Part II

DATA REDUCTION TOOLS AND APPLICATIONS

ANALYZING ONE-DIMENSIONAL SPECTRA

This chapter covers the use of several tasks for manipulating one-dimensional EUVE spectrum images. In addition to the IRAF/EUV tasks described here, there are many spectral analysis tasks in **noao.onedspec** which may be used on EUVE spectra as well. See Appendix D for a listing of such tasks.

Note: The nominally reduced spectra from the standard Guest Observer data set are not high quality, and are intended as a starting point only. The EGO Center strongly advises *all* GO's to re-extract their EUVE spectra before performing further analyses. Some notes on using the **apextract** package to extract spectra from EUVE spectral images can be found in chapter 5, § 5.2.

Many of the tasks covered in this chapter require access to a set of instrument reference data, which is on disk at the EGO Center, or has been provided on your EUVE data tape. The reference data can also be obtained from the CEA anonymous ftp site, in the EGODATA directory. Before proceeding with spectrum analysis or simulations, make sure the directory *reference* is present in the configuration shown in figure 0-5 and described in section 0.4.1. All FITS files should have been converted to IRAF images or ST tables.

The EUVE spectra are delivered in units of raw counts.

Many GO's will want to compare their spectra to models. To do this, the 1-d spectra must be joined and flux calibrated, or else the effect of the spectrometer must be imprinted on the model spectrum. Most EUVE spectra have relatively low signal to noise, even after the spectral extraction has been optimized, so that comparisons are usually more trustworthy if the model spectrum is converted to simulated EUVE channel spectra in counts for a comparable exposure time. Therefore, the recommended method is to use the EUVE simulation task **specmod** to transform model spectra into simulated EUVE channel spectra with the same binning as the data, before comparing the results. This method is illustrated In figure 4-1.

In cases where the EUVE extracted spectra have relatively high signal to noise, and the effects of higher ($n = -2, -3$,...) orders are known to be negligible, the IRAF/EUV task **euvcombine** can be used to join the three channel spectra, and divide by the exposure and first order effective area. This produces a reasonable first-order flux-calibrated spectrum which can be compared directly to a model image. However, since many spectra have substantial contributions from higher orders, and **euvcombine** does NOT do a reliable job of removing higher orders when the signal-to-noise is low, this method does not work well on the spectra obtained in many Cycle I (1993) observations.

Section 4.1 begins with guidelines for creating simulated one-dimensional EUVE spectra from the ASCII output of a modeling program supplied by the user. We first introduce a task that

will rebin an ASCII model spectrum into a 1-d IRAF image by doing a linear interpolation. While it is not necessary to have IRAF spectra to create EUVE simulations, it will be desirable in some cases to rebin model spectra before using the simulation task. The utility **listspec** is useful for inspecting the rebinned model, or for re-converting the pixel values of 1-d images to ASCII form. Section 4.1.4 gives instructions for running the EUVE simulation task **specmod**, and section 4.1.5 gives a detailed example of creating a simulation. Section 4.2 treats the tasks in the IRAF/EUV packages which compare and combine spectra. **Compspec** is a general utility that performs a χ^2 test, and **euvcombine** combines a set of three EUVE channel spectra into a single image.

4.1 Making Simulated EUVE Spectra from Models

The EUV packages include facilities for producing simulated 1-dimensional EUVE spectra from model spectra of the source flux at the telescope. "Flux at the telescope" means photons/cm^2/sec, and that **model spectra must include the effects of interstellar absorption on the source spectrum.** GO's may be interested in the ISM simulation programs available through anonymous ftp from the Center for EUV Astrophysics[1] The task **euvtools.mkeuvspec** is useful for creating input spectra compatible with simulation and comparison software. The task **euvspec.specmod** creates simulated EUVE spectral images with a user-specified bin size, given a model spectrum in the form of a suitable ASCII line list or 1-d IRAF image and the integration time.

4.1.1 MKEUVSPEC creates images from model spectra

If the source model to be used is in the form of an ASCII line list in units of photons/cm^2/sec, the spectrum may be used without modification to create simulated EUVE spectra. The user may wish to skip over this section and go on to section 4.1.4, on the use of the simulation task **specmod**.

However, if the source model represents a continuum with units of photons/cm^2/sec/Å, or an emission source with broad or blended features, it must first be rebinned into a 1-d spectral image using the task **euvtools.mkeuvspec** to supply data in the correct units. **Mkeuvspec** treats an input list of fluxes as a piecewise continuous spectrum. It integrates and rebins the flux into bins with a size specified by the user.

Continuum input is simulated by a sufficiently dense array of lines, where sufficiently dense means that the line spacing is narrow compared to the spectrometer PSF that will be applied to the output by **specmod** [2]. To convert a model continuum spectrum called *cont.model*, edit the parameter file for **mkeuvspec** as shown in figure 4-2.

[1] See the directory *pub/archive/ism* at *cea-ftp.cea.berkeley.edu.*

[2] The spectrometer resolution element is eqivalent to the half width of the local PSF. Resolution elements in the short, medium and long wavelength channels are approximately .5 Å, 1.0 Å, and 2.0 Å, respectively.

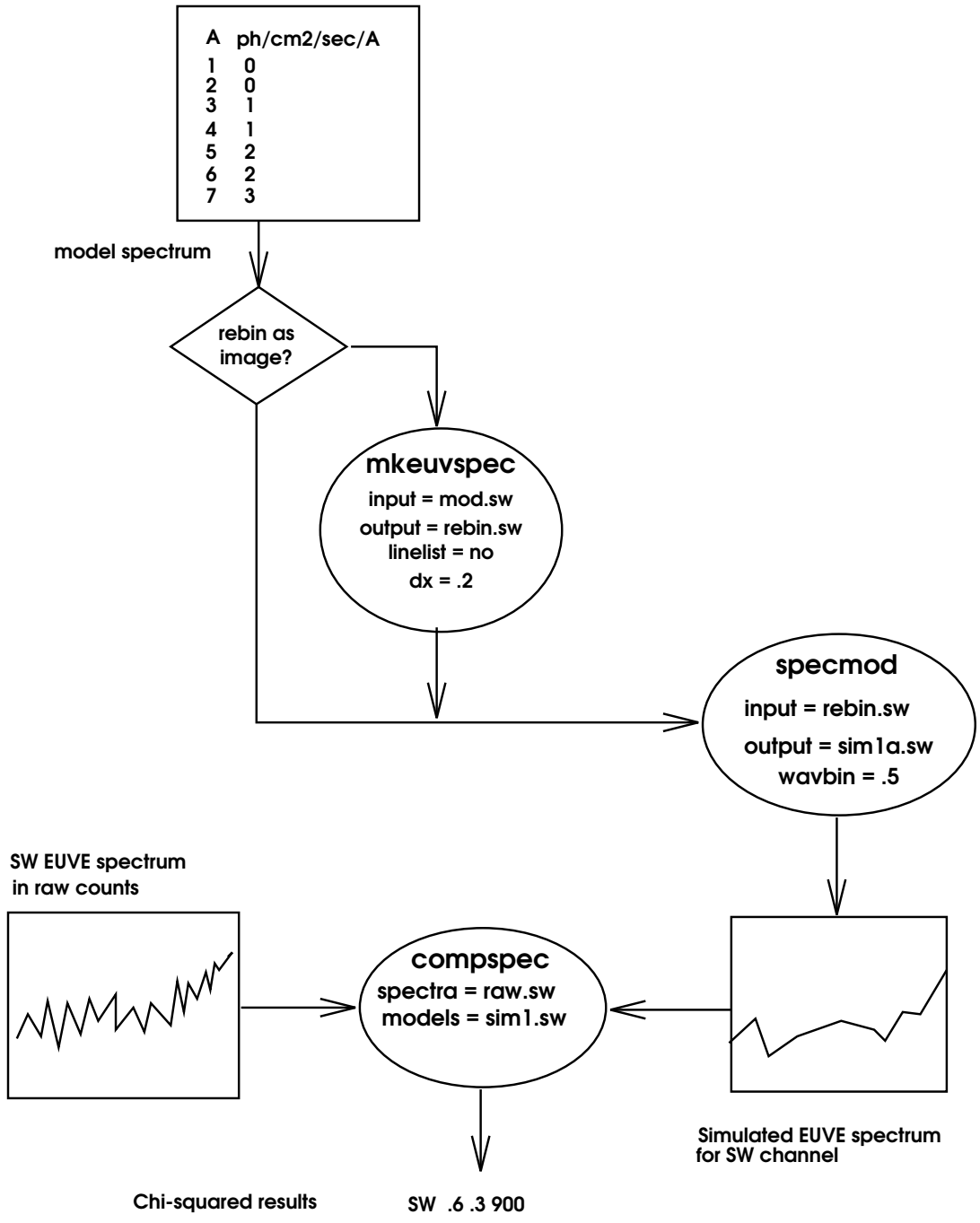

Figure 4-1: Comparing models to a typical EUVE spectrum.

```
eu> lpar mkeuvspec

     inputs = "cont.model"    Input file list
    outputs = "cmod_in"       Output file list
      (xcol = 1)              Wavelength column
      (ycol = 2)              Y value column
        (x1 = INDEF)          First bin's wavelength
        (x2 = INDEF)          Last bin's wavelength
        (dx = 0.05)           Wavelengths per pixel
  (linelist = no)             Line list or plf
 (integrate = yes)            Is the output to be multiplied by wpc
      (mode = "al")
```

Figure 4-2: Sample **mkeuvspec** parameters to make an IRAF image from an ASCII spectrum.

