeded. Note that, for EMMI data, the A/D converter "saturates" before the CCD itself. You can check the maximum count level under the cursor by pressing "m" in imexam. It may also be helpful to look at the radial profile of an object with the "r" key, or get a surface plot with the "s" key.

### 2.3.3 Preparation of calibration frames

#### Master bias frame

The first step is to prepare a master bias frame, since this will need to be subtracted from both the flat-field and science frames. You can see that 10 bias frames are included. We will average combine these to a single master frame, but first it is good to check that they are all ok (e.g. that no science or flat-field frames have been mislabelled as "bias" frames).

- Make a text file with a list of all the bias frames. This can be done with the UNIX "ls" command:

  ```
  --> !ls bias_*.fits > bias.txt
  ```

  (note that UNIX commands can be called within pyraf by putting an "!" mark in front of them).

- Now use the `imstatistics` command to list some basic statistics for the bias images:

  ```
  --> imstat @bias.txt
  ```

  Many tasks in IRAF can work with *lists* of images. The `imstat` call above is an example of how this works: instead of giving a single image name, the name of a text file with a list of images can be given, preceded by an "@" character.

- Verify that all the bias frames have (about) the same mean level and standard deviation.

We are now ready to combine all the individual bias frames to a single master bias image. For this purpose we will use the `imcombine` task in IRAF. The task has just two required parameters: a list of images to be combined, and the output image. However, there are many optional parameters, so let's first take a look at those:

```
--> lpar imcombine
```

The most relevant ones are:

- `combine`: average / median / sum. The value of this parameter determines how images are combined.
- `reject`: none / minmax / ccdclip / crreject / sigclip / avsigclip / pclip. This specifies how outlying pixels are rejected. For sigclip/avsigclip/pclip, the rejection of bad pixels is based on the standard deviation calculated from the actual pixel values. For ccdclip/crreject, the standard deviation is instead calculated based on Poisson statistics, using the information about the CCD gain and read-noise.
- `scale`: none / mode / median / mean / exposure / list / keyword. `imcombine` can apply a scaling factor to each image before they are combined. The value of the `scale` parameter specifies how this is done.
- `zero`: none / mode / median / mean / list / keyword. Similarly, an offset can be applied to each image.
- `weight`: none / mode / median / mean / exposure / list / keyword. For average combination, this parameter specifies how the weighting of each image is calculated.
- `statsec`: This parameter specifies the region of the images used for the calculation of the mode/median/mean used in the scaling, zero-point offsets and weights. It is given in the usual notation for image sections: `statsec = [x1:x2,y1:y2]`. If left blank, the whole image is used.
- `lsigma / hsigma`: The limits below and above the mean where outlying pixels are rejected
- `rdnoise / gain / snoise`: CCD read noise and gain parameters, used if `reject` is set to `ccdclip` or `crreject`.

For a full discussion of these parameters and others, see the built-in help for `imcombine`.

Here we suggest to combine the bias frames using `combine=average` and `reject=crreject`. Remember to set the relevant `rdnoise` and `gain` parameters! For the bias frames, no image scaling or zero-point offsets are needed, so set `scale=none` and `zero=none`.

- Use the `epar` command to set the parameters for the `imcombine` task.
- Then produce the master bias by calling `imcombine`:

  ```
  --> imcombine @bias.txt masterbias.fits
  ```

**Subtract masterbias from flatfields**

Now that we have a masterbias frame, we can proceed to produce our flatfield frames. Since the flatfield correction can be wavelength dependent, a separate flatfield is needed for each filter used for the observations.

We first need to subtract the masterbias.fits frame from each of the flatfield images. For this we can use the `imarithmetic` task. Here we can again make use of IRAF's ability to handle lists of images, with an extra "trick":

- As for the bias frames, make a text file with a list of all the raw sky flats:

  ```
  --> !ls skyflat_??.fits > skyflat.txt
  ```

- Manually edit the `skyflat.txt` file and remove the `.fits` extension from each filename, so that the file looks as follows:

  ```
  --> !cat skyflat.txt
  skyflat_b1
  skyflat_b2
  skyflat_b3
  skyflat_i1
  skyflat_i2
  skyflat_i3
  skyflat_v1
  skyflat_v2
  skyflat_v3
  ```

  (you could have skipped this last step by using a bit of extra UNIX trickery:

  ```
  --> !ls skyflat_*.fits | sed s,.fits,, > skyflat.txt
  ```

  )

- The nifty bit is now that by appending a string to the output *list* when calling `imarithmetic`, this string will be appended to each of the output `images`. We can therefore carry out the subtraction of the masterbias frame from all the skyflats in just one step:

  ```
  --> imarit @skyflat.txt//.fits - masterbias.fits @skyflat.txt//_b.fits
  ```

- We now have a bias-subtracted version of each skyflat with an extra "_b", e.g. `skyflat_b1_b.fits` is the bias-subtracted version of `skyflat_b1.fits`

**Combine the bias subtracted flat-field frames**

We now need to combine the flat-field frames. This can be done in the same way as for the bias frames, with one exception: We need to let `imcombine` scale the individual frames so that the rejection algorithm works properly. Here, the median is appropriate since we do not want the scaling to be affected by outlying pixels.

- Set the `scale` parameter to `median`
- Set the `statsec` parameter to an appropriate image section. We only want to use the illuminated part of the flatfields for scaling. Something like `statsec=[1100:1500,200:1800]` should be fine.
- Generate lists of the bias-subtracted skyflats and call imcombine, i.e.

  ```
  --> !ls skyflat_b?_b.fits > ff_b.lst
  --> imcombine @ff_b.lst ff_b.fits
  ```

  and similar for the other filters. We are now left with the combined (but not yet normalised) flat-field images, `ff_b.fits`, `ff_v.fits` and `ff_i.fits`.

**Normalise the combined flat-fields**

The (almost) last step before we are ready to reduce the science frames is to normalise the flat-fields.

- Use the `imstatistics` task to calculate the mean of each flatfield. Again, it is better to use only the illuminated fraction of each frame, e.g.

  ```
  --> imstat ff_b[500:1900,150:1800]
  ```

- Then use `imarithmetic` to divide each flatfield by the mean, e.g.

  ```
  --> imarith ff_b / 29850 masterflat_b.fits
  ```

Note that these examples are deliberately somewhat sloppy about specifying the `.fits` extension - IRAF will generally find the images even if the extension isn't specified (confusion may arise if there are two images where only the extension differs). However, when a new image is produced it is better to be explicit (although most installations will now have the default output format set to `.fits`).

Oh, one last thing: If you now display the normalised flatfield images you should see that the illuminated area has pixel values close to 1. The non-illuminated part will be close to, but not exactly, 0. The same will be the case for the science images, so in these non-illuminated parts the pixel values will fluctuate wildly in the final reduced images. This is quite undesirable, since we will later use an automatic algorithm to identify sources for photometry in the data, and such an algorithm will be confused by these fluctuations. Fortunately, there is an easy way around it:

The default behaviour of `imarithmetic` is to set output pixel values to zero in case of division by zero. If we therefore set the non-illuminated regions of the flat-field images explicity to zero, these regions will also be zero in the final reduced images. This can be done with the `imreplace` task:

- Use `imreplace` to set pixel values below a certain threshold to 0. This can be controlled by the `upper` and `lower` parameters in `imreplace`. For example,

  ```
  --> imreplace image.fits 0.0 lower=INDEF upper=0.5
  ```

  This will leave pixels with values above 0.5 D.N. unaffected, while pixels below this value are set to 0.

Done!

### 2.3.4 Reduction of the science frames

With all the preparatory work done, there are now just two remaining steps:

- Subtraction of the `masterbias.fits` frame from each image
- Division by the normalised `masterflat` frames

Since there are only three science images, it's probably not really worth the effort of going through the process of generating image lists. Instead, you may just want to call `imarithmetic` individually for each image.

- Reduce the science images by first subtracting the masterbias frame, and then dividing by the appropriate masterflat image. Remember to check the `divzero` parameter in `imarithmetic`.
- Those of you who wonder if both operations can be combined into one may be delighted to learn that the answer is *yes*. The `imcalc` task in the `stsdas` package can perform multiple arithmetic operations in one step, though the syntax is slightly cryptic - see the built-in help for details!

### 2.3.5 Reduction of the standard star frames

This is done in exactly the same way as the science frames.

PLATE 27

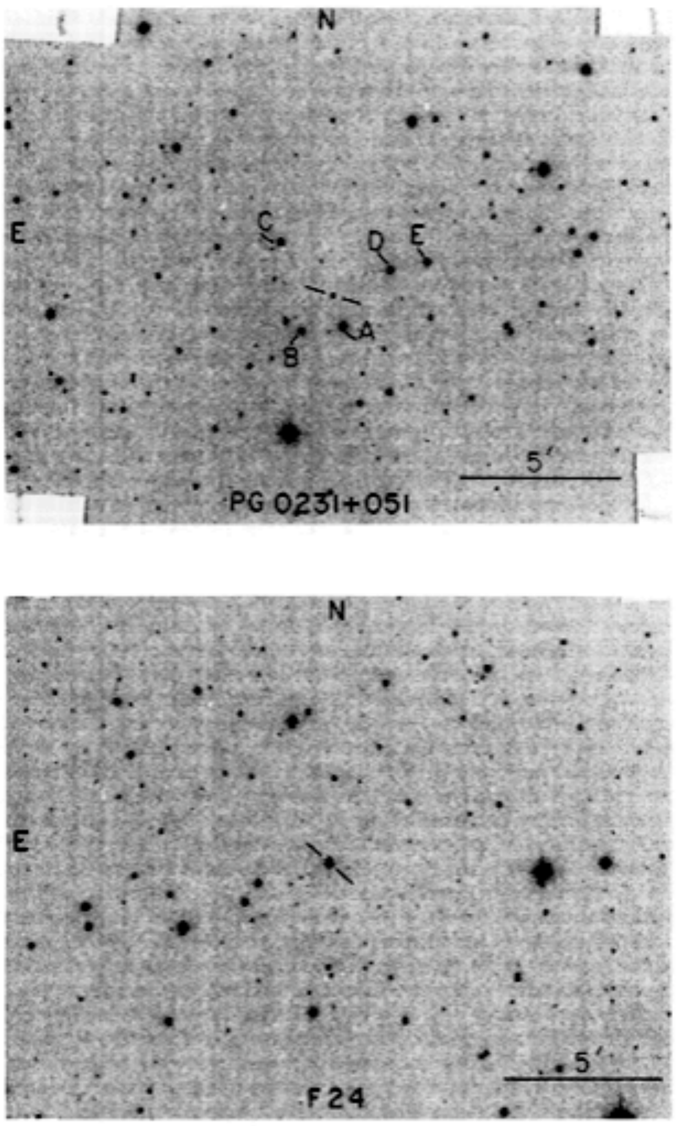


PLATE 27. (a) Sequence in the field of the star PG 0231+051. (b) The field of the star Feige 24.

A. U. Landolt (see page 347)

Figure 2.3: Finding chart for the Landolt field PG0231+051 (top)

Table 2.2: Landolt (1992) photometry for standard stars in the field of PG0231+051

| | | | | | | Err. on the Mean | | |
|---|---|---|---|---|---|---|---|---|
| Star | $\alpha$(2000) | $\delta$(2000) | V | B−V | V−I | V | B−V | V−I |
| PG0231+051E | 02:33:28 | +05:19:44 | 13.804 | 0.677 | 0.757 | 0.0046 | 0.0040 | 0.0023 |
| PG0231+051D | 02:33:33 | +05:19:28 | 14.027 | 1.088 | 1.256 | 0.0029 | 0.0075 | 0.0110 |
| PG0231+051A | 02:33:40 | +05:17:38 | 12.772 | 0.710 | 0.799 | 0.0008 | 0.0015 | 0.0030 |
| PG0231+051C | 02:33:41 | +05:20:19 | 13.702 | 0.671 | 0.783 | 0.0014 | 0.0078 | 0.0085 |
| PG0231+051 | 02:33:41 | +05:18:40 | 16.105 | −0.329 | −0.534 | 0.0068 | 0.0083 | 0.1221 |
| PG0231+051B | 02:33:45 | +05:17:30 | 14.735 | 1.448 | 1.951 | 0.0030 | 0.0072 | 0.0057 |

### 2.3.6 Additional steps

We have now carried out the basic reduction steps and are ready to analyse the images further. However, it should be remarked that there may be variations and additions to the steps followed above. A few examples:

- *Overscan* correction: Instead of generating and subtracting a master bias frame, some CCDs include an *overscan* region that can be used for bias subtraction. This is done simply by reading out more columns than are physically present on the CCD. For example, if 2100 columns are read out from an $2048 \times 2048$ pixel array, then the last 52 pixels in each row that is read out will already have been cleared by the read-out process, and the (mean) bias level can be determined from these pixels.

  Sometimes, bias and overscan corrections are treated independently. There might be variations in the bias level from pixel to pixel, which cannot be corrected using an overscan correction which only gives a mean bias level for each column. This then requires a full bias image. On the other hand, the bias level may vary systematically from one exposure to another, and in such a case the overscan region is needed to determine the mean bias level.

- Repair of bad pixels: The EMMI detectors are cosmetically quite good, with few defect pixels. However, this is not always the case. For detectors with many bad pixels, or entire bad rows or columns, it may be desirable to “fix” these in an additional step. IRAF includes a task called `fixpix`, which interpolates across image sections specified in a “bad pixel map”. Clearly, such fixing should not be overdone, as bad data should ideally be excluded from analysis rather than “repaired”. A better way to eliminate the effect of bad pixels is to obtained dithered exposures.

- Illumination correction: In the discussion above, it was implicitly assumed that the flat-field image is a good representation of sensitivity variations across the chip. Again, this assumption may not always be true. In some instruments it is very difficult to obtain a uniformly illuminated image, because internal reflections in the instrument optics can introduce “ghosts”.

In particular, a common problem is "sky concentation" in which light is scattered from the edges of the field towards the centre, so that the flat-field image appears brighter near the centre. This effect can be quite deceptive, because the science images will suffer from the same effect so that the sky background will indeed appear "flat" after flat-fielding. However, stars near the centre will appear systematically too faint. The solution to this problem is to divide by an *illumination* frame, which can be constructed by placing a star (e.g. in a standard field) in various positions across the image and measure the counts from this star vs. $(x, y)$ position after flat-field correction. The illumination frame can then be constructed by fitting a smooth two-dimensional function (e.g., a polynomial) to counts versus position.