Using **mkeuvspec**

- Letting `x1` and `x2` equal "INDEF" causes **mkeuvspec** to determine the wavelength range from the first and last values in the file.
- Setting `linelist` to "no" tells it to treat the input as a continuum and do a linear interpolation between data points.
- Setting `integrate` to "yes" causes the function to be integrated over each bin, changing the units of photons/cm^2/sec/Å to photons/cm^2/sec.
- To produce input to **specmod**, the value of the bin size in `dx` should be no more than one fourth of one resolution element in the simulated EUVE spectrum (see foonote nearby). Values comparable to the resolution size may cause the bin size of the spectral image to "beat" against the PSF used in **specmod**, producing spurious features.

Creating simulated EUVE spectra of a continuum model in photons/cm^2/sec/Å from the command line requires two steps:

```
eu> mkeuvspec contin_1e18 contin18 dx=.05 linelist=no integrate=yes

eu> specmod cont18.imh contin18_sim time=50000 wavbin=.2
```

The use of **specmod** is discussed in section 4.1.4.

Making emission line lists into IRAF spectra for use as input to **specmod** is not recommended, as it will either produce inaccurate results (input bins larger) or slow down processing by **specmod** (input bins smaller). However, if an emission spectrum has lines or blends which already span several bins, the spectrum should be treated as a continuum by setting `linelist` to "no", and `integrate` to "yes". If emission spectra in photons/cm^2/sec are to be rebinned for some other purpose, set the parameter `linelist` to "yes" and `integrate` to "no". Then each line will be recorded in only one bin in the output spectrum.

4.1.2 ISM calculates transmission of the interstellar medium

GO's who wish to compare their data with simulated EUVE spectra made from a theoretical model need to supply the modeled target spectrum *at the telescope*. The task **ism** is a tool for calculating the effect of absorption by the interstellar medium on spectra produced by modeling programs. This transforms the model spectrum into the desired spectrum at the telescope.

The task currently supports a single ISM model, presented in the paper by Rumph, et. al.[3]. The required inputs are the column density of neutral hydrogen in the direction of the target, in cm^{-2}, the ratios of HeI/HI, and HeII/HI, and a two-column list with wavelengths in Å in the first column and fluxes in photons/cm^2/sec/Å in the second, for which to compute the local flux. Each flux value is multiplied by a dimensionless transmission for that wavelength. A typical parameter file for **ism** is shown in figure 4-3

The output from this task is suitable for input to **specmod**, after processing with **mkeuvspec** to produce a spectral image in units of photons/cm^2/sec. Figure 4-4 a) and b) show plots of the white dwarf model before and after processing with **ism**.

```
        eu> lpar ism


   hcolumn = 1.0000000000000E18 neutral hydrogen column density (cm**-2)
(he1hratio = 0.1)            HeI/HI ratio
(he2hratio = 0.)             HeII/HI ratio
    (model = "rumph")        ism model
    (input = "WD7.sim")      Name of input file (or stdin)
   (output = "WD7.local")    Name of output file (or stdout)
 (document = no)             output comment lines that document parameters
     (mode = "al")
```

Figure 4-3: Parameters for calculating flux at the telescope from a white dwarf model

[3]Interstellar Medium Contiuum, Autoionization, and Line Absorption in the Extreme Ultraviolet, T. Rumph, S. Bowyer, and S. Vennes, Submitted to *Ast. J.*

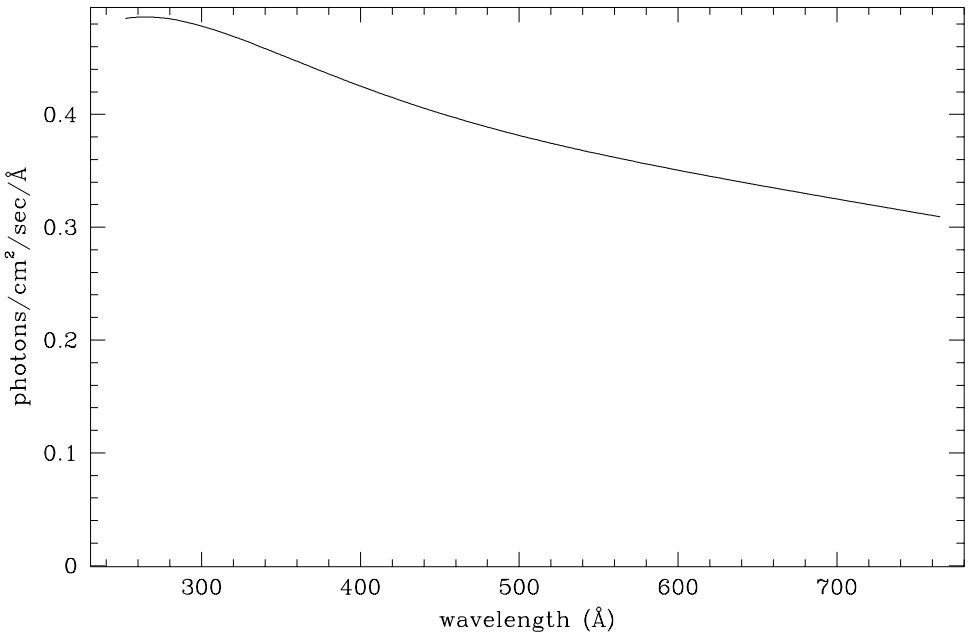

Figure 4-4: a) White dwarf spectral model in photons/cm^2/sec/Å.

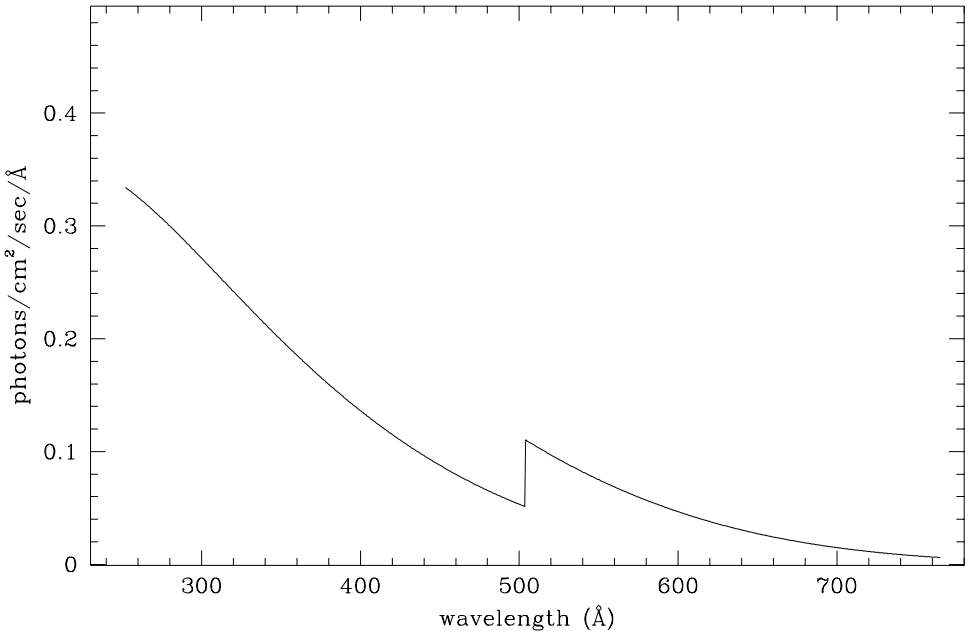

Figure 4-4: b) White dwarf model after processing with **ism**

4.1.3 LISTSPEC lists spectra in WCS units

The task **euvtools.listspec** is analagous to **images.listpixels**. It takes a 1-d spectrum as input and creates a listing of wavelength vs. the pixel values, counts or flux, according to the input. **Listspec** can determine the wavelengths from the WCS in the image header, or use a different bin size supplied by the user in the parameter `dx` and calculate the appropriate fluxes by linear interpolation. The user also has the option of setting the parameter `average` to divide each value by `CD1_1`, the bin width in Å. This produces an average over the bin represented by each pixel, and converts a spectrum image in photons/cm^2/sec to an ASCII listing in photons/cm^2/sec/Å. **Listspec** can be used to list the output from **mkeuvspec** for comparison to the input model, or applied to flux-calibrated 1-d EUVE spectra of continuum sources to convert them to the same units as model spectra, as in the following example:

```
eu> listspec g191_all x1=70 average=yes > g191_list
```

4.1.4 SPECMOD makes simulated EUVE spectra

The task **specmod** simulates the effect of the EUVE Spectrometer on model spectra. The user defines the integration time, the binning of the output images, and can choose whether to convolve the spectra with a point spread function. Input model spectra can be either ASCII of wavelengths and emission line fluxes, or IRAF images containing integrated flux in photons/cm^2/sec, with a world coordinate system in Å per pixel. Fluxes in ASCII input files need not be regularly spaced; the model may contain any list of the centers and integrated fluxes of an arbitrary collection of narrow emission lines. Note that model spectra to be used as input to **specmod** must have units of integrated flux (photons/cm^2/sec). Continuum spectra in photons/cm^2/sec/Å should first be processed with **mkeuvspec**, as described in section 4.1.1.