- Dark current removal: This has been mentioned above.
- Fringe correction: Sky lines may cause an interference pattern across the CCD image, especially at longer wavelengths. Correction for this effect is tricky because the intensity of these lines, and hence the fringe pattern, can vary strongly during the night. Assuming that the *shape* of the fringe pattern is at least constant, one can attempt to remove fringing by constructing a single fringe frame by observing an "empty" region of the sky and scaling the fringe pattern as required for each science frame. In practice, a full correction for this effect is very difficult.

## 2.4 Summary

In this lesson you have learned:

- Basic properties of CCD images
- Commands in IRAF for basic reduction of CCD images.
- How to handle *lists* of images

# Chapter 3

# Photometry in IRAF

In Chapter 2 we reduced a set of CCD images of the asteroid 4829 Ser and a Landolt standard field. We will now carry out aperture photometry on these images and thereby measure the *B*, *V* and *I* magnitudes of 4829 Ser. For this we will use the DAOPHOT package, which is available within IRAF. We will go through the steps of measuring instrumental magnitudes for the science and standard star field and calibrate the science data to the standard system.

## 3.1 DAOPHOT

DAOPHOT is originally a Fortran programme, written by Peter Stetson (1987: PASP 99, 191). It includes a number of specialised tasks for *crowded field photometry*, but here we will just use the basic aperture photometry task, called `phot`. `phot` will measure the number of counts in a circular aperture centered on each star, subtract the background measured in an annulus around the star, divide by the exposure time and convert to an instrumental magnitude.

Within IRAF, DAOPHOT is found in the `noao.digiphot.daophot` package, so start by loading this package.

## 3.2 Standard calibration

In order to calibrate the photometry we will use the Landolt field (PG0231+51) from Chapter 2. Our first step will be to produce a coordinate list for the stars we want to measure. We will then feed this list to the `phot` task in DAOPHOT.

### 3.2.1 Generating a coordinate list

DAOPHOT includes a task, `daofind`, which will automatically find objects that stand out above the background noise in an image. This is very useful if the image contains a large number of stars (e.g. an image of a globular cluster). However, since we are dealing with a small number of stars in each image, it is straight forward to generate the coordinate list manually.

The `phot` task will recalculate the centroid of each star in the coordinate list, so if all the images are well aligned with each other we can use a single coordinate list for all three images.

- Check that the *B*, *V* and *I* images of the Landolt field are well aligned. You can do this by displaying each image in a separate frame in `ds9` and cycle through the frames. You can also use the `imexamine` task to check that the same star has similar coordinates in the different images (within 1–2 pixels).
- Next, make a list of coordinates for the standard stars in Table 2.2. Use the finding chart in Fig. 2.3. The list should be a simple text file with two columns, listing the *x* and *y* CCD coordinates for each star. You can measure these with `imexamine`. Make sure you remember in which order the stars appear in your input list, so that you can easily identify them later in the photometry output file!

### 3.2.2 Aperture photometry for the standard stars

We are now ready to carry out aperture photometry on the standard stars. For this we use the `phot` task in DAOPHOT. As input, `phot` takes an image and the corresponding coordinate list, and as output it produces a list of magnitudes and associated parameters (sky background, errors, etc.) for each object. As in other IRAF tasks, the default behaviour of `phot` is controlled by a set of additional parameters.

- Look at the parameters for the `phot` task.
  - The `skyfile` parameter can be left empty (only required if you have measurements of the sky background already and want to use these)
  - You probably want to set the `interactive` parameter to NO.

Many of the important parameters are in separate parameter files called `datapars`, `centerpars`, `fitskypars` and `photpars`. In `pyraf` you can access these by simply clicking on them within the `epar` window, or by calling them explicitly from the command line, e.g.

```
--> epar datapars
```

Since these parameters are common to all tasks in the DAOPHOT package, not all of them are relevant for every individual task.

- In `datapars` you find parameters that contain information about the data. You can leave most of these at the default values. The important ones here are:
  - `epadu` and `readnoise`: Should be set to the values appropriate for EMMI (see Sec. 2.3.1)
    *Important note: be careful not to confuse the* `epadu` *parameter with the* `gain` *parameter. The latter specifies an image header keyword which contains the gain.*
  - `exposure`: Set this parameter to the header keyword (EXPTIME) containing the exposure time.

- In `centerpars` you define how `phot` centers the aperture on each star. We let PHOT do some recentering so that we can use the same input coordinate file for all images, despite some small shifts. So make sure to set `calgorithm` to `centroid`. Also make sure that `maxshift` is large enough to accommodate the necessary shifts.

- `fitskypars`: As the name suggests, the sky annulus and the method to compute the sky background level from the pixel values in the annulus are defined here.

  - `salgorithm`: many choices, pick one you find appropriate (see the help text). It is not so critical for these relatively uncrowded fields. `mean` should be just fine.
  - `annulus` and `dannulus`: These specify the inner radius and width of the sky annulus. Again, rely here on your judgment.

  Keep notes of the sky parameters you choose for the report. Remember to justify your choices!

- `photpars`: Here you specify the aperture radius (`apertures`). It is also possible to specify a list of apertures, given as a sequence of numbers separated by commas (e.g. 3,5,10).

  Keep a note of the aperture radius for the report

  The `zmag` parameter contains the zero-point for the instrumental magnitudes. The instrumental magnitudes $m_i$ are then defined as

$$m_i = \mathrm{zmag} - 2.5 \log_{10} DN_{\mathrm{star}} + 2.5 \log_{10} t_{\mathrm{exp}} \tag{3.1}$$

  where $t_{\mathrm{exp}}$ is the exposure time. See the built-in help (and Appendix A) for details.

- When you are happy with the parameter settings, simply run the `phot` task for each image (or, you can specify the `image`, `coords` and `output` parameters as lists and do everything in one call).

Now use a text editor (or the UNIX `less` command) to take a look at the output files from `phot`. They will contain a header listing the relevant parameter settings, followed by the actual measurements. The format is explained in the PHOT help text, which is also reproduced in Appendix A.

- For one or two stars, verify the magnitudes computed by `phot`, using the measured FLUXes.

### 3.2.3 Defining the calibration relations

Next, we want to establish the transformations from our instrumental magnitudes to the standard system. For this we need the output from `phot`, as well as the standard photometry published by Landolt (Table 2.2). DAOPHOT contains a separate package, called `photcal`, designed to derive and apply photometric transformations. However, this package is somewhat complicated (although very versatile) to use and in the following we will adopt a simpler approach.

Since the photometry files contain a lot of information which is not really necessary in order to derive these transformations, it is useful to dump just the magnitude and associated error for each standard star to a separate text file. This can be done with the `txdump` task:

```
--> txdump pg0231v.mag "ID MAG MERR" yes > pg0231v.txd
```

where, in the example above, it is assumed that your photometry output file is called “pg0231v.mag”. The second parameter is a list of fields from the `phot` output file to be extracted.

It may be useful to combine the three text files with the magnitudes to a single file where all the information for each star is contained in a single row. This can be done with the `tjoin` task:

```
--> tjoin file1.txt file2.txt output.txt 1 1
```

(the last two parameters specify that the first column in each input file is used to match the two files). First join the *B* and *V* photometry files to a temporary file, then join this temporary file with the *I* band photometry file.

You should now have a file that looks something like this:

```
--> cat pg0231bvi.txd
#c C1 i %2d
#c C2_1 d %7.5g
#c C3_1 d %6.4g
#c C2_2 d %7.5g
#c C3_2 d %6.4g
#c C2 d %7.5g
#c C3 d %6.4g
 1  14.661  0.004  13.241  0.003  12.655  0.002
 2  15.312  0.006  13.452  0.003  12.377  0.002
 3  13.661  0.002    12.2  0.001  11.599  0.001
 4  14.559  0.003  13.139  0.003  12.524  0.002
 5  15.962  0.011  15.528  0.021  15.988   0.04
 6   16.39  0.016  14.139  0.006  12.397  0.002
```

Your actual magnitudes and errors may be somewhat different from the table above due to your choice of aperture- and background parameters.

Next, we carry out the actual fits. We will assume that the calibration equations can be written as

$$V - v_{i,e} = c_v(b_{i,e} - v_{i,e}) + z_v \quad (3.2)$$

$$B - V = c_{bv}(b_{i,e} - v_{i,e}) + z_{bv} \quad (3.3)$$

$$V - I = c_{vi}(v_{i,e} - i_{i,e}) + z_{vi} \quad (3.4)$$

where capital letters *BVI* indicate standard magnitudes and small letters ($b_{i,e}, v_{i,e}, i_{i,e}$) indicate *extinction-corrected* instrumental magnitudes. So we assume that, in addition to a zero-point offset, the colour terms in our transformations can be expressed as a linear function of the instrumental colours. The choice of the $b-v$ colour for the colour term in the first transformation is somewhat arbitrary; we could also have chosen $v - i$.

- Determine the transformation coefficients in Eq. (3.2) - (3.4). You can use any programme capable of fitting a straight line to a set of $(x, y)$ values. For example, the `linfit` function in IDL:

  ```
  IDL> x = [1,2,3,4,5]
  IDL> y = [5,4,3,2,1]
  IDL> print, linfit(x, y)
            6.00000      -1.00000
  ```

  *Remember to apply extinction corrections to the instrumental magnitudes.* You can use the airmass values in Table 2.1 (taken from the image headers) and assume extinction coefficients of $k_B = 0.180$, $k_V = 0.123$ and $k_I = 0.054$.

  Keep a note of the transformation coefficients for the report

- Check that the fits you have carried out are ok. Are there any outlying data points?

  Make plots showing the standard transformations for the report

  Think of a good way to represent these results yourself.

## 3.3 Aperture photometry of the science field

Now that the standard transformations have been defined, all there is left to do is run `phot` on the science frames and transform the instrumental magnitudes to the standard system.

- In the same way as for the standard field, generate a coordinate list (it's a short list, since there is only one object of interest.. but you are of course welcome to include others if you like). Our target, 4829 Ser, is at $(x, y) \approx 1375, 1043$. However, the offsets from one frame to the other are a bit larger than for the standard field. You may need to allow for a larger centroid shift when running `phot`, or generate a separate coordinate list for each image.

- Run `phot`. Make sure to use the same aperture and background definitions as for the standard stars.

- Apply the standard transformations to the instrumental magnitudes. Remember the extinction correction!

Make a note of the standard magnitudes. Keep these for the report

`phot` includes error estimates on the measured magnitudes, along with information that is used to compute these errors.

- Verify that the error estimates given in the `phot` output files follow from the other information given in the file (cf. Appendix A).
- For your choice of aperture and background annulus, what are the contributions to the total error on the measured (instrumental) $V$-band magnitudes from:
  - Photon poisson noise from the asteroid itself?
  - Background noise within the photometry aperture?
  - Uncertainty on the mean background level?

Include this information in your report.

Two possible sources of background noise are: 1) Poisson noise from the sky background, 2) CCD read-out noise. Can you think of others?

In your report, discuss whether the total background noise measured on the images can be explained as a combination of Poisson noise from the sky background and CCD read noise, or whether additional sources of noise are needed.

## 3.4 Additional steps

We have now finished the most essential steps of the photometric calibration of the images of 4829 Ser. For completeness, we here mention some additional steps that are often undertaken:

**Aperture corrections**

In many cases, it is desirable to measure the aperture magnitudes for the science target in a smaller aperture than the standard stars. Standard stars are generally bright, and can be measured in a large aperture without seriously degrading the accuracy. Using a large aperture for standard stars is desirable in order to avoid being too sensitive to variations in the seeing. In some cases, it is even necessary to defocus the telescope in order to avoid saturation of the standard stars. For science targets, on the other hand, one would usually aim to use an aperture size that maximises the S/N for faint objects.

It may therefore be necessary to apply *aperture corrections* from the aperture used for the standard stars to that used for the science target. These corrections can be determined from bright stars in the science field, by measuring these stars both in a large aperture (similar to that used for the standard stars) and in a smaller one (similar to that used for the science objects). A more sophisticated method is to construct a *curve-of-growth* from measurements in a sequence of many apertures. The `photcal` package in IRAF contains a task called `mkapfile` that can determine such aperture corrections. See the help text to this task for further details.

**PSF-fitting photometry**

If the stellar density is high then it may be impossible to reach a suitable compromise between an aperture size that is large enough to include most of the flux from a star and small enough to avoid including neighbouring stars. In such cases, we talk about *crowded* fields, and more sophisticated techniques than simple aperture photometry are called for.

DAOPHOT solves this problem using *PSF-fitting* photometry. Instead of simply measuring the flux from each star in a circular aperture, the brightness of a star is determined by scaling the PSF until the best match to the observed profile is observed. This scaling is carried out simultaneously to all stars in the image, so that two (or multiple) overlapping stellar images are fitted by scaling multiple PSFs. In crowded fields, this technique can represent a vast improvement over traditional aperture photometry.

The crucial point in PSF photometry is to obtain a good estimate of the PSF itself. This is, in general, not known *a priori* and must be determined from the image itself. Ideally, one would use a number of bright, isolated stars for the PSF determination. If such stars are not available, it may be necessary to adopt an iterative procedure whereby an initial crude estimate of the PSF is obtained and a first pass of the PSF-fitting photometry is carried out. A set of PSF-stars is then selected, and all other stars are subtracted from the image using the initial estimate of the PSF. A better estimate of the PSF can now be obtained, and the fitting procedure repeated. This sequence can be repeated as many times as necessary.

The IRAF version of DAOPHOT contains a task called `allstar` that carries out PSF-fitting photometry on a single image. The stand-alone Fortran version of DAOPHOT also has a task called `allframe` that simultaneously carries out PSF-fitting photometry on an arbitrary number of images (see Stetson 1994: PASP 106, 250).

## 3.5 Summary

In this lesson you have learned:

- How to carry out aperture photometry on a CCD image in IRAF
- How to define a transformation from the instrumental system to a standard system
- How to apply this transformation to your science data

# Chapter 4

# Spectroscopy in IRAF

In this exercise we will use the tools in IRAF to carry out the basic reduction steps for a spectroscopic data set. As usual, we will only scratch the surface here to get a general idea about the steps that are involved in the reduction of spectroscopic data. The relatively manual procedure we follow here gives a good feeling for the main concepts, but can be cumbersome for large datasets. Hence, many spectrographs have dedicated *pipelines* that carry out the reduction more or less automatically. For a general introduction to reduction of spectra in IRAF, see the document *A User's Guide to Reducing Slit Spectra with IRAF* by Phil Massey, Frank Valdes and Jeanette Barnes (`http://iraf.net/irafdocs/spect.pdf`).

The tools we will use are located in the package `noao.imred.specred`, so start by loading this package.

## 4.1 The data

We are again using data from the EMMI instrument. This time EMMI was used as a multi-object spectrograph to obtain spectra of multiple star clusters in the nearby spiral galaxy NGC 5236 (M 83). This was done by 1) inserting a *slitmask* into the focal plane of the telescope, and 2) replace the filters used for imaging with a *grism*. The EMMI detectors then record a spectrum of each individual object. The whole slitmask used here had 22 slitlets, of which some are only included for alignment purposes. In this exercise we will extract and calibrate one object.

Figure 4.1 shows a typical sequence of images taken during the observation of a given target.

1. First, a direct image is taken of the field to ensure that the telescope is pointed in the right direction (left-hand panel). This image is similar to any other normal imaging exposure, with no slit or dispersing element inserted into the light path. In general, a filter with a central wavelength similar to that of the spectra is used (in this case an *R*-band filter). The positions of a few reference stars are measured on this image and adjustments made to the telescope pointing, if necessary.

2. Next, the *slitmask* is inserted into the light path and an exposure made through the mask (centre panel). If the telescope pointing is accurate, the science targets should now appear visible through the slits. You can see them if you look closely at the figure.

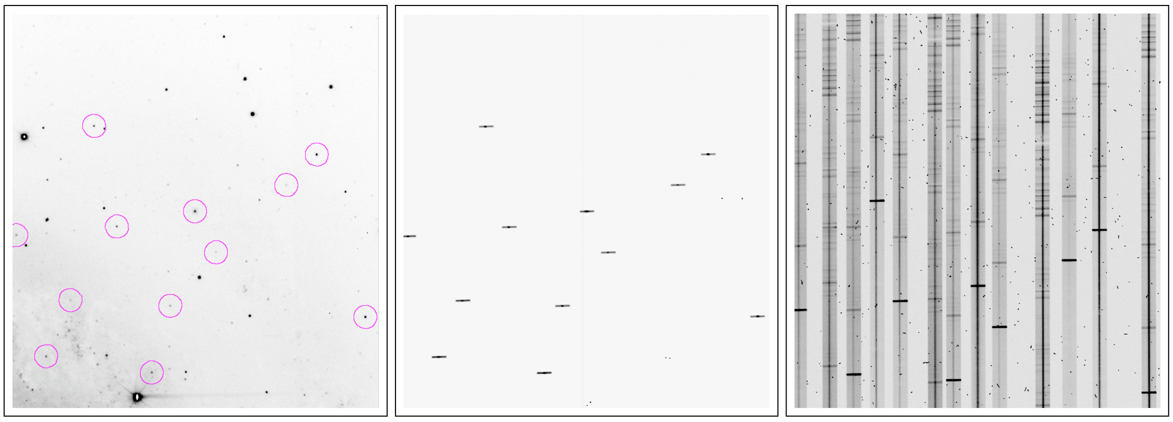
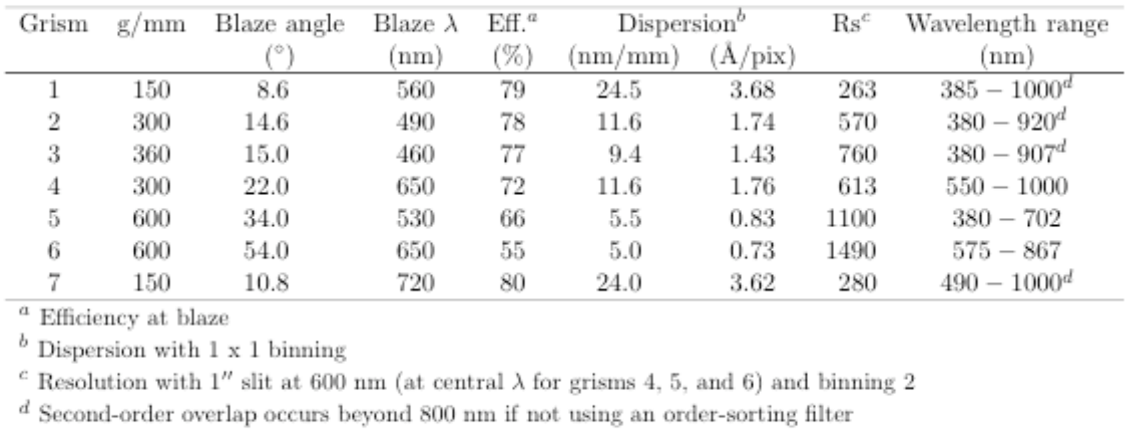


Figure 4.1: Subsections of EMMI spectroscopic dataset. Left: a direct (*R*-band) image of the field with the spectroscopic targets marked. Centre: a "through-slit" image of the field. Right: The spectroscopic exposure.



| Grism | g/mm | Blaze angle | Blaze $\lambda$ | Eff.[a] | Dispersion[b] | | Rs[c] | Wavelength range |
|---|---|---|---|---|---|---|---|---|
| | | (°) | (nm) | (%) | (nm/mm) | (Å/pix) | | (nm) |
| 1 | 150 | 8.6 | 560 | 79 | 24.5 | 3.68 | 263 | 385 – 1000[d] |
| 2 | 300 | 14.6 | 490 | 78 | 11.6 | 1.74 | 570 | 380 – 920[d] |
| 3 | 360 | 15.0 | 460 | 77 | 9.4 | 1.43 | 760 | 380 – 907[d] |
| 4 | 300 | 22.0 | 650 | 72 | 11.6 | 1.76 | 613 | 550 – 1000 |
| 5 | 600 | 34.0 | 530 | 66 | 5.5 | 0.83 | 1100 | 380 – 702 |
| 6 | 600 | 54.0 | 650 | 55 | 5.0 | 0.73 | 1490 | 575 – 867 |
| 7 | 150 | 10.8 | 720 | 80 | 24.0 | 3.62 | 280 | 490 – 1000[d] |

[a] Efficiency at blaze
[b] Dispersion with 1 x 1 binning
[c] Resolution with 1″ slit at 600 nm (at central $\lambda$ for grisms 4, 5, and 6) and binning 2
[d] Second-order overlap occurs beyond 800 nm if not using an order-sorting filter

Figure 4.2: EMMI grisms

3. Finally, the dispersing element (in the case of EMMI, a grism) is inserted and the spectroscopic exposure started. While the direct image and through-slit exposures are typically short, just long enough to check the alignment (typically 30-60 s), most of the observing time will be spent exposing through the grism. At the end, a spectrum of each slitlet is produced (right-hand panel). In this case, an exposure time of 2600 s was used.

Fig. 4.2 lists the basic properties of grisms in EMMI. The data used here were taken with Grism #3.

The data for this exercise are available at:
`http://www.astro.uu.nl/~larsen/teaching/OA1/data/spectro.tar.gz`

- In the same way as for the photometry exercise, make a directory for these data, download and unpack the tar archive.

## 4.2 Initial reduction

The initial reduction of spectroscopic data follows the same basic steps as for photometric data. Bias is subtracted, and the image is then divided by a flat-field. However, flat-fielding of spectroscopic data can be somewhat tricky, since a sky-flat would essentially be a solar spectrum, so not very "flat"! Alternatively, dome flats can be used, or (as is often the case), special lamps are built into the spectrograph for this very purpose.

In this exercise you will be given data that have already been bias subtracted and flat-fielded (using flat-fields from the built-in continuum lamps in EMMI). The `cosmicrays` task in the `noao.imred.crutil` package has been used to remove most of the many cosmic-ray hits that have occurred during the science exposure (visible in Fig. 4.1).

After unpacking the tar archive, you should find the following files:

- n5236_direct.fits: The direct image of NGC 5236
- n5236_slits.fits: Through-slit image
- n5236_spec.fits: bias-subtracted and flat-fielded spectroscopic exposures
- n5236_arcs.fits: exposure of arc lamps (for wavelength calibration)
- LTT2415a.fits: exposure of a flux standard star (for flux calibration)
- lines.lst: A list of calibration line wavelengths for the HeAr lamp in EMMI

## 4.3 Extracting the science spectrum

Start by displaying the spectroscopic exposure (`n5236_spec.fits`) in `ds9`. You should see the spectrum of each slit as a vertical band, with the spectrum of the science target itself roughly in the middle of this band. You may need to adjust the contrast settings in `ds9` to see the spectra clearly. The many short horizontal lines are sky lines, which need to be subtracted out. There may also still be a few cosmic-ray hits left that `cosmicrays` didn't manage to remove.

We will extract the spectrum of one of the brighter objects in the slitmask, which appears at the centre of the direct and through-slit images in Fig. 4.1. The slit that covers this object is centred at $(x, y) = (1476, 1096)$ on the through-slit image, so the spectrum is found around $x = 1476$. In the through-slit image you'll note that the detector has a bad column at $x = 1469$, but fortunately this just misses the spectrum of our target. At the blue end of the spectrum where this becomes particularly bothersome, we have interpolated over it with the `fixpix` task. Of course, in an ideal world we would not try to "fix" bad data, but simply exclude it from the analysis. In practice, you'll often find that such work-arounds are necessary, and in this case justifiable since we are just interpolating over columns that, in any case, for the most part contain signal from the sky.

You can also find the object in the direct image at $(x, y) = (1388, 1172)$ (note the significant shift with respect to the through-slit image).

We now need to *extract* the spectrum. In principle, this is equivalent to measuring the number of counts from a star in aperture photometry, except that we now want to do this for each wavelength sampling point on the spectrum. Hence, the basic steps are as follows:

- We will first need to tell IRAF where to look for the spectrum. Instead of a set of $(x, y)$ coordinates, we will just need to specify the column where the spectrum is found.
- Since the spectra are not dispersed *exactly* along the columns of the CCD, we need to *trace* them. This is done by fitting a high-order polynomial to the centroid of the spectrum vs. $y$-coordinate.
- Once these steps have been taken, the spectrum can be extracted. Similar to the case of aperture photometry, we choose an *aperture* (in this case, one-dimensional) and a *background window* on either side of the aperture. For each wavelength sampling point, the counts within the aperture are added, and the background level subtracted. The output of this process is a table of counts versus CCD $y$-coordinate.

These steps are all carried out with the `apall` task in IRAF. For your convenience, the help text for this task is listed in Appendix B. Below we discuss some of the most important parameters.

First, however, we need to set a global parameter in the `specred` package:

- `epar specred` - set `dispaxis=2`. This specifies that the spectra are dispersed along the $y$-axis of the image (as opposed to the $x$-axis)
- Now take a look at the `apall` parameter file (`-->` `epar apall`). Some of the most important parameters are:
  - `input` - (list of) input images. In this case just a single image, `n5236_spec`.
  - `output` - (list of) output spectra. If left empty, `apall` will append `.ms` to the input name. That's fine.
  - `format` - specifies the output format. If `onedspec`, multiple spectra extracted from a single image will be stored in separate files. For `multispec`, all spectra will be stored in the same output file. We recommend selecting `multispec` here.
  - `references` and `profiles` - leave blank for now
  - Next follows a set of boolean parameters that define what you actually want `apall` to do. We want to run it interactively, so set `interactive=YES`, but we want to define the apertures ourselves so set `find=NO`, `recenter=NO`, `resize=NO` and `edit=YES`. We also want to trace the apertures along the spectrum, so `trace=YES` and `fittrace=YES`, and we want to extract the spectra so `extract=YES`, `extras=YES` and `review=YES`.
  - Under `DEFAULT APERTURE PARAMETERS` you define the extraction aperture via the `lower` and `upper` parameters (equivalent to the `apertures` parameter in `phot`). It may be useful to inspect the image with `imexamine` (the "j" key will carry out a

local Gaussian fit to a row and show a plot). For the EMMI spectra used here, good values might be ∼ 5 pixels. This can be redefined interactively later. The `implot` task may also be useful in inspecting the data to decide on the aperture parameters.

- Under `DEFAULT BACKGROUND PARAMETERS` the background fitting parameters are defined. The options here are a little different from the photometry case. `b_funct` is the function used to interpolate between the background windows and the aperture. Different choices are available: `chebyshev` or `legendre` polynomials, or first or third-order spline functions (`spline1`, `spline3`). `b_order` is the order of the polynomial or the number of segments in the spline.

  *Note:* due to the way these polynomials are defined, `b_order=1` is a constant value, `b_order=2` is a straight line, etc..

  `b_sampl` sets the background sampling region. So for example, `b_sampl = -15:-6,7:12` indicates that one background window goes from −15 to −6 pixels with respect to the aperture centre, and the other one from 7 to 12 pixels from the centre.

  For the EMMI data, we clearly want to make sure that the background windows remain inside each slitlet!

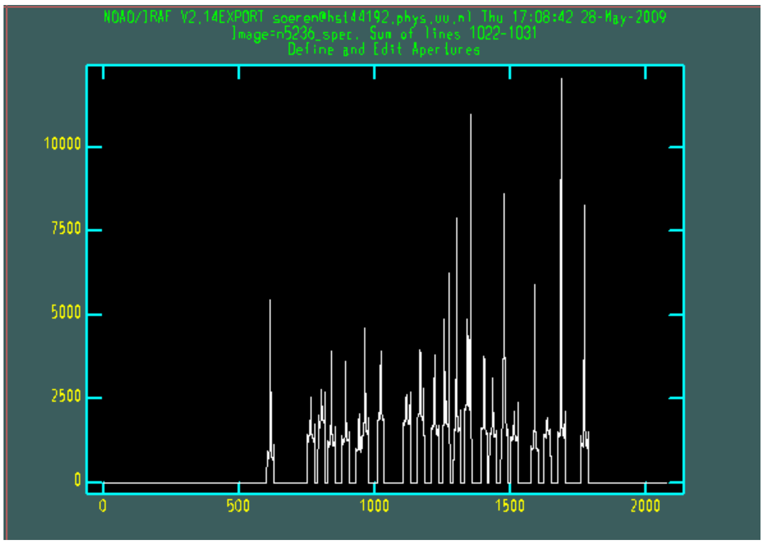
  Make a note of the aperture and background parameters you select and include this in the report. Remember to justify your choices.