Also, model programs do not always produce spectra with small enough bins for input to **specmod**. This can cause problems, since the current version of **specmod** treats each input line flux as a delta function at the midpoint of a bin, assuming nothing about the distribution of flux within a bin. This is a necessity on some level with any simulation program which must make discrete calculations. But if the bin size of the input spectrum is not several times smaller than the resolution specified by the user for the output, there may be dips or spurious features in the spectrum even after it is convolved with the PSF.

A spectrum with infinite resolution can be produced by setting the parameter `psf` to "no", which turns off convolution of the spectrum with the instrument point spread function. The resulting "delta-function" spectrum can be useful for locating line centers exactly.

Assuming these considerations have been satisfied, a typical parameter setup is pictured in figure 4-5, and some guidelines for using **specmod** are elaborated below.

Using **specmod**

- The input spectrum must consist of pairs of wavelengths in Å and fluxes in photons/cm^2/sec. These may be in a two-column ASCII file with one wavelength/flux per line, or in a 1-d IRAF image. Input in IRAF images must have a bin size about 4x smaller than the

resolution of the output spectrum. If the source model is in photons/cm^2/sec/Å and represents a continuum, it should first be integrated and rebinned into an IRAF image using **mkeuvspec** as in section 4.1.1.

- When the spectrum input to **specmod** is an image, the ".imh" suffix must be supplied.
- **Specmod** uses the instrument reference data: wavelength solution, effective areas, and PSF. The default value of `refdata`, *reference/detector.tab*, should be correct for the standard EUVE directory setup shown in figure 0-5. If the reference data are stored elsewhere, *reference* should be defined as an IRAF system variable, and a "$" should be substituted for the "/".
- The option of simulating selected channels has not yet been completely implemented. If only one channel is specified in `detectors`, only that channel's spectrum is produced. Single channels should be produced if the user wants different bin sizes for the different spectra[4] Specifying any combination of channels in `detectors` will cause **specmod** to produce all three channel spectra with the same bin size. This feature will be refined in a future version of the software.

Users should also be aware of the tasks in the package **noao.artdata** which can be used to construct artificial spectra or add noise to spectra. These may be useful in trying to simulate EUVE spectra including the background or Poisson noise. Keep in mind that the actual extracted spectra have noise properties which depend on the number of lines in the aperture over which each spectrum was summed and the algorithm used to estimate and subtract the background.

4.1.5 Examples: Comparing model simulations with EUVE spectra

4.1.5.1 Simulating EUVE spectra for an emission source

Suppose a hypothetical GO named Dr. Igor EGO wishes to make simulated SW, MW, and LW EUVE spectra from an ASCII file containing spectral line wavelengths and fluxes for an emission source from a model that assumes, among other things, a certain hydrogen and helium column along the line of sight. The first few lines of the model file might look like this:

[4] Recall from section 4.1.1 that the resolution element sizes vary by a factor of two from channel to channel.

```
eu> lpar specmod

      input = "latestar.ASCII" Name of input linelist file
     output = "latestar.sim"  Root name of spectral output file
      (time = 40000.)         Integration time (seconds)
    (wavbin = 0.2)            Wavelength bin (Angstroms)
       (psf = yes)            Convolve with point spread function?
   (refdata = "reference$detector.tab") Name of ST table for reference data
 (detectors = "sw,mw,lw")     Input detectors for spectral images
      (mode = "al")
```

Figure 4-5: **Specmod** parameters for making simulated EUVE spectra from an ASCII line list. Broadened or blended lines should have fluxes specified at intervals of .05 Å or less.

```
72.799999 0.008758
74.200001 0.004903
74.799999 0.009914
75.299999 0.009670
77.860001 0.006975
79.009998 0.006224
81.099998 0.015461
82.750000 0.004913
82.889999 0.007008
83.119999 0.003802
83.340000 0.009537
83.799999 0.008645
```

The command:

```
eu> specmod emission_mod line time=50000 wavbin=.2
```

creates the spectral images *line.sw.imh*, *line.mw.imh*, and *line.lw.imh*. A plot of *line.sw* is shown in figure 4-6.

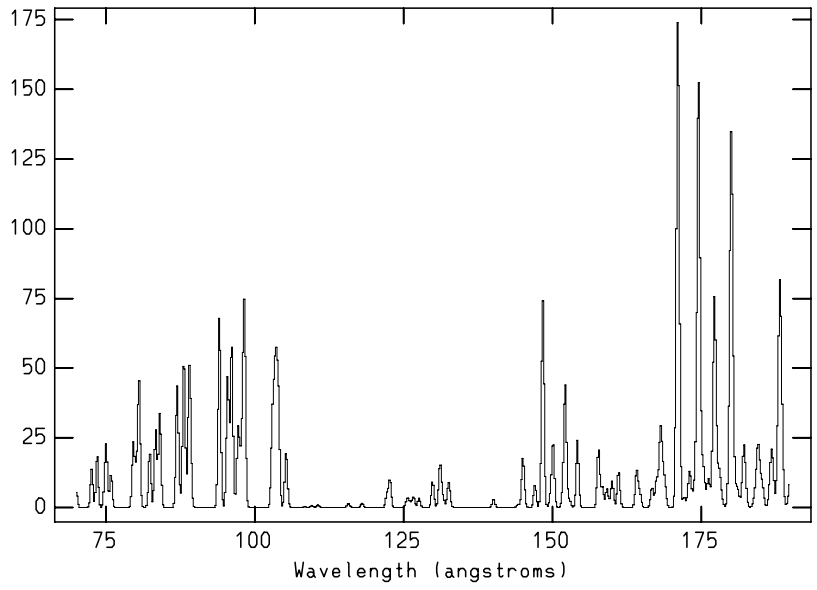

Figure 4-6: Model spectrum of a line source in the SW spectrometer. The exposure time is 10000 seconds and the bin size is 0.2 Å. The y-axis shows counts per bin.

The medium and long wavelength spectra are binned at a greater resolution than that of the respective spectrometer channels, but this will be compensated by the comparison task **compspec**, described below.

4.1.5.2 Simulated EUVE spectra for continuum sources

Now suppose that Dr. Ego wishes to use **specmod** with a model spectrum for a continuum source to evaluate his model against his EUVE spectrum. He starts with an ASCII file named *cont.model* containing the flux in units of photons/cm^2/sec/Å, tabulated at various wavelengths. The first few lines of that file are shown below:

```
135.0 0.000000
136.5 0.000000
138.0 0.000961
139.5 0.000974
141.0 0.001009
142.5 0.001050
144.0 0.001128
145.5 0.001342
147.0 0.001398
148.5 0.001429
150.0 0.001517
151.5 0.001567
```

Dr. Ego needs to make a line list from this piecewise-linear function before running **specmod**. This is done with **mkeuvspec**; The parameters for that task are assumed to be arranged as shown in figure 4-2. He executes:

```
eu> mkeuvspec cont.model cont
```

Then this 1-d spectrum is put through **specmod** with:

```
eu> specmod cont.imh cont time=53729
```

(remember to use ".imh" if input is an image), yielding three output spectra, *cont.sw.imh*, *cont.mw.imh*, and *cont.lw.imh*. The medium wavelength output spectrum is plotted in figure 4-7.

The model spectra constructed in this way include flux from all orders for which effective areas are available. If the proper exposure time is used, and the bin size has been appropriately set, they should be directly comparable to extracted EUVE spectra.

4.2 Manipulating and Comparing Spectra

The section introduces two tasks for doing a statistical comparison between pairs of real and model spectra, and for performing a first-order flux calibration when this is appropriate.

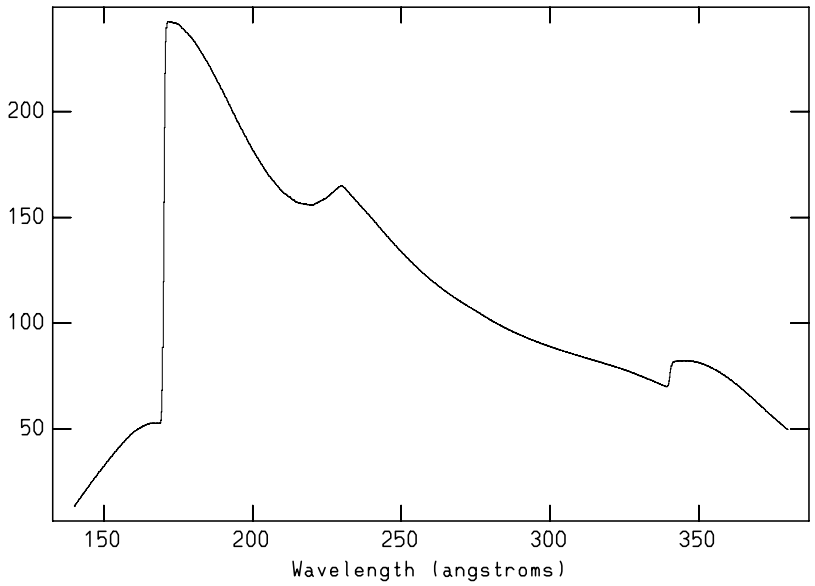

Figure 4-7: Model spectrum of a continuum source in the MW spectrometer. The exposure time is 30000 seconds and the bin size is 0.2 Å. The y-axis shows counts per bin.