- Under `EXTRACTION PARAMETERS` you want to set `background = fit`.
- The `weights` parameter can be set to `none` or `variance`. For `none` the pixels within the aperture are simply added up. For `variance` the spectra are extracted using the *optimal extraction algorithm* described in Horne, K. (1986: PASP, 98, 609). If you want to use this option, you need to set the corresponding CCD parameters (`readnoi`, `gain`). For the purpose of this exercise, `weights=none` will be just fine.

• With these parameters set, we are now ready to go! So start apall by typing

```
--> apall n5236_spec
```

You should get a window up that looks like this:


NOAO/IRAF V2.14EXPORT soeren@hst44192.phys.uu.nl Thu 17:08:42 28-May-2009
Image=n5236_spec, Sum of lines 1022-1031
Define and Edit Apertures
10000
7500
5000
2500
0
0
500
1000
1500
2000

You are now working inside the *aperture editor* of `apall`. It starts by showing a cut across the centre of the image (if you left `line` at INDEF in the `apall` parameters). If we had asked `apall` to automatically find apertures, they would have been indicated on the plot. However, since we disabled that option we need to define our extraction aperture manually.

As in other tasks in IRAF that interact with the graphics display, you can enter various commands directly here using the keyboard. By pressing "?" (make sure the cursor is inside the graphics window) you will get a list of commands (in the main `pyraf` windows). This will produce the following output:

```
                  APEXTRACT CURSOR KEY SUMMARY

?  Print help               j  Set beam number        u  Set upper limit(s)
a  Toggle all flag          l  Set lower limit(s)     w  Window graph
b  Set background(s)        m  Mark aperture          y  Y level limit(s)
c  Center aperture(s)       n  New uncentered ap.     z  Resize aperture(s)
d  Delete aperture(s)       o  Order ap. numbers      I  Interrupt
e  Extract spectra          q  Quit                   +  Next aperture
f  Find apertures           r  Redraw graph           -  Previous aperture
g  Recenter aperture(s)     s  Shift aperture(s)      .  Nearest aperture
i  Set aperture ID          t  Trace aperture(s)
```

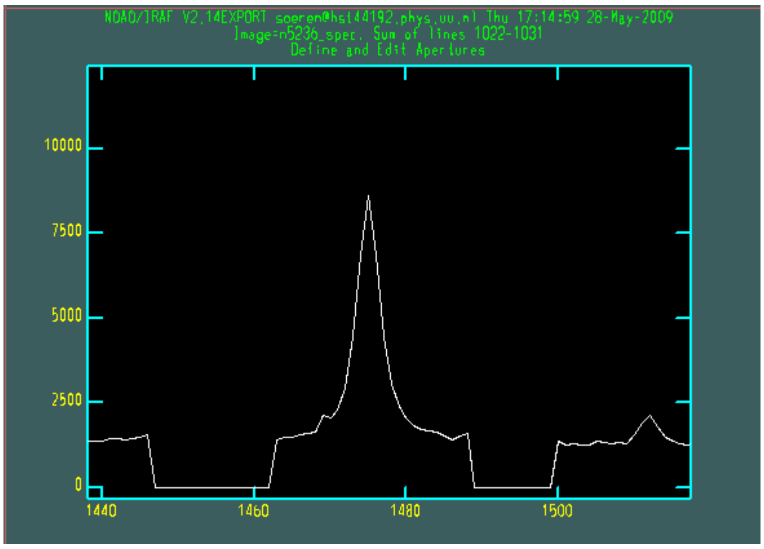
(The cursor will now have moved to the main `pyraf` window, and you need to "q" the help viewer to get back to the aperture editor).

The first thing we want to do is to zoom in on the columns of interest. Recall that our spectrum is at $x \approx 1476$.

- Use the "window" function to zoom in on the relevant spectrum. That is, first press "w" to redefine the window, then "j" to set the new left border of the window (with the cursor) or "k" to set the new right border. As usual, you can press the "?" key after "w", which will give you a list of keystrokes accepted.

*Note: the "w" (window) function only accepts one key at the time. So to zoom in, you need to press "w", then "j", then "w" again, then "k", etc..*

After zooming in, the display might look as follows:



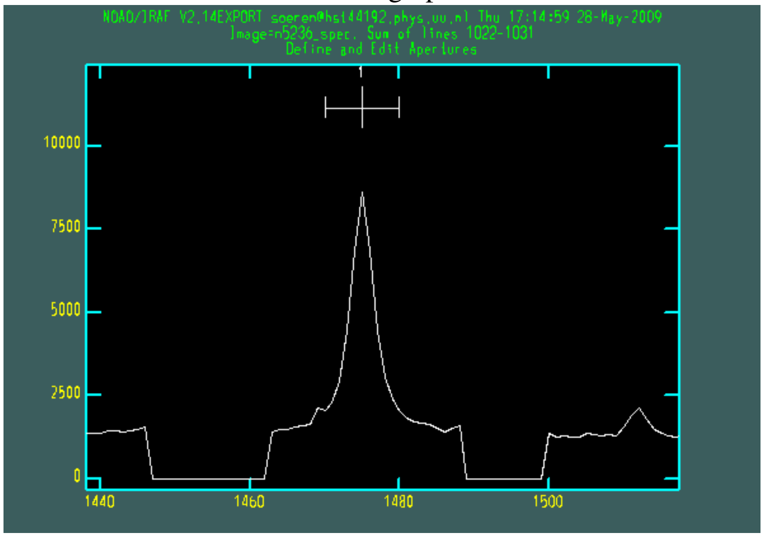
```
NOAO/IRAF V2.14EXPORT soeren@hs144192.phys.uu.nl Thu 17:14:59 28-May-2009
Image=n5236_spec. Sum of lines 1022-1031
Define and Edit Apertures
```

We can now clearly see the peak of the spectral trace at $x \approx 1476$ and the edges of the slit at $x \approx 1463$ and $x \approx 1488$. Then there is a gap, and the next slits follow.

It is now time to define the extraction aperture.

- Move the cursor near the feature of interest, and use the "m" key to mark a new aperture.

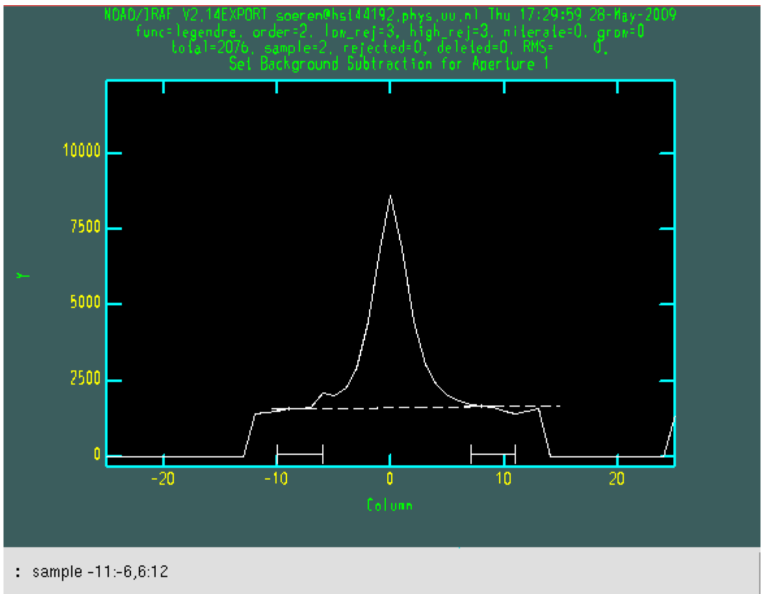
The new aperture will be now be marked on the graph:



```
NOAO/IRAF V2.14EXPORT soeren@hs144192.phys.uu.nl Thu 17:14:59 28-May-2009
Image=n5236_spec. Sum of lines 1022-1031
Define and Edit Apertures
```

The aperture can be redefined using the `:lower` and `:upper` commands, or interactively by simply pressing "l" or "u" while pointing the cursor at the new limits. If we wanted to extract more spectra, we could define additional apertures by repeatedly using the "m" command.

Next, we need to inspect the background windows. We therefore hit the "b" key. By doing so, we (temporarily) leave the aperture editor, and instead enter IRAF's generic `icfit` task with its own set of commands (sorry folks..):



```
NOAO/IRAF V2.14EXPORT soeren@hst44192.phys.uu.nl Thu 17:29:59 28-May-2009
func=legendre, order=2, low_rej=3, high_rej=3, niterate=0, grow=0
total=2076, sample=2, rejected=0, deleted=0, RMS=     0.
Set Background Subtraction for Aperture 1
```

```
: sample -11:-6,6:12
```

You can now see the background regions and a dashed line representing the interpolation function. These can both be redefined now, if you like, by using the `:sample` and `:order` commands. Just typing `:sample` will showing the current sample region, or `:sample -11:-6,6:12` will specify the background sampling regions accordingly (as in the screenshot above). Remember to hit the "f" (for fit) key after making changes.

Once you are happy with the background windows, hit "q" to exit the `icfit` routine and get back to the aperture editor. We are now ready to trace the spectrum.

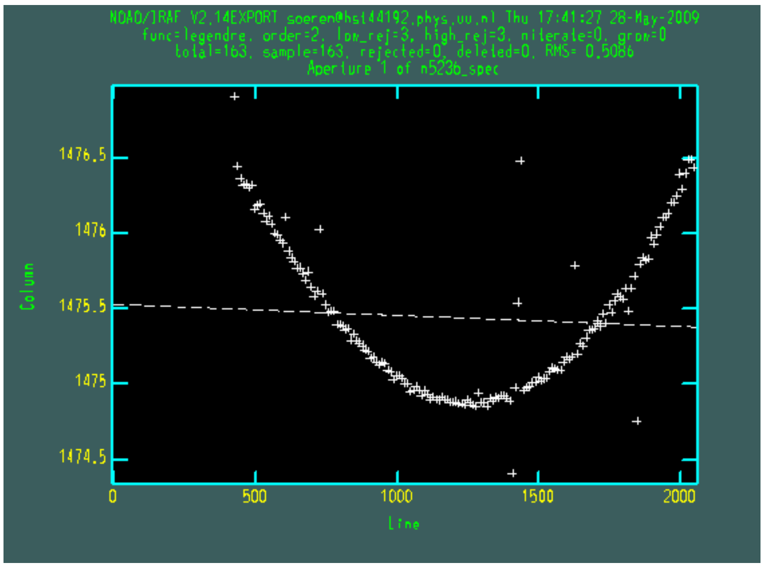
- Type "q" to quit the aperture editor.

  `apall` will now ask: `Trace apertures for n5236_spec?` to which we answer "y" (enter).

  Next, it asks: `Fit traced positions for n5236_spec interactively?` and we again say "y" (enter).

  It then asks: `Fit curve to aperture 1 of n5236_spec interactively?` And of course, we say "y" to that too.

We are now back in the `icfit` routine, and get a plot like this:



```
NOAO/IRAF V2.14EXPORT soeren@hst44192.phys.uu.nl Thu 17:41:27 28-May-2009
func=legendre, order=2, low_rej=3, high_rej=3, niterate=0, grow=0
total=163, sample=163, rejected=0, deleted=0, RMS= 0.5086
Aperture 1 of n5236_spec
```

The plus symbols show the measured trace, and the dashed line is the best fit. Obviously, the fit is not (yet) very good. From the header, we can see that the current fit is for a 2nd order legendre polynomial, so let's try a third order order polynomial instead: `:order 3`, followed by "f". The fit now looks much better! There are a few outlying data points, which you can exclude from the fit with the "d" key if you like, but for the example here it won't make much difference. We are now ready to proceed further.

- When you are happy with the fit, hit the "q" key to exit from `icfit`, and `apall` will ask:

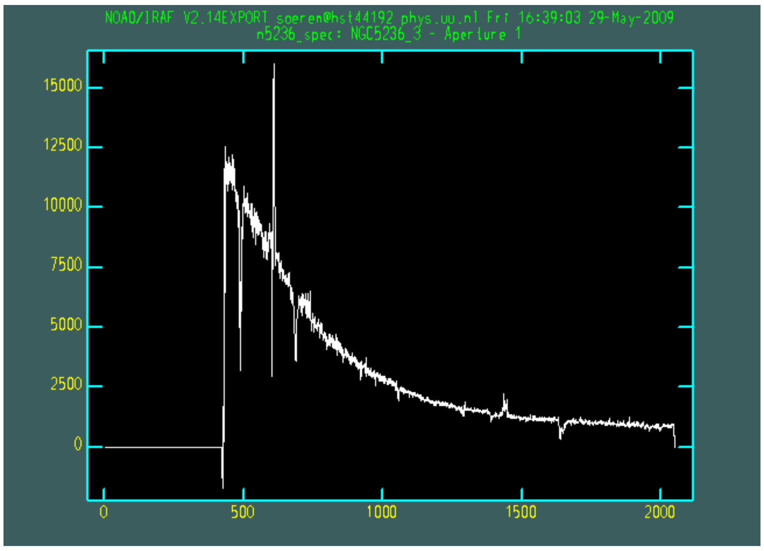
  `Write apertures for n5236_spec to database` - "y"

  `Extract aperture spectra for n5236_spec?` - "y"

  `Review extracted spectra from n5236_spec?` - "y"

  `Review extracted spectrum for aperture 1 from n5236_spec?` - "y"

`apall` will now plot the extracted spectrum:



We clearly see the overall shape of the continuum of the spectrum. There are also some absorption lines visible, but some of the "features" are actually cosmic ray hits that have not been fully removed by `cosmicrays`. If these happen to fall within the science aperture they will appear positive, but if they fall in the background windows they will appear negative. These could be removed by combining multiple spectra.

This is just a quick-look plot with no interactive functions, so the only thing left to do is hit "q" and we're done!

## 4.4 Wavelength calibration

The next step will be to define a *wavelength calibration* and apply it to the data. This is done by observing a set of calibration lamps mounted in EMMI, and fitting a high-order polynomial (or spline function) to wavelength versus pixel offset. This function can then be applied to the science data. The line list for the He+Ar lamps used in EMMI is given in Appendix C. In practice, the task of matching a set of observed emission lines to a list of wavelengths can be a source of much frustration, especially if there is no information about their relative strengths. Even if such information is available, it may not be accurate because the lamps take time to warm up and during this process the line strengths can vary. So there can be quite a bit of trial-and-error involved until a sufficient number of lines in the list have been matched with the calibration spectrum to get a useful solution. To make life a bit easier for you, plots of the EMMI calibration line spectra are given in Appendix D.