4.2.1 COMPSPEC does statistical comparisons

To compare two spectra in either counts or flux units, the task **euvtools.compspec** performs a pixel-by-pixel computation of the χ^2 statistic and the associated probability that the two spectra sample the same parent distribution. Multiple models can also be compared to a single spectrum. A sample parameter file for comparing a single combined observation spectrum to a list of model spectra is shown in figure 4-8.

```
          eu> lpar compspec


    spectra1 = "mw_G191B"      Input one set of spectral 1d images
    spectra2 = "@all_models"   Input another set of spectral 1d images
      output = "g191_chi"      Table of results
(resolution = INDEF)           Resolution
        (x1 = INDEF)           Minimum wavelength
        (x2 = INDEF)           Maximum wavelength
      (mode = "al")
```

Figure 4-8: **compspec** parameters for comparing EUVE spectrum to a model.

Using **compspec**

- Images to be compared must be one-dimensional, and must be of the same size and wavelength range, and therefore have the same resolution.
- In most cases, un-fluxed EUVE spectra will be compared to simulated raw count spectra made with the task **specmod** from a model spectrum of the target spectrum.
- A number of model spectra can be compared to a single observation by using a comma-separated list or the *cl* pointer notation "*@filename*" for the value of `spectra2`.
- To compare only part of a spectrum, the beginning and ending wavelengths and the resolution may be specified using `w0` and `wn` to limit the bandpass or rebin the spectra before comparison.
- If comparing a single extracted EUVE spectrum with a number of models, use the parameter `output` to specify the root name for the ST table of results.

4.2.2 EUVCOMBINE joins channel spectra

GOs whose spectra have very good signal-to-noise can use the task **euvcombine** to join their background-subtracted EUVE spectra before using spectral comparison tools. This task uses the instrument effective areas for the three channels to adjust the levels of overlapping spectral regions and combine the three channels, with options for fluxing the spectrum in first order and specifying the wavelength range of the output spectrum.

Limitations of euvcombine

Although the functionality to remove counts from orders other than first order has been implemented in **euvcombine**, this algorithm does not work reliably because of problems with noise and behavior near sharp edges.

Therefore, in the current version (**1.4**) of the IRAF/EUV software, **euvcombine** should only be used when the effects from higher orders can be safely ignored, as in a continuum with very little short and medium wavelength flux, or easily separated, as with an emission source that has no lines that overlap with higher order features. In these cases, the EUVE spectra can be joined using **euvcombine**, but dividing by only the first order effective area. This produces a flux-calibrated first order spectrum that can be compared directly to model spectra. GO's should look at the ST tables *sw_ea.tab, mw_ea.tab,* and *lw_ea.tab* to see the relative efficiencies for higher orders.

An example parameter file setup for using **euvcombine** is shown in figure 4-9.

```
          eu> lpar euvcombine


     spectra = "g191sim.sw,g191sim.mw,g191sim.lw" Input spectra raw count 1d i
   detectors = "sw,mw,lw"      Input detectors for images
      output = "g191sim_com3"  Output spectra 1d image
    (refdata = "reference/detector.tab") Detector table
   (exposure = 40000.)         Exposure time in seconds
  (maxorders = 1)              Maximum number of orders
     (counts = no)             Output counts
   (autosize = no)             Determine output range from inputs
   (resample = 2)              Sampling factor to conserve memory
(minwavelengt = 70.)           Minimum output wavelength
(maxwavelengt = 760.)          Maximum output wavelength
       (mode = "al")
```

Figure 4-9: **euvcombine** parameters for combining three spectra.

Using **euvcombine**

- Both `spectra` and `detectors` should be specified in order of increasing wavelength, and combinations of adjacent channels are allowed, as in (sw,mw,lw), (sw,mw), or (mw,lw).
- When `counts` = "no", the parameter `exposure` is used to compute photon fluxes at the telescope. In this case, the raw counts in each bin of the output spectrum are divided by `exposure` and the first order effective area to compute the intensity in photons/cm^2/sec for each pixel.
- `maxorders` is intended to determines the maximum spectral order to be included in the combined spectrum. For the present, the value of `maxorders` should always be set to 1, so that all counts are assumed to be in first order and none are subtracted (see the box above). If it is an integer greater than one, **euvcombine** attempts to remove counts from

higher orders using effective areas for orders 2–`maxorder`, but it may misjudge the first order signal if it has a significant fraction of noise, and will then subtract too many counts. GO's with bright sources who ardently desire to remove higher orders are strongly urged to do so during a visit to the EGO Center, with the help of EUVE scientists.

- Set `counts` to "yes" to make a raw count spectrum, "no" to create a flux calibrated spectrum.
- Set the range of the output spectrum in Å using `autosize`, to automatically select limits from the input images, or explicitly, using `minwave` and `maxwave`.
- The parameter `resample`, which specifies the binning of the output spectrum, is not currently implemented. The resolution of the output spectrum defaults to the lowest resolution present in the input spectra.
- The value of `table` should be set to the location of the ST table *detector.tab* in the directory *reference*. The table contains pointers to instrument reference data. The actual value of `table` should include the root table name and the ".tab" extension.

Of course, parameters can always be set on the command line. To combine three channel spectra from a white dwarf observation:

```
eu> euvcombine grumpy.sw,grumpy.mw,grumpy.lw detectors=sw,mw,lw
grumpy.com exposure=39873 maxorders=1 counts-
```

To combine raw counts from the medium and long wavelength spectra of a late star:

```
eu> euvcombine latest.mw,latest.lw detectors=mw,lw counts+
```

4.2.3 Example: flux calibration and comparison for bright sources

Lucky Dr. Ivana Ego! She has obtained the first EUV spectrum of the dwarf binary system MICK-EN Min in outburst, and it is so bright, that the spectrum is visible in the image of a single orbit!

In this example, Dr. Ego has re-extracted her spectra, and observes that the vast majority of the counts are between 250 and 540 Å. She edits the parameter file for **euvcombine** to bracket the non-zero part of the spectrum by setting `autosize` to "no", `minwave` to 170 and `maxwave` to 650. She then joins them with **euvcombine**, to produce a single spectrum with the exposure time and first order effective area divided out:

```
eu> euvcombine mickenmin.mw,mickenmin.mw,mickenmin.lw sw,mw,lw
mickenmin.com exposure=53729 maxorde=1 counts-
```

She then converts several possible model spectra from photons/cm^2/sec/Å, the units output by her modeling program, to photons/cm^2/sec. and compares them to her observed spectra with **compspec**:

```
eu> mkeuvspec @models_in @models_int dx=.2 linelist=no
integrate=yes
```

```
eu> compspec 1 mickenmin.com @all_models output=save_chi resolution=2
```

Note that the models and actual spectrum are compared at the approximate spectral resolution of the long wavelength channel. The reblocking of the spectra is taken care of by **compspec**.

Chapter 5

WORKING DIRECTLY WITH QPOE FILES

In this chapter, we will discuss more complex operations on your QPOE files, using macros for data selection, extracting 1-d spectra, and rebinning on the various attributes.

Section 5.1 describes the use of macro facilities, which are used to define new keywords for use in QPOE interface expressions (QPOE expression syntax is covered in section 2.2.3). Macros can be installed permanently in a QPOE or text file, to make it easier to select a certain data subset when the QPOE is accessed for processing. Several useful IRAF/EUV tasks are presented. For example, you may wish to use a filter similar to the exposure filter provided on your data tape, to select only events from "good" observing times as you re-extract a spectrum for flux calibration. There are a number of ways that you can direct the QPOE interface to the information that defines these good times, and different methods may be more convenient to different projects.

Section 5.2 discusses spectrum extraction with the tasks in the **twodspec.apextract** package and the EUV task **euvred.euvextract**. Section 5.3 describes rebinning your data to make time series of all or part of a spectrum. The **euv.euvtools** package contains several other tasks for manipulating QPOE files, which are described briefly.

If you are interested in more detailed information, you may also be interested in reading some of the general help topics in the PROS **xray** package. Instructions for accessing general help and information on QPOE files is printed to the screen when the **xray** package is loaded; type:

```
xr> help qpoe
```

to get a document which gives a reasonable overview of the structure and manipulation of QPOE files. Be reminded however, that many of the **xray** tasks assume that QPOE file arguments are ROSAT data files, and so cannot be used on EUVE data.

5.1 Using the QPOE Interface, II: Macro Facilities

Filtering QPOE files when they are accessed is a powerful method for selecting data subsets to process. A filter may be applied to any or all attributes in a QPOE file, and the individual filters can be quite complex, as we saw in section 2.2.3. One can select photons from certain sections of the detector, or time sections of the observation. One way of applying a filter to a QPOE file was discussed in section 2.2.3, in the form of a (filter) keyword expression. However, it quickly becomes tedious to type the same filter specifications each time a QPOE file is accessed, and some filters are much too long to even attempt to type on the command line.

Macros may be defined in the language of the IRAF/QPOE interface to simplify processing commands. You may recall from figure 2-1, the example of a QPOE "header" display, that we saw several lines that contained strings such as:

'TIME = ' macro[002] macro definition'.

This was **imheader**'s rather opaque way of telling you that the QPOE file contained macros that substituted more memorable expressions, like `time, x,` and `y` for the attribute names `d0, s8, s10`, etc. The macro facilities can also be used to include much more complex expressions that are used as filters by the interface.

Macros can be defined in several places, allowing for a hierarchy of precedence. We will discuss the major types of macros and how to use them, but this section is not meant to be an exhaustive treatment of the possibilities of the QPOE interface. When in doubt about whether what you want to do is possible, make an attempt and then email your questions to *egoinfo* and the PROS group at *rsdc@cra.harvard.edu*.

There are three major categories of macro references, which are summarized in table 5-1, and defined in more detail below:

1. **text replacement:** This is by far the most commonly used type of macro. When a defined macro name is encountered in a QPOE expression, the interface searches the QPOE file and any macro definition files for a macro definition with the same name, and the name is replaced by the definition of the macro. For example, the macro "x" is defined in the EUVE format QPOE files as "s8", the interface expression for a short integer attribute beginning at byte 8 of a line. Then the expression `[x=50]` is converted to `[s8=50]` before being parsed by the QPOE interface. This convention is very convenient, as a number of macros can be inserted in the QPOE file, thus making it possible to store a great deal of processing information right in the file itself.
2. **file include macro:** This looks a bit like the IRAF mechanism for suppling a list of arguments from a file "filename" to a task parameter with "@filename". Instead, the file should contain one or more complete QPOE filter expressions . As always, "filename" must include the complete path to the file from the current working directory. For example, a user who wanted to create a number of images from different time-slices of his QPOE file could use the command:

   ```
   im> imcopy 'mw.qp[@timeseg]' mw_orb30-33
   ```

   where the file *timeseg* contains the QPOE expression:

   time=(774683832.748086:774685492.6521089,774689532.3321639:774691192.2361872,
   774695231.9162419:774696886.7002651)

   In fact, one of the standard data products delivered with an observation, the exposure-corrected spectral image, can be duplicated with the command:

   ```
   im> imcopy 'mw.qp[@timeseg]' myimage
   ```

   where em timeseg contains the complete QPOE expression for the exposure filter:

time=(774683832.748086:774685492.6521089,774689532.3321639:774691192.2361872,
774695231.9162419:774696886.7002651,774700931.5003198:774702592.4283431,
774706631.084398:774708293.03642,774712329.6438759:774713800.107896,
⋮
774809209.2612002:774809521.5812038,774809549.2292047:774810897.8372231,
774814907.8212779:774815222.1892821)

Note that both these examples also make use of the text substitution macro which substitutes the string '`d0`' wherever the string 'time' is used. Macros may be nested this way up to 20 levels. This structure also processes newlines.

3. **command substitution:** This type of macro is mentioned mainly as a possibility with some limited applications in the IRAF/EUV software. The macro reference is given as a CL command surrounded by quotes, which is replaced by the output of the command. An example of this would be the expression `[block=4,time='tfilter']`, where **tfilter** is a task which renders a table of start and end times into a set of time intervals in QPOE filter format (see section 5.1.2). The parameter file of **tfilter** must be set up correctly beforehand. This includes setting the task mode to `a` (auto), to suppress queries. Since there are not many commands that produce output in valid QPOE expression format, use of this type of macro is limited.

Table 5-1: Form of QPOE file macro references

macro type	action	examples
text substitution	`macro` translated to part of expression from known definition	'sw.qp[`time`=(1:3000,9001:12000)]' `time` => d0
file inclusion	*file* is opened to find `macro` definition	'mw.qp[@*timefilter*]' *timefilter* => time=(1:3000,9000:12000)
command subst.	output of the named **task** concatenated to complete expression	'lw.qp[time=('**tfilter**')]' **tfilter** => 1:3000,9000:12000

In all three macro types, the replacement text may contain embedded spaces, tabs, and newlines, each of which is converted to just one space on expansion. Recursion is also allowed, meaning that the replacement text may contain references to other macros, which will be expanded as encountered.

The interface recognizes two categories: "global" macros, which reside in a separate file and may be referenced in any QPOE expression, and "local" macros, which are stored in the QPOE file and are specific to that file. Macros may be defined in more than one place; for example, a macro called `time` may be defined in both a QPOE file and in an eternal file such as *QPDEFS*. If a

macro of the same name is defined in more than one place, then a locally defined macro takes precedence over a global macro of the same name.

5.1.1 QPMEDIT creates and edits QPOE macros

Qpmedit is a task which can be used to list, add, change, or delete the macros stored in QPOE file. It is quite analogous to **hedit** and is used for examining, adding to, and editing the keyword values and macros in the "header" section of the QPOE file. The **qpmedit** parameter file supplies a number of options for safeguarding the header cards from incorrect and inadvertent changes. Figure 5-1 shows a parameter file setup to display the definition of the exposure filter macro **sw_gt**, using the special character "." to get a listing to STDOUT.

```
eu> lpar qpmedit


  files = "sw.qp"         QPOE files to be edited
 macros = "sw_gt"         macros to be edited
  value = "."             value expression
   (add = no)             add rather than edit fields
(delete = no)            delete rather than edit fields
(verify = yes)           verify each edit operation
  (show = yes)           print record of each edit operation
(update = yes)           enable updating of the image header
  (mode = "al")
```

Figure 5-1: **qpmedit** parameter file.

Users frequently find it convenient to use the task **imheader** to look at the global information in a QPOE file, but the listing from **imheader** can be misleading. This is because QPOE files are *not* images—the listing one gets by running **imheader** on a QPOE file is merely an attempt to represent the contents of the QPOE file in the more restrictive format of an IRAF image header. One example of this is that **imheader** lists the macro names in uppercase regardless of their true spellings. Macro names used in QPOE expressions , however, are case sensitive. One can easily view the true names of all the macros in a QPOE files by using **qpmedit** from the command line:

```
eu> qpmedit sw.qp * .
```

This command will list all the macro definitions in their entirety, with with correct spellings. You may want to compare the output from **qpmedit** with the header listing shown in § 2.1.2.

Qpmedit can also be used to add or delete macros of any size directly. To add a filter that selects only the center of the image. The commands:

```
eu> qpmedit lw.qp "spec" "1000:1040" add+
```

```
eu> lpar tfilter

   input = "det7filt"      Input table name
  (output = "lw.qp")       Output file
    (name = "deadtime")    Name of output macro
   (start = "stime")       Name of starting column
     (end = "etime")       Name of ending column
(cmdline = no)             Command line output flag
  (format = ".3f")         Interval format control
    (mode = "al")
```

Figure 5-2: **tfilter** parameter file.

```
eu> imcopy 'lw.qp[y=spec]' spec_lw.imh
```

will make a [2048 × 40] image of a central strip of a long wavelength QPOE.

Macros can also be edited by setting `add` and `delete` to "no", and giving the macro name and definition as before. For example, to redefine an existing exposure filter in a QPOE,

```
eu> qpmedit mw.qp exposure "@mytimes" add- delete-
```

will replace the former definition of `exposure` with the contents of the file *mytimes*.

5.1.2 TFILTER installs time filters made from ST tables

The task **tfilter** converts tables containing time intervals into a string in QPOE expresssion format and writes the string to one of several possible output locations. Its parameter file is shown in figure 5-2.

The value of `input` may be a list of several tables. **Tfilter** is capable of combining several lists of intervals and producing the intersection. Thus, if one has multiple sets of time intervals that need to be applied as a filter, each set should be entered into a table using either the **tcreate** command, or **tedit**. Each table should have the columns of start and end times in the same order, with the same column names. Then **tfilter** should be executed on the entire set of tables to make one filter string, which can be installed in the QPOE file as a single filter expression. This is important for users of IRAF versions earlier than 2.10.3 (see the "Warning" box in § 2.2.3). The ".tab" need not be used in **tfilter**.

Tfilter may be used to define either a global QPOE macro in a macro definition file or a local macro in the QPOE file itself. If the `output` file is a textfile of global macros, the macro definition with the name `macro` is appended to the file. If `output` is a QPOE file, the macro definition is inserted into the file. Alternatively, output may be directly to STDOUT. This provides a useful check and a way to combine making the filter with applying it to the QPOE in a command substitution macro reference, as described in section 5.1. The exposure time filter in each EUVE

QPOE is defined in the nominal reduction of the data at the EGO center using **dqselect** and added to the QPOE as a local macro called **sw/mw/lw_gt** using **tfilter**.

It is also possible to place a good times filter in a QPOE file using the data selection task **dqselect**, as described more fully in chapter 6, § 6.1.1. Why then, would one one use the **gtsave** command and **tfilter** to add the filter to a QPOE file? One reason is to work around the QPOE filter bug described in section 2.2.3. Since one cannot specify mulitple filters on a single attribute in a QPOE macro or expression, the only solution is to combine such filters into one. **Tfilter** can accept multiple tables as input, in which case it intersects the time intervals to produce one filter which passes only events which would pass each and every set of intervals input. Thus, one can merge the original exposure filter with other time filters by supplying both **w_gt.tab* and the new good times table to **tfilter**.