### 4.4.1 Extracting the calibration spectrum

- Start by displaying the calibration frame, `n5236_arcs` in `ds9`.

Describe what you see.

We now need to extract the spectrum of the calibration lamp for the proper slit. Fortunately, this is easily done since `apall` wrote the information about the apertures and trace function used for our science data to a *database* (in case you are curious, this information is kept in the `database/` subdirectory which has been created by `apall`). So we don't have to do anything interactively, we just have to tell `apall` that we want to use the aperture database created for the `n5236_spec` image. And, of course, we do *not* want to subtract any background from the calibration spectrum.

- In `apall`, we now set the `references` parameter to `n5236_spec`. This indicates that we will use the apertures defined for `n5236_spec` to extract a spectrum from the input image.
- We set `interactive` to NO, and likewise for the `edit`, `trace` and `fittrace` parameters.
- Under `EXTRACTION PARAMETERS`, set `background` to NONE.
- We are now ready to run `apall`:

  ```
  --> apall n5236_arcs
  ```

  This time you won't get any plot, but you should now find the spectrum (`n5236_arcs.ms.fits`) in the data directory.
- You can use the task `splot` to look at the spectrum:

  ```
  --> splot n5236_arcs.ms
  ```

  This task has a lot of different commands that allow you to examine a spectrum interactively. You can measure the flux from emission lines, the centroid of lines, etc. Press "?" for a list of cursor commands. Of course, many of these are only useful for spectra that are calibrated in wavelength and/or flux. We'll get back to `splot` later.

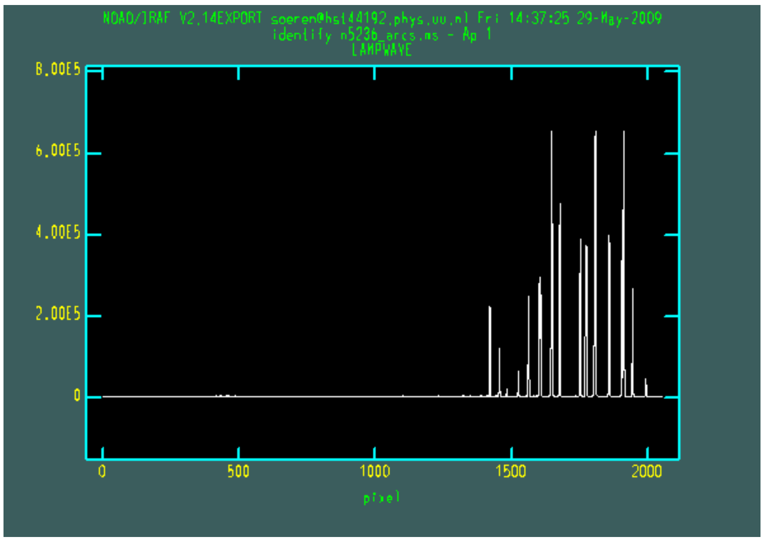
### 4.4.2 Defining the wavelength solution

The actual wavelength solution is established with a separate task, called `identify`. This task takes as input a calibration line spectrum and a list of wavelengths, and allows you to interactively carry out a fit. This fit can then later be applied to the science spectra.

- Look at the `identify` parameters. For now, the only ones to worry about are `images` (n5236_arcs.ms) and `coordlist` (lines.lst).
- You can now start `identify`

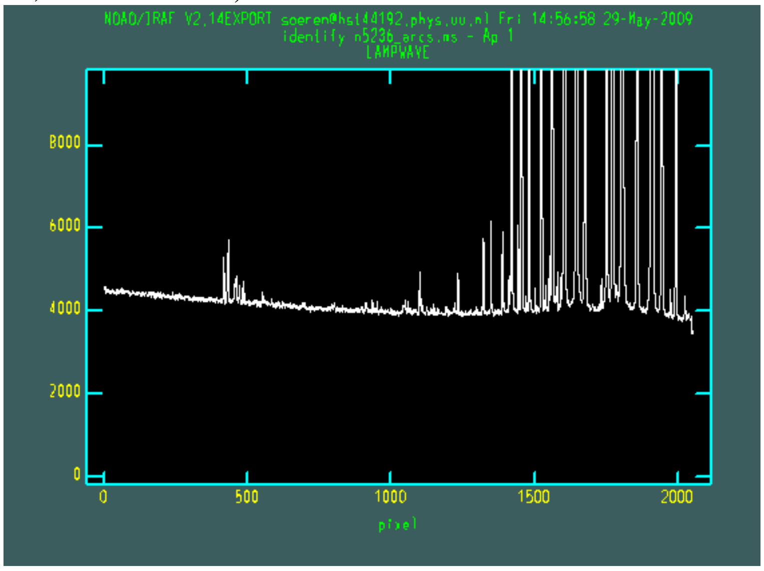
  ```
  --> identify n5236_arcs.ms
  ```

  and should get a plot like this:



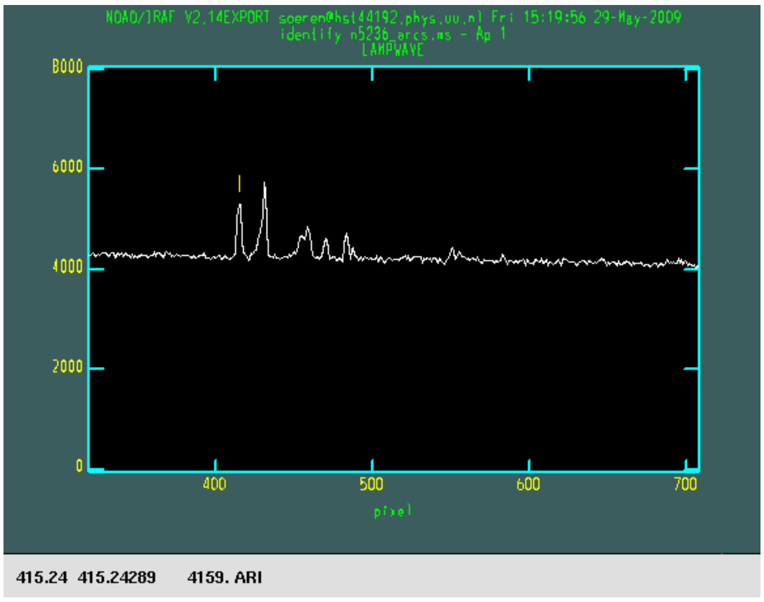
This just shows the spectrum of the calibration lamp in counts versus pixel. We now have to assign wavelengths to as many individual lines as possible in order to get a reliable solution. From the figures in Appendix D you can see that the strongest lines are concentrated at long wavelengths. At wavelengths shorter than ∼ 7000Å the lines get weaker, and are not visible at the default scaling of the plot. However, it is essential to also include these weaker lines in order to get a good calibration over the whole wavelength range.

- Use the `window` function to make the weak lines more clearly visible (w+t sets the top of the window, w+b the bottom). You should now be able to see the weaker lines:



Comparing with Appendix D, we can now identify the "cluster" of lines around 4000–4500Å. Some of these are known Ar lines, while others were identified manually and added to the list by hand.

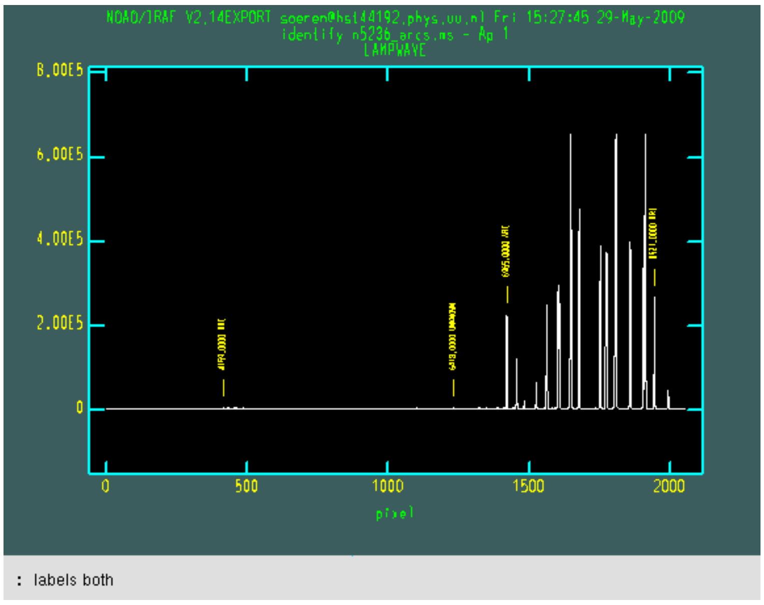
- Use the “m” key to mark features in the calibration spectrum. If you position the cursor near a feature and press “m”, `identify` will calculate the centroid of the feature. If you then enter an approximate wavelength, `identify` will select the closest match in the line list.



```
NOAO/IRAF V2.14EXPORT soeren@hst44192.phys.uu.nl Fri 15:19:56 29-May-2009
identify n5236_arcs.ms - Ap 1
LAMPWAVE
```

```
415.24  415.24289      4159. ARI
```

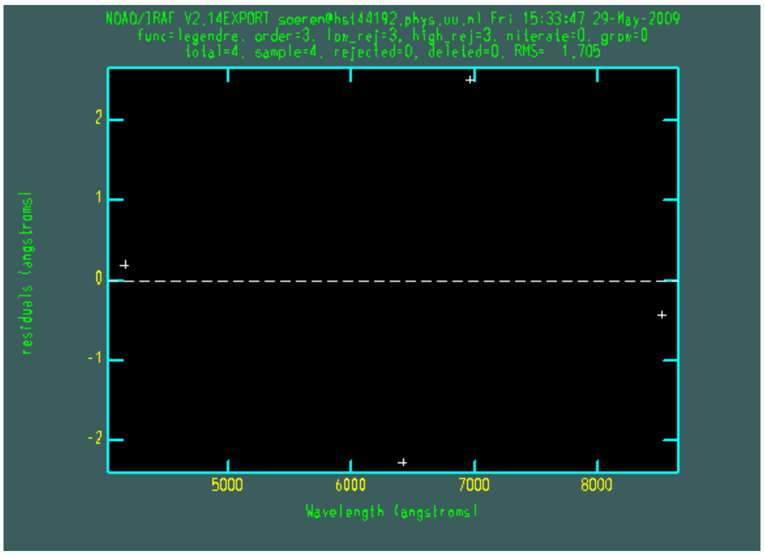
In the figure above, we have marked the Ar feature at 4159 Å. You can use the window function to zoom in further. The combination w+a will return to the autoscaling of the plot used initially. You can use the `:labels both` command to label the lines you have marked:



```
NOAO/IRAF V2.14EXPORT soeren@hst44192.phys.uu.nl Fri 15:27:45 29-May-2009
identify n5236_arcs.ms - Ap 1
LAMPWAVE
```

```
: labels both
```

- Once you have a reasonable number of features identified (3–5) it is time to try doing a first fit. Use the “f” key to enter the `icfit` routine. As usual, you can use the `:func` and `:order` commands to change the fitting function and order. The figure below shows a fit for a 3rd order legendre polynomial function.

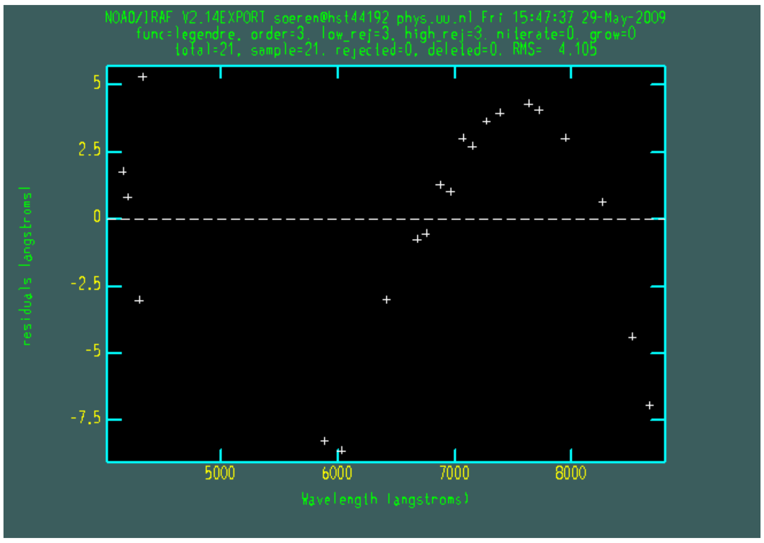


```
NOAO/IRAF V2.14EXPORT soeren@hsi44192.phys.uu.nl Fri 15:33:47 29-May-2009
func=legendre, order=3, low_rej=3, high_rej=3, niterate=0, grow=0
total=4, sample=4, rejected=0, deleted=0, RMS=  1.705
```

  You can see that the r.m.s. scatter is about 2Å, so at this point there is no need to go to a higher order polynomial. Note that you can use the h–l keys to display different quantities on the *x*- and *y*-axis of the plot within `icfit` (“?” for help).

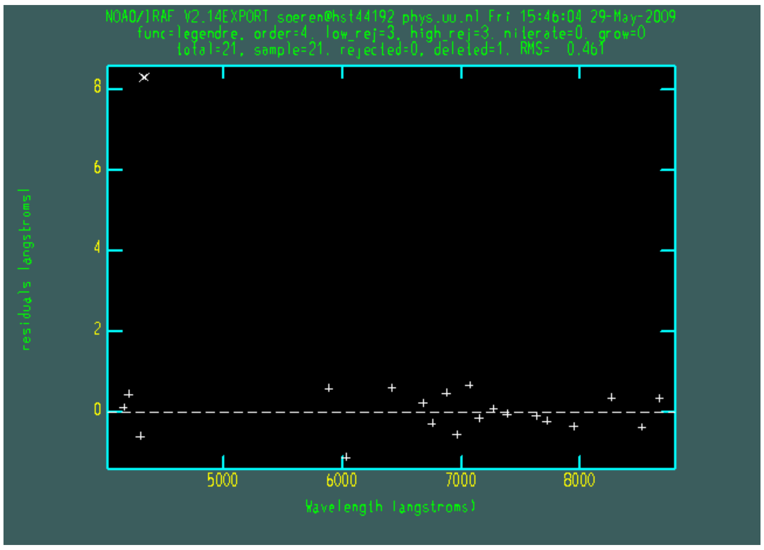
- Now exit `icfit` with the “q” task. You’ll see that the *x*-axis on the plot has now changed to wavelength units! This makes it much easier to identify additional features from the line list. In fact, if you now mark a feature with “m”, `identify` will look for the best match in the line list, and all you have to do is accept it by pressing return. You can even try using the “l” key, which will tell `identify` to go through the whole line list and look for matching features in the spectrum.