5.1.3 Global macros and *QPDEFS*

Global macros are defined in external ASCII files, called macro definition files. These macros are called global because they are available to any QPOE file expression. The CL environment parameter **qmfiles** controls which ASCII files the definitions can appear in. It should contain a comma-separated list of files to be searched for macro definitions (IRAF virtual paths are allowed). There is no default setting for this variable; it must be explicitly set by the user.

An example of a macro definition file exists in all IRAF environments, in the file *qpoe$QPDEFS*. It contains a set of QPOE interface parameter settings and some examples of macro definitions but its contents are not required to run any of the IRAF/EUV software packages. It's just a handy template for making one's own macro definition files. You can copy the file into your IRAF startup directory and add definitions as you create global macros. If you use only your personal version of *QPDEFS* to store global macros, **qmfiles** will be set with the command:

```
cl> set qmfiles "home$QPDEFS"
```

which can be added to your *login.cl* file.

When the QPOE interface encounters a macro name in parsing an expression, the QPOE file being accessed is searched for the macro definition. If it is not found, then the interface searches the macro definition files named in **qmfiles**. This is why local macro definitions take precedence. Be aware, however, that if you change the contents of a macro definition file during an IRAF session, the changes may not take effect until you reset **qmfiles** or restart IRAF.

5.2 Extracting Spectra from QPOE Files

Many GO's will want to make their own one-dimensional spectra from their wavelength-corrected QPOE files at some time. This section describes the use of the aperture extraction tasks in IRAF on EUVE spectra, as well as the simpler extraction task used in the initial reduction of GO data at the EGO Center.

5.2.1 Using APEXTRACT on EUVE QPOE's

As mentioned in Appendix B on EGO Center processing, there is a sophisticated set of tasks in the package **noao.twodspec.apextract** for extracting spectra from two-dimensional images. Using the "master" task **apextract.apall** allows a more careful extraction than that done during the EGO Center's nominal processing; if it is applied skillfully, an increase in signal-to-noise can sometimes be achieved. It is also likely that many GO's will want to re-extract their spectra from images that result from filtering, subject to limits on aspect, timing, orbital parameters, or count rates and dead times.

It is recommended that users **imcopy** the QPOE file to an IRAF image before any spectral extraction, as this will make interactive access to the file much more efficient. Be certain to apply at least the exposure time filter when creating new images; for example:

```
im> imcopy 'sw.qp[time=(sw_gt)]' sw.imh
```

would make an exposure-corrected spectral image from a SW spectrometer QPOE file. Other time filters may be applied at the same time as discussed in § 2.2.3. Two-dimensional exposure-corrected spectral images are supplied on the Guest Observer data tape as FITS images, and can be used without accessing the QPOE files if no other filters are needed.

Users who are not already familiar with the interactive operation of **apall** should consult the various help pages for the **apextract** tasks. More detailed information on the algorithms used can be found in IRAF User's Guides on **apextract** tasks, such as that listed in Appendix D.

Set up the **apall** parameter file for `onedspec` format spectra. Set `interactive` and `fittrace` to "yes", so that you can edit the aperture and examine your spectrum trace in the interactive mode of **icfit**.

Spectrum extraction with apall

1. **edit** an aperture, including background subtraction parameters
   EUVE spectra are very faint, and may be slightly rotated, and/or curved at the ends. They are typically difficult to trace, so apertures should at first be rather wide, to make sure the spectrum is included even when the trace wanders a little. Be sure to start with the parameter `t_nsum` of **apall** set to a large value, like 100, so there are enough counts in each sum for the peak-finding algorithm to locate the spectrum profile.

   If you are reducing a line emission spectrum, try to define your aperture on a feature. One can display an average of lines including the spectrum with **implot** to determine approximately where the profile will be clear, and then set the parameter `line` to that value (recall that "line" in the **apall** parameter file refers to image axis *perpendicular* to the dispersion direction).

   Use the "`b`" key to set background subtraction parameters. Since low background count rates render most fitting algorithms useless, background subtraction is usually best done by averaging over a large area perpendicular to the spectrum. The background subtraction package allows the user to set a sample area on each side of the spectrum, and to select an average, rather than a fitted background. Set the parameter `b_sample`[1] to the number of

[1] To set the background parameters in interactive mode using the colon commands, omit the "`b_`".

lines desired (relative to aperture center) and the `background` parameter for the extraction to "average". The average over the sample lines at each column is then subtracted from each pixel in the aperture before summing the spectrum.

Areas used for background subtraction can typically be 5–10 times the spectrum height, provided they are free of hotspots, pinhole images or other high-count features. The average of the specified lines is subtracted from each pixel in the spectrum aperture. IRAF will convert pixels to type "real." It can be useful to extract with `extras` set to "yes" so that a background spectrum will be extracted and stored in a second image line. Then you can check for detector features.

2. **trace** the spectrum, with an option to adjust the trace interactively.
   One major difference between EUVE spectra and typical optical wavelength spectra for which **apall** was designed is that EUVE spectra have much lower source *and* background count rates. The background level after a typical exposure can be only a few counts per pixel, and faint sources can have comparable count levels. This can make tracing the spectrum quite difficult. Emission line sources present the biggest challenge, as their spectra often have only bright spots (the lines) separated by large regions of nothing but background. For very bright continuum source spectra, much of the following is not necessary, and one can use **apall** as for optical wavelength spectra, but even bright continua may have low spots where the trace is lost.

   The tracing routine will make an attempt, possibly print a lot of "trace lost" and "trace recovered" messages, and then prompt you to enter interactive mode. The **icfit** display shows the points used for the fit, and the user can change the tracing parameters, refitting as often as necessary until satisfied with the trace. Try changing `nsum`, the number of lines summed to find the peak, and `naverage`, the number of peak locations that are averaged to make a fitting point, One can also vary the parameters `width`, `radius`, and `threshold` in order to prevent the trace from wandering off of the spectrum. Values of 10, 2, and 0, respectively, are good for a start. Finally, use `t_step` to try to step across faint regions of the spectrum; a value of 5 is a good first guess, but larger step sizes may help bridge faint regions if strategically placed. If all efforts at tracing fail, see the notes below.
3. **extract** the background-subtracted spectrum from the image.
   This is done automatically, with the option to review the results and save or delete them. Most spectra will have some subtraction errors, notably near the detector edges. Since the detector is circular, the (quasi-rectangular) background sample region near the ends will contain few, if any counts. This produces a spurious "rise" in the background-subtracted spectrum at the ends. Places where the artificially brightened rim areas are in the background region can cause over-subtraction. A record of the aperture and trace are saved in a database file.

Some spectra will turn out to be untraceable no matter what set of parameters is used. Most commonly, some part of the spectrum will be traced, but the tracing algorithm will break down in an extended faint region, even in regions where the spectrum is visible in an image display. In these cases it is sometimes possible to help **aptrace** along. Using the cursor in your image display, note down the (x,y) pixel locations of points along the spectrum, taking special care to get a number of points in faint regions. Then run **apall**, perform the usual operations up to tracing, and trace as best you can. When you get into the interactive curve fitting mode, use the "a" keystroke to add points with locations you recorded. You can use the "x" and "y" keys to fine-tune the positions of the points. Be sure to give the points a high weight. When you refit, you will force the fit to go though those points.

In any case, always check the trace of any faint spectrum against reality after completing the trace by changing the value of `line` to observe where **apall** thinks the spectrum center is at various columns of the image. Also make sure the trace agrees with the spectrum's appearance in the image display. By getting a valid trace, you may be able to extract using a smaller aperture than that used in the nominal reduction.

Remember one other important difference from typical optical wavelength spectra: the QPOE files (or spectral images made from them) containing the spectra to be extracted are *already* dispersion corrected. Therefore, after spectral extraction there is no additional step to create a wavelength scale for the extracted spectrum. Instead, the wavelength calibration is inherited from the parent two-dimensional image.

5.2.2 EUVEXTRACT performs spectral extraction on EUVE images

The task **euvextract** is used in the EGO Center's nominal data reduction to do extraction and background subtraction on a standard rectangular aperture in each 2-d spectral image. The user defines the centerline and width of a spectral and a background aperture in the parameter file rather than fitting the spectrum, and the background is averaged, scaled by the aperture width, and subtracted without fitting. Details of the methods used in EGO Center reductions may be found in Appendix B, section B.3. Since **euvextract** has none of the capabilities of **apextract** for finding and tracing apertures, it will be of little interest to most observers.

5.3 Attribute Binning and Timing Analysis

5.3.1 QPBIN rebins on any attribute

The task **qpbin** is used to rebin or "project" a QPOE file with respect to one of its attributes. Since rebinning along the coordinate axes is already easy to do in IRAF image format, **qpbin** is most helpful for time binning, which converts QPOE's into a counts vs. time image format. In practice, it is usually used to make light curves of some interesting section of a spectrum. Since timing information is lost when copying to an IRAF image, light curves must be made from the (unsubtracted) QPOE files. An example of using **qpbin** in this way is given in § 5.3.2. Typical parameter settings for making a light curve **qpbin** are shown in figure 5-3.

The parameter file is set up to project onto the time attribute, making a 1-d image with 100-second bins. Since the range parameters `x1` and `x2` are both "INDEF" **qpbin** will create a many pixels as needed to cover the entire observation, starting with the earliest time. One of the parameters `dx` and `nx` must always be defined (i.e. not INDEF); one may set the values of both `dx` and `nx` to define a time window and image size. If the values of `dx` and `nx` conflict, `dx`, the bin size, takes precedence, and **qpbin** creates as many pixels as necessary to span the interval.

Most users will also want to restrict **qpbin** to a small section of the spectrum. For EUVE QPOE's, this means using `rect` expressions on the key coordinates (`x` and `y`), or filters on the detector coordinates (`dx` and `dy`), or the wavelength and imaging angle (`lambda` and `theta`), to cut out a rectangular region as high as the spectrum and as wide as needed to include the wavelength(s) of interest. With emission line spectra, for example, it is easy to make a rectangle

```
eu> lpar qpbin

      input = "lwdet.qp[rect=(1230:1400,1205:1600)]" QPOE file
     output = "myobs_lw304"   Image file
(attribute = )                QPOE field name
        (x1 = 774683832.74808) Lowest value
        (x2 = 774815222.18928) Highest value
        (dx = 30.)            Delta value
        (nx = 64)             Number of output bins
     (units = "seconds")      QPOE field units
     (range = yes)            Provide range information
      (mode = "al")
```

Figure 5-3: **qpbin** parameters for making a light curve.

which covers just one emission line. Users can inspect the QPOE file with their image display tools to determine the boundaries of the desired region.

Time binning of satellite data can pose an additional problem because of gaps in the data. Usually, **qpbin** divides the interval `x1-x2` by `dx` or `nx` and bin together any events it finds in each of the time bins. Because the EUVE detectors turn on and off at varying intervals as the spacecraft passes in and out of nighttime and the SAA, it is difficult to predict in advance how many seconds of any time bin include valid data.

For this reason, it is recommended that the exposure filter, which may also incorporate user-defined quality filters, be used in conjunction with the `range` parameter. When the exposure filter is referenced on the command line, and the parameter `range` = "yes", **qpbin** will compute the number of seconds in each bin that would have passed the exposure filter. This information is included in the output image as the value of a corresponding pixel in a second line.

To normalize the pixel values, the user can use the **imarith** task to divide the first line of the output image by the second. Be sure to set `divzero` in **imarith** to "0" in case some bins have no valid times. This produces a light curve of the count *rate* versus time. Appropriate error bars can be computed using **imarith** and the information in the two lines of the output image. For bins with very small effective exposure, there will be very few counts and the data will be noisy.

Background subtraction poses another complication. The background count rate can vary substantially during an EUVE observation, and **qpbin** makes no background correction. The user can perform this task by making a light curve of a background region with **qpbin**, using, say, the same `x` filter but a different `y` filter, and then using **imarith** to subtract the two light curves.

5.3.2 Example: making a light curve

This example will demonstrate the use of the task **qpbin** by Dr. Ivana Ego to make a light curve of a selected spectral region.

First, Dr. Ego sends the QPOE file *sw.qp* to an image display, and determines the x coordinates of the wavelengths she wants to study. Using the cursor to choose a region that spans the spectrum (pixels 1020–1030) in **y**, Dr. Ego decides to look at the ~ 22 Å region between 106 and 128 Å. She has chosen a background region of the same width as the source region, but above the spectrum and 50 lines high, from pixels 633–956 along the **x**-axis. If no image tool is available, one can also find these boundaries by viewing various sections in **implot**.

Dr. Ego sets parameters of **qpbin** as shown in figure 5-3. She knows that the exposure time filter from the nominal reduction at the EGO Center is stored in the table *sw_gt.tab*. She uses the command:

```
eu> qpbin 'sw.qp[rect=[1020:1030,x=633:956],time=(sw_gt)]' scurve
```

to extract the light curve into the image *scurve.imh*. She extracts a background light curve with a similar command:

```
eu> qpbin 'sw.qp[rect=[1070:1080,633:956],time=(sw_gt)]' bcurve
```

and subtracts it from *scurve*:

```
eu> imarith scurve[*,1] - bcurve[*,1] scorr
```

where she has chosen a background region of the same size as the source region, but 50 lines above the spectrum. Dr. Ego then corrects the exposure using **imarith** and the "time line" in line 2 of the subtracted image:

```
eu> imarith scorr / scurve[*,2] final_curve
```

This final step derives a count rate and corrects for the exposure variations between the time bins. Our final product is shown in figure 5-4.

The count rate shown is over the entire wavelength range chosen with the QPOE filter in **x**. The bins with the very highest rates are actually derived from low-exposure low-count times, and so will have large error bars. Dr. Ego constructs the error bars for her light curve from the information at hand, using the commands:

```
eu> imarith scurve[*,1] + bcurve[*,1] variance

eu> imfunction variance deviation sqrt

eu> imarith deviation / scurve[*,2] error
```

construct an image containing the properly scaled Poisson errors in the final light curve. The result is illustrated in figure 5-5.

Fortunately or unfortunately, Dr. Ego has found that this feature of her spectrum is non-variable. The small deviations apparent in figure 5-5 illustrate variations of up to 5%, due mainly to

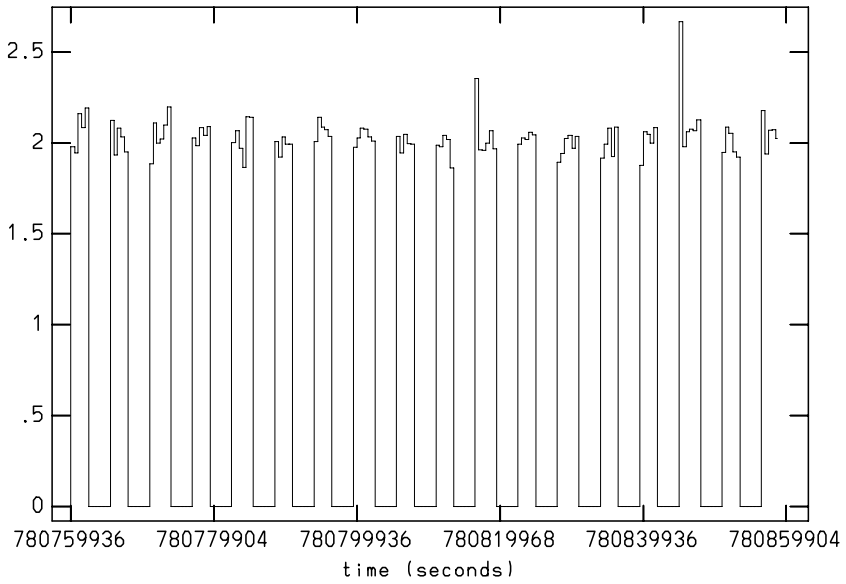

Figure 5-4: Light curve of a portion of a spectrum in the SW spectrometer, with background subtraction and exposure correction.

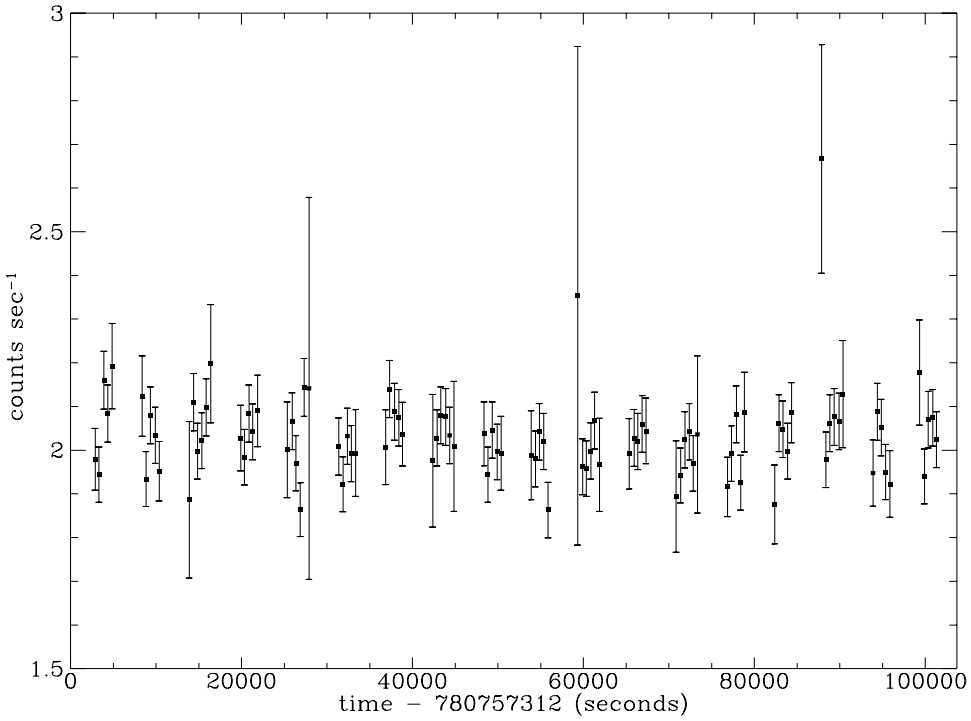

Figure 5-5: Light curve of a portion of a spectrum in the SW spectrometer. as in previous figure, but showing errors.

unremoved Primbsching/deadtime effects.