- When you have a larger number of features identified, go back to `icfit` with the “f” key. It is now clear, with a larger number of data points, that a 3rd order legendre polynomial doesn’t fit well:



```
NOAO/IRAF V2.14EXPORT soeren@hst44192 phys.uu.nl Fri 15:47:37 29-May-2009
func=legendre, order=3, low_rej=3, high_rej=3, niterate=0, grow=0
total=21, sample=21, rejected=0, deleted=0, RMS=  4.105
```

However, after deleting a single outlying point, a 4th order fit works quite nicely:



```
NOAO/IRAF V2.14EXPORT soeren@hst44192 phys.uu.nl Fri 15:46:04 29-May-2009
func=legendre, order=4, low_rej=3, high_rej=3, niterate=0, grow=0
total=21, sample=21, rejected=0, deleted=1, RMS=  0.461
```

- Once you're happy with your fit, quit `icfit` and `identify`. You'll be asked to confirm that you want to save the solution in the database.

- That's it!

### 4.4.3 Apply the wavelength solution to the science data

This consists of two steps:

1. Assign a given wavelength solution to the science data

2. Apply the solution

The first step is done with the task `refspec`:

- Use `refspec` to assign your newly derived wavelength solution to the science data. `refspec` can take lists of input and reference spectra as input and automatically assign the "best" reference spectrum to each science spectrum based on various criteria. In our case things are simple because we only have one science and one reference spectrum, so most of these options aren't relevant. Apart from the `input` and `references` options, you can leave the `sort` and `group` keys empty, and set `ignoreaps` to NO (this would actually be important only if we had extracted multiple spectra from the data).

The second step is done with the task `dispcor`:

- Use `dispcor` to apply the wavelength solution to the science spectrum.
- Done!

## 4.5 Inspecting the spectrum interactively

Now that we have calibrated the wavelength scale of our spectrum, it is time to look at it again. The `splot` task has many built-in functions that are useful for interactive analysis of a spectrum, here will just look at a couple of examples:

- Use the `splot` task to interactively view the science spectrum.

In addition to the science spectrum itself, also the background has been stored in the file by `apall`. The file has two *bands*, one for the science spectrum and one for the background. In `splot`, you can select a different band by pressing "%", followed by the number of the band (1=science, 2=background)

- Look at the background spectrum with `splot`

We can use the background spectrum to check our wavelength calibration. The sky background often shows strong [O I] emission lines at 5577.35Å and 6300.34Å. It is not uncommon to find that adjustments to the wavelength scale are necessary. Typically, this can be due to mechanical flexure in the instrument, or differences in the light path for the science target and the calibration lamps. Shifts in the wavelength scale can be applied with the `specshift` task.

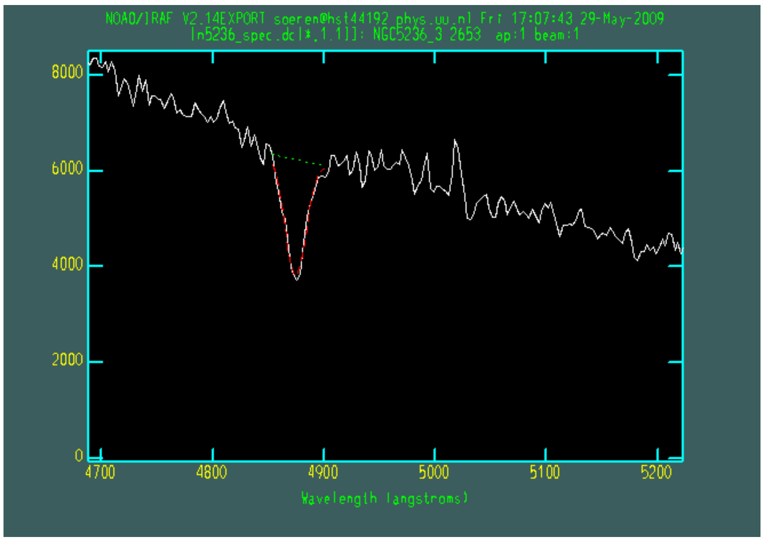
- Use `splot` to measure the wavelengths of the [O I] sky lines. You can use the "k" key to carry out a fit to lines (k+g fits a Gaussian).

  Based on your measurements, give an assessment of the systematic error on the wavelength calibration. Include this in the report.

- If necessary, use `specshift` to apply a shift to the wavelength scale of the spectrum. You may need to adopt an average value based on the two [O I] lines.

  Mention any shift you apply in the report.

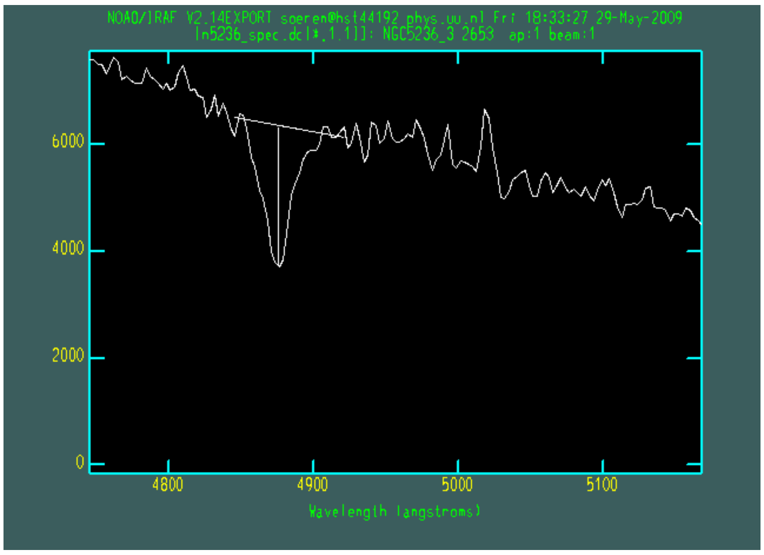
We can now try to identify some features in the science spectrum. This is a spectrum of a young star cluster, so we expect strong Balmer absorption lines. One of these is the H$\beta$ line at rest wavelength 4861Å. The figure below shows the spectrum near H$\beta$:



- Use `splot` to measure the wavelength of the H$\beta$ feature.

| What is the observed wavelength of H$\beta$? Use this to estimate the radial velocity of the cluster. |
|---|

- Use the "e" key to measure the equivalent width of the H$\beta$ line. You need to first position the cursor at the continuum level on one side of the line, press "e", then point at the continuum level on the other side of the line, and press "e" again"



| What is the equivalent width of the H$\beta$ line? |
|---|

## 4.6 Additional steps

There are many additional steps that one could take after this initial reduction and inspection of the spectra. For one thing, we have not applied any flux calibration to the data. If you would like to do this, you can play around with the `standard`, `sensfunc` and `calibrate` tasks in the `specred` package. The package includes a spectrum of a flux standard star, LTT2415a.fits. However, as noted above, accurate flux calibration of spectra can be difficult. EMMI does not have an atmospheric dispersion corrector, and since these are multi-slit spectra this constrains the slit orientation. Therefore, wavelength-dependent slit losses are unavoidable.

For radial velocity measurements, velocities are usually referred to a heliocentric system of rest. There are tasks in IRAF to remove the orbital velocity of the Earth (e.g. `rvcorrect` in the `astutil` package).

As with direct images, it is common practice to combine multiple spectra to get rid of remaining cosmic-ray hits.

Beyond this, much will depend on the specific science question you are addressing. That is a topic for another course.

## 4.7 Summary

In this lesson you have learned:

- How to extract a spectrum imaged onto a CCD detector
- How to perform wavelength calibration of the spectrum
- Carry out a few basic analysis steps in IRAF.

# Chapter 5

# The final report

At the end of the course you should hand in a report about these exercises. Your final grade for the course will be a combination of the written exam (max 100 points) and the report (max 50 points). This report should contain:

- A brief description of the data used in the exercises.
- The steps you have followed when reducing your data - how did you go from the raw CCD images to the data used in the photometry exercise? Discuss bias correction, flat-fielding, and other relevant issues. To get a high grade, include some general discussion of CCD image reduction and the properties of CCD detectors.
- The steps you have followed when carrying out photometry on the asteroid 4829 Ser.
- In the spectroscopy exercise, discuss the steps involved in extracting and calibrating a spectrum.
- In Chapter 3 and 4, some points to be included in the report are written inside $\boxed{\text{boxes}}$.

# Appendix A

# PHOT output format

```
OUTPUT

    In interactive mode the following  quantities  are  printed  on  the
    standard  output  as each object is measured. Err is a simple string
    indicating whether or not an error was  detected  in  the  centering
    algorithm,  the  sky  fitting algorithm or the photometry algorithm.
    Mag are the magnitudes in apertures 1  through  N  respectively  and
    xcenter,  ycenter and msky are the x and y centers and the sky value
    respectively.

        image  xcenter  ycenter  msky  mag[1 ... N]  error

    In both interactive and batch mode full output  is  written  to  the
    text  file output. At the beginning of each file is a header listing
    the current values of the parameters when the first  stellar  record
    was  written.  These  parameters  can  be subsequently altered. For
    each star measured the following record is written

            image  xinit  yinit  id  coords  lid
               xcenter  ycenter  xshift  yshift  xerr  yerr  cier error
               msky  stdev  sskew  nsky  nsrej  sier  serror
               itime  xairmass  ifilter  otime
               rapert  sum  area  mag  merr  pier  perr

    Image and coords are the name  of  the  image  and  coordinate  file
    respectively.  Id  and lid are the sequence numbers of stars in the
    output and coordinate files respectively. Cier and  cerror  are  the
    centering  algorithm  error  code  and  accompanying  error  message
    respectively.  Xinit, yinit, xcenter, ycenter, xshift,  yshift,  and
    xerr,  yerr  are  self  explanatory  and output in pixel units.  The
```

sense of the xshift and yshift definitions is the following.

```
        xshift = xcenter - xinit
        yshift = ycenter - yinit
```

Sier and serror are the sky fitting error code and accompanying error message respectively. Msky, stdev and sskew are the best estimate of the sky value (per pixel), standard deviation and skew respectively. Nsky and nsrej are the number of sky pixels and the number of sky pixels rejected respectively.

Itime is the exposure time, xairmass is self-evident, ifilter is an id string identifying the filter used in the observations, and otime is a string containing the time of the observation in whatever units the user has set up.

Rapert, sum, area, and flux are the radius of the aperture in scale units, the total number of counts including sky in the aperture, the area of the aperture in square pixels, and the total number of counts excluding sky in the aperture. Mag and merr are the magnitude and error in the magnitude in the aperture (see below).

```
        flux = sum - area * msky
         mag = zmag - 2.5 * log10 (flux) + 2.5 * log10 (itime)
        merr = 1.0857 * err / flux
         err = sqrt (flux / epadu + area * stdev**2 +
               area**2 * stdev**2 / nsky)
```

Pier and perror are photometry error code and accompanying error message.

In interactive mode a radial profile of each measured object is plotted in the graphics window if radplots is "yes".

In interactive and batchmode a radial profile plot is written to plotfile if it is defined each time the result of an object measurement is written to output .

# Appendix B

# APALL help

```
APALL (Sep96)            noao.twodspec.apextract          APALL (Sep96)



NAME
    apall -- Extract one dimensional sums across the apertures


USAGE
    apall input


PARAMETERS

    input
        List of input images.

    output = ""
        List  of  output  root names for extracted spectra.  If the null
        string is given or the end of the output list is reached  before
        the  end  of the input list then the input image name is used as
        the output root name.  This will not  conflict  with  the  input
        image  since  an aperture number extension is added for onedspec
        format,  the  extension  ".ms"  for  multispec  format,  or  the
        extension ".ec" for echelle format.

    apertures = ""
        Apertures  to  recenter,  resize, trace, and extract.  This only
        applies  to  apertures  read  from  the  input   or   reference
        database.   Any new apertures defined with the automatic finding
```

algorithm or interactively are always selected. The syntax is a list comma separated ranges where a range can be a single aperture number, a hyphen separated range of aperture numbers, or a range with a step specified by "x<step>"; for example, "1,3-5,9-12x2".

format = "multispec" (onedspec|multispec|echelle|strip)
Format for output extracted spectra. "Onedspec" format extracts each aperture to a separate image while "multispec" and "echelle" extract multiple apertures for the same image to a single output image. The "multispec" and "echelle" format selections differ only in the extension added. The "strip" format produces a separate 2D image in which each column or line along the dispersion axis is shifted to exactly align the aperture based on the trace information.

references = ""
List of reference images to be used to define apertures for the input images. When a reference image is given it supersedes apertures previously defined for the input image. The list may be null, "", or any number of images less than or equal to the list of input images. There are three special words which may be used in place of an image name. The word "last" refers to the last set of apertures written to the database. The word "OLD" requires that an entry exist and the word "NEW" requires that the entry not exist for each input image. Input images without/with a database entry are skipped silently.

profiles = ""
List of profile images for variance weighting or cleanning. If variance weighting or cleanning a profile of each aperture is computed from the input image unless a profile image is specified, in which case the profile is computed from the profile image. The profile image must have the same dimensions and dispersion and it is assumed that the spectra have the same position and profile shape as in the object spectra. Use of a profile image is generally not required even for faint input spectra but the option is available for those who wish to use it.

## PROCESSING PARAMETERS

interactive = yes

Run this task interactively? If the task is not run interactively then all user queries are suppressed and interactive aperture editing, trace fitting, and extraction review are disabled.

find = yes
Find the spectra and define apertures automatically? In order for spectra to be found automatically there must be no apertures for the input image or reference image defined in the database.

recenter = no
Recenter the apertures?

resize = no
Resize the apertures?

edit = yes
Edit the apertures? The interactive parameter must also be yes.

trace = yes
Trace the apertures?

fittrace = yes
Interactively fit the traced positions by a function? The interactive parameter must also be yes.

extract = yes
Extract the one dimensional aperture sums?

extras = no
Extract the raw spectrum (if variance weighting is used), the sky spectrum (if background subtraction is used), and sigma spectrum (if variance weighting is used)? This information is extracted to the third dimension of the output image.

review = yes
Review the extracted spectra? The interactive parameter must also be yes.

line = INDEF, nsum = 1
The dispersion line (line or column perpendicular to the

dispersion axis) and number of adjacent lines (half before and half after unless at the end of the image) used in finding, recentering, resizing, and editing operations. A line of INDEF selects the middle of the image along the dispersion axis. A positive nsum selects a sum of lines and a negative selects a median of lines.

## DEFAULT APERTURE PARAMETERS

lower = -5., upper = 5.
    Default lower and upper aperture limits relative to the aperture center. These limits are used for apertures found with apfind and when defining the first aperture in apedit.

apidtable = ""
    Aperture identification table. This may be either a text file or an image. A text file consisting of lines with an aperture number, beam number, and aperture title or identification. An image will contain the keywords SLFIBnnn with string value consisting of aperture number, beam number, optional right ascension and declination, and aperture title. This information is used to assign aperture information automatically in apfind and apedit.

## DEFAULT BACKGROUND PARAMETERS

b_function = "chebyshev"
    Default background fitting function. The fitting function types are "chebyshev" polynomial, "legendre" polynomial, "spline1" linear spline, and "spline3" cubic spline.

b_order = 1
    Default background function order. The order refers to the number of terms in the polynomial functions or the number of spline pieces in the spline functions.

b_sample = "-10:-6,6:10"
    Default background sample. The sample is given by a set of colon separated ranges each separated by either whitespace or commas. The string "*" refers to all points. Note that the background coordinates are relative to the aperture center and not image pixel coordinates so the endpoints need not be integer.

b_naverage = -3
    Default number of points to average or median.  Positive numbers average that number of sequential points to form a fitting point.  Negative numbers median that number, in absolute value, of sequential points.  A value of 1 does no averaging and each data point is used in the fit.

b_niterate = 0
    Default number of rejection iterations.  If greater than zero the fit is used to detect deviant fitting points and reject them before repeating the fit.  The number of iterations of this process is given by this parameter.

b_low_reject = 3., b_high_reject = 3.
    Default background lower and upper rejection sigmas.  If greater than zero points deviating from the fit below and above the fit by more than this number of times the sigma of the residuals are rejected before refitting.

b_grow = 0.
    Default reject growing radius.  Points within a distance given by this parameter of any rejected point are also rejected.

## APERTURE CENTERING PARAMETERS

width = 5.
    Width of spectrum profiles.  This parameter is used for the profile centering algorithm in this and other tasks.

radius = 10.
    The profile centering error radius for the centering algorithm.

threshold = 0.
    Centering threshold for the centering algorithm.  The range of pixel intensities near the initial centering position must exceed this threshold.

## AUTOMATIC FINDING AND ORDERING PARAMETERS

nfind = 1
    Maximum number of apertures to be defined.  This is a query parameter so the user is queried for a value except when given

explicitly on the command line.

minsep = 5.
    Minimum separation between spectra. Weaker spectra or noise within this distance of a stronger spectrum are rejected.

maxsep = 1000.
    Maximum separation between adjacent spectra. This parameter is used to identify missing spectra in uniformly spaced spectra produced by fiber spectrographs. If two adjacent spectra exceed this separation then it is assumed that a spectrum is missing and the aperture identification assignments will be adjusted accordingly.

order = "increasing"
    When assigning aperture identifications order the spectra "increasing" or "decreasing" with increasing pixel position (left-to-right or right-to-left in a cross-section plot of the image).

## RECENTERING PARAMETERS

aprecenter = ""
    List of apertures to be used in shift calculation.

npeaks = INDEF
    Select the specified number of apertures with the highest peak values to be recentered. If the number is INDEF all apertures will be selected. If the value is less than 1 then the value is interpreted as a fraction of total number of apertures.

shift = yes
    Use the average shift from recentering the apertures selected by the aprecenter parameter to apply to the apertures selected by the apertures parameter. The recentering is then a constant shift for all apertures.

## RESIZING PARAMETERS

llimit = INDEF, ulimit = INDEF
    Lower and upper aperture size limits. If the parameter ylevel is INDEF then these limits are assigned to all apertures. Otherwise these parameters are used as limits to the resizing

operation. A value of INDEF places the aperture limits at the image edge (for the dispersion line used).

ylevel = 0.1
    Data level at which to set aperture limits. If it is INDEF then the aperture limits are set at the values given by the parameters llimit and ulimit. If it is not INDEF then it is a fraction of the peak or an actual data level depending on the parameter peak. It may be relative to a local background or to zero depending on the parameter bkg.

peak = yes
    Is the data level specified by ylevel a fraction of the peak?

bkg = yes
    Subtract a simple background when interpreting the ylevel parameter. The background is a slope connecting the first inflection points away from the aperture center.

r_grow = 0.
    Change the lower and upper aperture limits by this fractional amount. The factor is multiplied by each limit and the result added to limit.

avglimits = no
    Apply the average lower and upper aperture limits to all apertures.

## TRACING PARAMETERS

t_nsum = 10
    Number of dispersion lines to be summed at each step along the dispersion.

t_step = 10
    Step along the dispersion axis between determination of the spectrum positions.

t_nlost = 3
    Number of consecutive steps in which the profile is lost before quitting the tracing in one direction. To force tracing to continue through regions of very low signal this parameter can be made large. Note, however, that noise may drag the trace

away before it recovers.

t_function = "legendre"
    Default trace fitting function. The fitting function types are "chebyshev" polynomial, "legendre" polynomial, "spline1" linear spline, and "spline3" cubic spline.

t_order = 2
    Default trace function order. The order refers to the number of terms in the polynomial functions or the number of spline pieces in the spline functions.

t_sample = "*"
    Default fitting sample. The sample is given by a set of colon separated ranges each separated by either whitespace or commas. The string "*" refers to all points.

t_naverage = 1
    Default number of points to average or median. Positive numbers average that number of sequential points to form a fitting point. Negative numbers median that number, in absolute value, of sequential points. A value of 1 does no averaging and each data point is used in the

t_niterate = 0
    Default number of rejection iterations. If greater than zero the fit is used to detect deviant traced positions and reject them before repeating the fit. The number of iterations of this process is given by this parameter.

t_low_reject = 3., t_high_reject = 3.
    Default lower and upper rejection sigma. If greater than zero traced points deviating from the fit below and above the fit by more than this number of times the sigma of the residuals are rejected before refitting.

t_grow = 0.
    Default reject growing radius. Traced points within a distance given by this parameter of any rejected point are also rejected.

EXTRACTION PARAMETERS

background = "none" (none|average|median|minimum|fit)

Type of background subtraction. The choices are "none" for no background subtraction, "average" to average the background within the background regions, "median" to use the median in the background regions, "minimum" to use the minimum in the background regions, or "fit" to fit across the dispersion using the background within the background regions. Note that the "average" option does not do any medianing or bad pixel checking, something which is recommended. The fitting option is slower than the other options and requires additional fitting parameter.

weights = "none" (none|variance)
Type of extraction weighting. Note that if the clean parameter is set then the weights used are "variance" regardless of the weights specified by this parameter. The choices are:

"none"
The pixels are summed without weights except for partial pixels at the ends.

"variance"
The extraction is weighted by the variance based on the data values and a poisson/ccd model using the gain and readnoise parameters.

pfit = "fit1d" (fit1d|fit2d)
Profile fitting algorithm to use with variance weighting or cleaning. When determining a profile the two dimensional spectrum is divided by an estimate of the one dimensional spectrum to form a normalized two dimensional spectrum profile. This profile is then smoothed by fitting one dimensional functions, "fit1d", along the lines or columns most closely corresponding to the dispersion axis or a special two dimensional function, "fit2d", described by Marsh (see approfile).

clean = no
Detect and replace deviant pixels?

skybox = 1
Box car smoothing length for sky background when using background subtraction. Since the background noise is often the limiting factor for good extraction one may box car smooth

the sky to improve the statistics in smooth background regions at the expense of distorting the subtraction near spectral features. This is most appropriate when the sky regions are limited due to a small slit length.

saturation = INDEF
    Saturation or nonlinearity level in data units. During variance weighted extractions wavelength points having any pixels above this value are excluded from the profile determination and the sigma spectrum extraction output, if selected by the extras parameter, flags wavelengths with saturated pixels with a negative sigma.

readnoise = 0.
    Read out noise in photons. This parameter defines the minimum noise sigma. It is defined in terms of photons (or electrons) and scales to the data values through the gain parameter. A image header keyword (case insensitive) may be specified to get the value from the image.

gain = 1
    Detector gain or conversion factor between photons/electrons and data values. It is specified as the number of photons per data value. A image header keyword (case insensitive) may be specified to get the value from the image.

lsigma = 3., usigma = 3.
    Lower and upper rejection thresholds, given as a number of times the estimated sigma of a pixel, for cleaning.

nsubaps = 1
    During extraction it is possible to equally divide the apertures into this number of subapertures. For multispec format all subapertures will be in the same file with aperture numbers of 1000*(subap-1)+ap where subap is the subaperture (1 to nsubaps) and ap is the main aperture number. For echelle format there will be a separate echelle format image containing the same subaperture from each order. The name will have the subaperture number appended. For onedspec format each subaperture will be in a separate file with extensions and aperture numbers as in the multispec format.

ADDITIONAL PARAMETERS

Dispersion axis and I/O parameters are taken from the package parameters.

DESCRIPTION

This task provides functions for defining, modifying, tracing, and extracting apertures from two dimensional spectra. The functions desired are selected using switch parameters. When the task is run interactively queries are made at each step allowing additional control of the operations performed on each input image.

The functions, in the order in which they are generally performed, are summarized below.

o Automatically find a specified number of spectra and assign default apertures. Apertures may also be inherited from another image or defined using an interactive graphical interface called the aperture editor.

o Recenter selected reference apertures on the image spectrum profiles.

o Resize the selected reference apertures based on spectrum profile width.

o Interactively define or adjust aperture definitions using a graphical interface called the aperture editor. All function may also be performed from this editor and, so, provides an alternative method of processing and extracting spectra.

o Trace the positions of the selected spectra profiles from a starting image line or column to other image lines or columns and fit a smooth function. The trace function is used to shift the center of the apertures at each dispersion point in the image.

o Extract the flux in the selected apertures into one dimensional spectra in various formats. This includes possible background subtraction, variance weighting, and bad pixel rejection.

Each of these functions has different options and parameters. In addition to selecting any of these functions in this task, they may also be selected using the aperture editor and as individual commands (which themselves allow selection of other functions). When broken down into individual tasks the parameters are also sorted by their function though there are then some mutual parameter interdependencies. This functional decomposition is what was available prior to the addition of the apall task. It is recommended that this task be used because it collects all the parameters in one place eliminating confusion over where a particular parameter is defined. However, documenting the various functions is better organized in terms of the separate descriptions given for each of the functions; namely under the help topics apdefault, apfind, aprecenter, apresize, apedit, aptrace, and apsum.

## EXAMPLES

1. This example may be executed if desired. First we create an artificial spectrum with four spectra and a background. After it is created you can display or plot it. Next we define the dispersion axis and set the verbose flag to better illustrate what is happening. The task APALL is run with the default parameters except for background fitting and subtracting added. The text beginning with # are comments of things to try and do.

```
ap> artdata
ar> unlearn artdata
ar> mk1dspec apdemo1d nl=50
ar> mk2dspec apdemo2d model=STDIN
apdemo1d 1. gauss 3 0 20 .01
apdemo1d .8 gauss 3 0 40 .01
apdemo1d .6 gauss 3 0 60 .01
apdemo1d .4 gauss 3 0 80 .01
[EOF=Control D or Control Z]
ar> mknoise apdemo2d background=100. rdnoise=3. poisson+
ar> bye
# Display or plot the spectrum
ap> dispaxis=2; verbose=yes
ap> unlearn apall
ap> apall apdemo2d back=fit
Searching aperture database ...
Find apertures for apdemo2d?  (yes):
Finding apertures ...
Number of apertures to be found automatically (1): 4
Jul 31 16:55: FIND - 4 apertures found for apdemo2d.
Resize apertures for apdemo2d?  (yes):
Resizing apertures ...
Jul 31 16:55: RESIZE - 4 apertures resized for apdemo2d.
Edit apertures for apdemo2d?  (yes):
# Get a list of commands with '?'
# See all the parameters settings with :par
# Try deleting and marking a spectrum with 'd' and 'm'
# Look at the background fitting parameters with 'b' (exit with 'q')
# Exit with 'q'
Trace apertures for apdemo2d?  (yes):
Fit traced positions for apdemo2d interactively?  (yes):
Tracing apertures ...
Fit curve to aperture 1 of apdemo2d interactively  (yes):
# You can use ICFIT commands to adjust the fit.
Fit curve to aperture 2 of apdemo2d interactively  (yes): n
Fit curve to aperture 3 of apdemo2d interactively  (no):
Fit curve to aperture 4 of apdemo2d interactively  (no): y
Jul 31 16:56: TRACE - 4 apertures traced in apdemo2d.
Write apertures for apdemo2d to apdemosdb  (yes):
Jul 31 16:56: DATABASE - 4 apertures for apdemo2d written to database.
Extract aperture spectra for apdemo2d?  (yes):
Review extracted spectra from apdemo2d?  (yes):
Extracting apertures ...
Review extracted spectrum for aperture 1 from apdemo2d?  (yes):
# Type 'q' to quit
Jul 31 16:56: EXTRACT - Aperture 1 from apdemo2d --> apdemo2d.ms
Review extracted spectrum for aperture 2 from apdemo2d?  (yes): N
Jul 31 16:56: EXTRACT - Aperture 2 from apdemo2d --> apdemo2d.ms
Jul 31 16:56: EXTRACT - Aperture 3 from apdemo2d --> apdemo2d.ms
Jul 31 16:57: EXTRACT - Aperture 4 from apdemo2d --> apdemo2d.ms
```

2. To extract a series of similar spectra noninteractively using a reference for the aperture definitions, then recentering and resizing but not retracing:

```
  ap> apall fib*.imh ref=flat inter- trace-
```

Note that the interactive flag automatically turns off the edit, fittrace, and review options and the reference image eliminates the find (find only occurs if there are no initial apertures).

REVISIONS

APALL V2.11

The "apertures" parameter can be used to select apertures for resizing, recentering, tracing, and extraction. This parameter name was previously used for selecting apertures in the recentering algorithm. The new parameter name for this is now "aprecenter".

The aperture ID table information may now be contained in the image header under the keywords SLFIBnnn.

The "nsubaps" parameter now allows onedspec and echelle output formats. The echelle format is appropriate for treating each subaperture as a full echelle extraction.

APALL V2.10.3

The dispersion axis parameter was moved to purely a package parameter.

As a final step when computing a weighted/cleaned spectrum the total fluxes from the weighted spectrum and the simple unweighted spectrum (excluding any deviant and saturated pixels) are computed and a "bias" factor of the ratio of the two fluxes is multiplied into the weighted spectrum and the sigma estimate. This makes the total fluxes the same. In this version the bias factor is recorded in the logfile if one is kept. Also a check is made for unusual bias factors. If the two fluxes disagree by more than a factor of two a warning is given on the standard output and the logfile with the individual total fluxes as well as the bias factor. If the bias factor is negative a warning is also given and no bias factor is applied. In the previous version a negative (inverted) spectrum would result.

SEE ALSO

apdefault, apfind, aprecenter, apresize, apedit, aptrace, apsum

# Appendix C

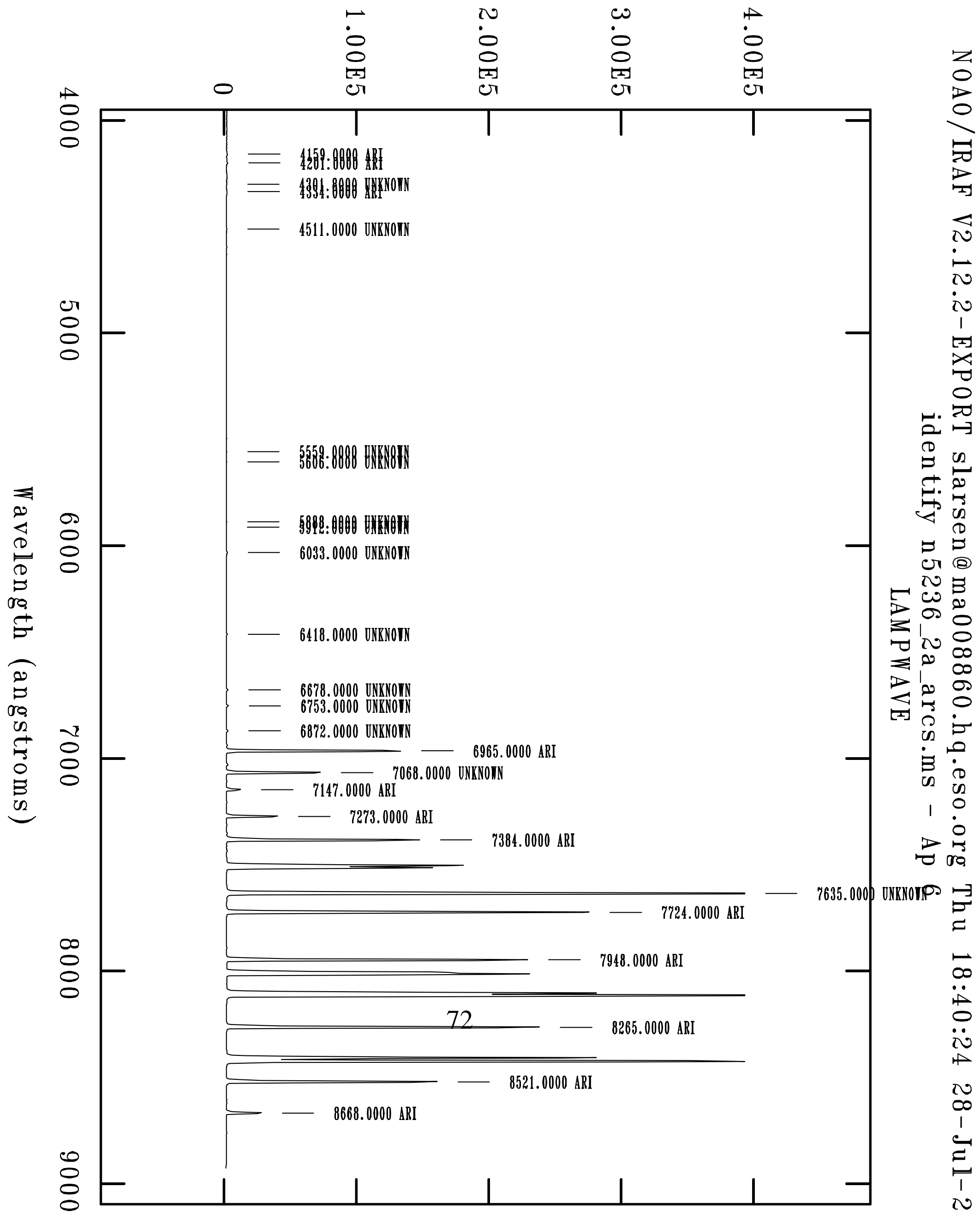
# EMMI line list

| Wavelength (Å) |
|---|
| 4159 ARI |
| 4201 ARI |
| 4301.8 UNKNOWN |
| 4334 ARI |
| 4511 UNKNOWN |
| 5559 UNKNOWN |
| 5606 UNKNOWN |
| 5888 UNKNOWN |
| 5912 UNKNOWN |
| 6033 UNKNOWN |
| 6418 UNKNOWN |
| 6678 UNKNOWN |
| 6753 UNKNOWN |
| 6872 UNKNOWN |
| 6965 ARI |
| 7068 UNKNOWN |
| 7147 ARI |
| 7273 ARI |
| 7384 ARI |
| 7635 UNKNOWN |
| 7724 ARI |
| 7948 ARI |
| 8265 ARI |
| 8521 ARI |
| 8668 ARI |
| 9124 UNKNOWN |

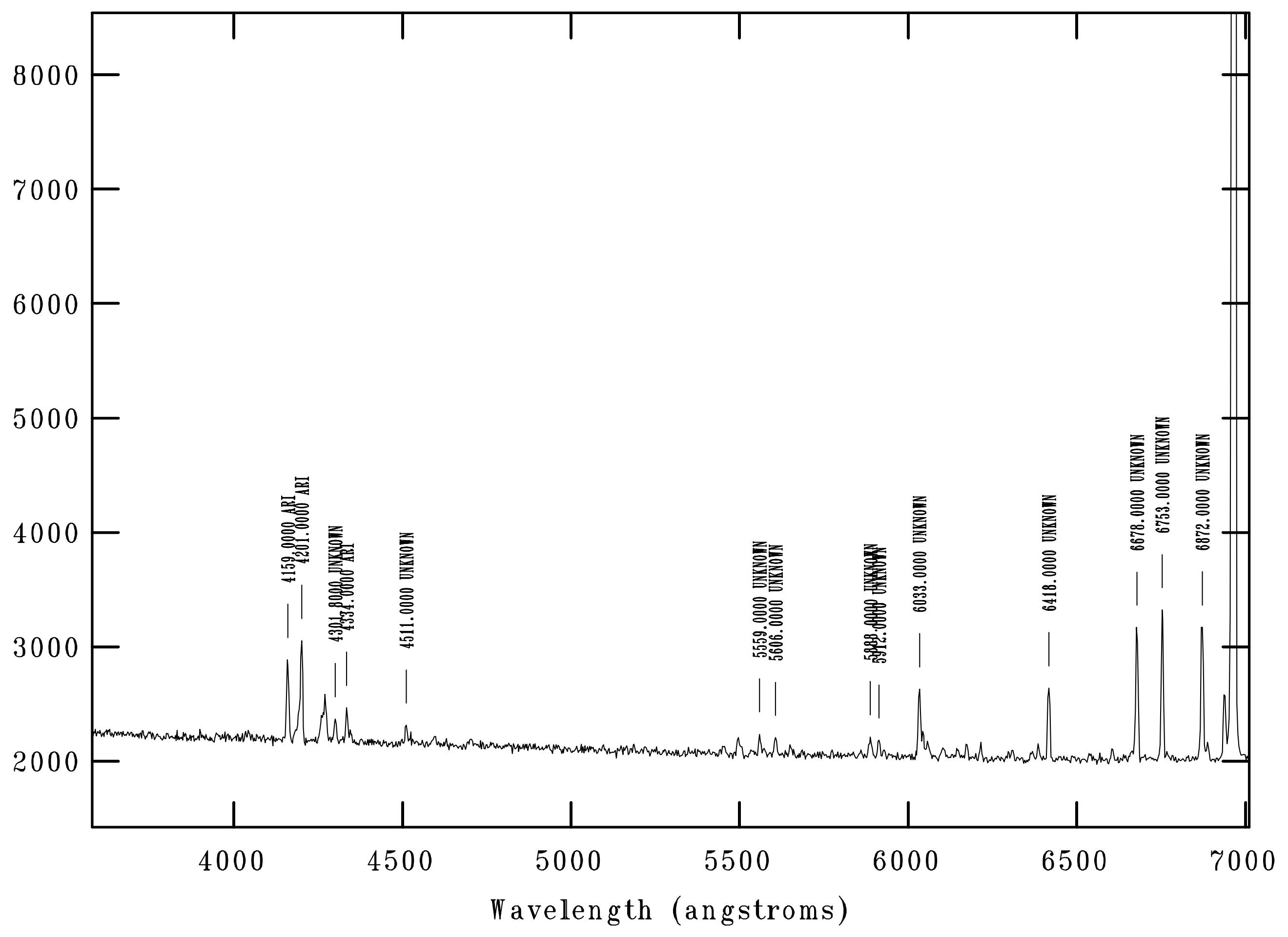
Table C.1: Line list for EMMI Grism #3.

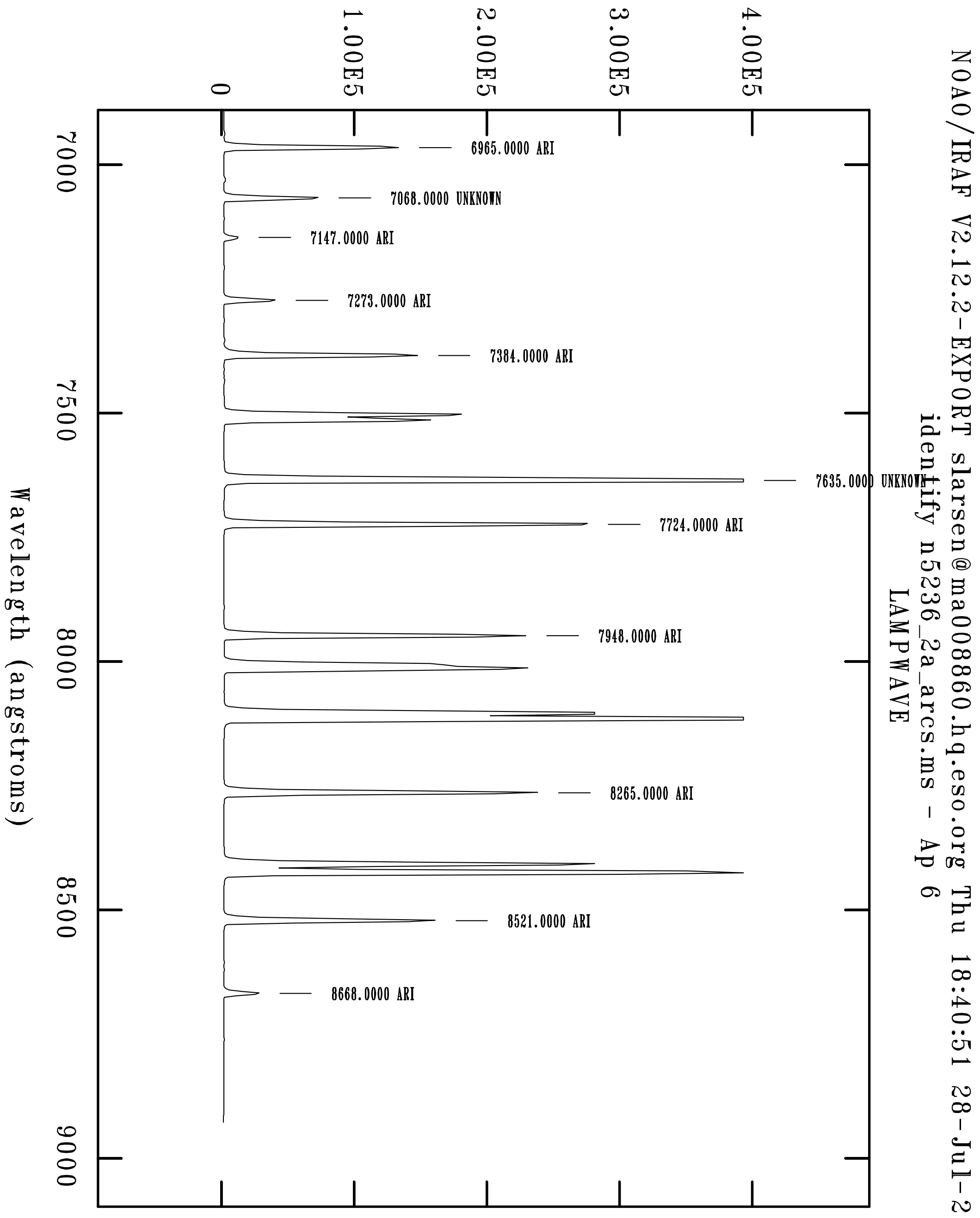
# Appendix D

# EMMI calibration lamp spectra



NOAO/IRAF V2.12.2-EXPORT slarsen@ma008860.hq.eso.org Thu 18:41:41 28-Jul-2
identify n5236_2a_arcs.ms - Ap 6
LAMPWAVE




NOAO/IRAF V2.12.2-EXPORT slarsen@ma008860.hq.eso.org Thu 18:40:51 28-Jul-2
identify n5236_2a_arcs.ms - Ap 6
LAMPWAVE
0
1.00E5
2.00E5
3.00E5
4.00E5
7000
7500
8000
8500
9000
Wavelength (angstroms)
6965.0000 ARI
7068.0000 UNKNOWN
7147.0000 ARI
7273.0000 ARI
7384.0000 ARI
7635.0000 UNKNOWN
7724.0000 ARI
7948.0000 ARI
8265.0000 ARI
8521.0000 ARI
8668.0000 ARI