5.3.3 Example: making time-dependent spectra

Having made a light curve of some spectral region and determined the time boundaries of some interesting event (perhaps a stellar flare), Dr. Igor Ego may want to extract a spectrum over a particular time interval. The process of actually extracting the spectrum is no different from the techniques for extracting a spectrum from any EUVE image (§ 5.2), except that an additional time filter must be applied when the QPOE file is copied into a spectral image.

Because of the bug in the QPOE filter processing discussed in § 2.2.3, Dr. Ego must use the task **tfilter** to combine the his desired time intervals with the exposure time filter. Let's suppose Dr. Ego wants to extract a spectrum from the SW spectrometer taking only photons from the time interval (762023822:762024792). In order for **tfilter** to be able to read this interval it must be contained in a time intervals table like the ones produced by the task **dqselect**.

The easiest way to make the table is to modify another table with a similar form. Dr. Ego has received a set of tables used to define the exposure time filter on his data tape. The table for the SW spectrometer it is called `sw_gt.tab`, and is in his working directory.

Dr. Ego makes a copy of the first row of this table with the command:

```
ta> tselect sw_gt mytimes "row()=1"
```

and then uses the task **tedit** to change the start and end times in the table to the ones he wants and write the table to *mytimes.tab*

If Dr. Ego wanted to extract several non-overlapping time intervals at once, he would simply include more lines in the table. See the help page for **tedit** for details on its use, or type "`^-d help`" on the prompt line.

Next, the user must intersect their time intervals with the nominal EGO Center exposure intervals and create a QPOE macro named `mytimes` containing the resulting filter expression with:

```
eu> tfilter mytimes,sw_gt output=sw.qp name=mytimes
```

Now this filter is applied in creating an image from the QPOE file with:

```
im> imcopy 'sw.qp[time=(mytimes)]' sw_flare.imh
```

The image can then be used for spectral extraction and other analysis steps.

Chapter 6

DATA QUALITY SELECTION WITH ST TABLES

Most of the EUVE ST tables for an observation contain engineering data, electronic monitor readings, and other quantities reported by various spacecraft systems. They are mostly control voltages and logic signals, which track instrument health, satellite aspect, and conditions like detector count rates. Each monitor is subject to upper and lower limits, beyond which the associated subsystem's functioning is considered suspect or wrong. These limits should rarely be overstepped in data which has been accepted for observations, but many observers will have more stringent standards of data quality than the pre-set "hard" limits specify.

In cases when some factor correlated with a particular monitor affects data quality, observers will want to select subsets of their data to meet certain criteria for special processing, using the data stored in *table1–table7*, or other tables derived from this data, as guidelines. As we saw in sections 2.2.3 and 5.1, QPOE data selection "filters" are made by limiting the values of one or several of the file's attributes on accessing the file, using a QPOE file interface expression or an equivalent macro.

High voltage monitors tell when detectors are operating normally. Aspect readings stabilize when the target is found, and high detector count rates indicate periods with higher deadtimes. The obvious example, which is provided for in the nominal reduction, includes only data from times during orbit night when the Spectrometer detectors were turned on. This chapter is about how the GO can determine which data they want to include in their processing, and how to specify these subsets in the language of the ePOE interface.

6.1 Making Time-Correlated QPOE File Filters

Because the information in the data tables is time-ordered, limits on any of the quantities tabulated can be conveniently translated into a filter on the event time attribute in the QPOE file. You may have already used the IRAF/EUV task **euvred.dqselect** to plot various table columns as time series In section 3.2.2. The **euv.euvred** package provides several interactive tasks to help GO's create filters from these plots and link them to the QPOE interface.

6.1.1 DQSELECT creates and installs QPOE filters

Dqselect is a task which allows the user to conveniently view the many engineering and science monitors stored in the telemetry tables and to construct time filters for QPOE files based on the values of those monitors. The user can load any table, plot table columns against time, set and mark limits on the values, and command the task to create a table of valid times. If you have

not done so already, refer to Appendix A for definitions of the data in the telemetry tables and indications of their usefulness to GO's. Read the **dqselect** help page for a detailed description of the operation of the task; here we will discuss some terminology and try to clarify some points.

The parameter file for **dqselect** is shown in Fig. 6-1.

```
eu> lpar dqselect


 tables = "observation/table[1-7].tab" Monitor Tables
(limits = "")              Limits Tables
   (gap = 1)               Time Gap
 (imode = "line")          Input mode
(format = "14.4f")         Interval format control
(device = "stdgraph")      Graphics device
  (mode = "al")
```

Figure 6-1: **dqselect** parameters with default values for processing EUVE data.

The `tables` parameter specifies the tables containing monitors of interest. The nominal setting shown will load all the EUVE telemetry tables except the event list. The parameter `imode` determines whether **dqselect** starts in either "line" or "cursor" mode. In line mode, users enter commands to a command line cursor, choosing from a displayed menu. Cursor mode is the normal IRAF cursor mode; all line mode commands are still accessible as colon commands, but cursor control is in the graphics display. You may want to use `limits` to input a table of predefined limits if you have saved limits from a previous session in **dqselect**. The parameter `gap` is also very important. This is the number of readings that may be missing from the sequence in the table before the line of the plot is interrupted to show a gap (The frequency of readings is predetermined, but the transmission of telemetry is seldom perfect.). Make `gap` equal to 1 on the first viewing, so you can see everything (well–close; see further discussion below). For an example showing **dqselect** display in use, see § 3.2.2.

The keys to understanding **dqselect**'s filtering capabilities are the concepts of "limits", "valid times" and "gaps". In effect, the combination of limits and gaps define the valid times when instrument is in some desired state.

6.1.1.1 User-defined monitor limits

The first part of most **dqselect** sessions is to display the monitors that are of most interest, and determine the levels that correspond to the desired state.

The command `display` brings up plots of monitors from the entire observation against time. Because the files are very long, **dqselect** often cannot display all the data points, given the resolution of the display device. In this case, the data in the tables is evenly sampled at the smallest interval which can be fully displayed. This use of sampling implies that high-frequency variations in the data may not be visible. If it is crucial to see the data at full resolution (every row), a workaround is possible using the **tables.tselect** task. Make a table containing a subset of the original table with as many rows as can be displayed without sampling (typically 500),

and run **dqselect** on the new table. This step may have to repeated many times to encompass an entire observation, so it is not really practical except for spot checking, or for treating short intervals that are problematic. Fortunately, it is usually not necessary to see every detail in order to set appropriate limits. How **dqselect** handles the table is independent of the display limitations, as is discussed in the next section.

Each monitor in the tables **dqselect** has read can have limits set on it. Limits for each monitor determine upper and lower bounds of an acceptable range of monitor values. Limit values for any number of monitors may be entered from the display with the `edit` command, or they may be reloaded after being saved in table format from a previous session, using the parameter `limits`. Once limits are set, they appear on the monitor display as horizontal bars at the left axis. The current limits can then be saved as an ST table at any time. Thus, by effectively drawing lines across the time series plot at the desired upper and lower limits, one can define times when the data were produced under the desired conditions. The capability to set limits on several monitors at once allows the user to be very selective indeed.

6.1.1.2 Defining valid time intervals

Once appropriate limits are set, a template for valid times can be created from the times when each monitor transits the limit value. Valid times are computed by **dqselect** when the `gtgen` command is issued. This command causes valid times (sometimes referred to as "good times", hence the command name) to be computed for all monitors which currently have limits set on them; then all the sets of valid times are intersected, producing the set of valid times when *all* monitors are within their limits. These times are saved in memory; they may be written to a QPOE file as a macro using the `filter` command or saved in a table using the `gtsave` command.

An important point to remember is that `gtgen` will combine each set of valid times with any existing set, either previously generated or reloaded in the current session. If, having generated valid times for some monitors, a user wishes to repeat the process on a different set of monitors, any previously generated valid times must be stored and removed by executing the `gtnew` command, and any unwanted limits must be deleted using the command `delete`, prior to re-issuing the `gtgen` command.

Also, if your display device does sample the input, keep in mind that sampling is *not* done when generating valid times. Therefore, the list made by `gtgen` may show breaks in the valid time intervals not visible in the graphic display. In this case, the time intervals should be considered more accurate than the display.

So why use `gtsave` and `tfilter`?

...instead of the **dqselect** command `filter` to write the filter into a QPOE file? One reason is to work around the QPOE filter bug (see the warning in § 2.2.3. Since that bug prevents the user from being able to specify multiple filters on a single attribute in a QPOE expression, the only solution is to combine the desired filters into one, using something like **tfilter** (section 5.1.2). **Tfilter** can accept multiple tables as input; in such cases it intersects the intervals to produce one filter which passes only events which would pass all of the input filters. The bug will be fixed in IRAF version 2.10.3.

6.1.1.3 Gaps in the data

There are times when data is not present for a monitor, referred to here as gaps. The most common example is the roughly 2/3 of each 90 minute orbit of EUVE when the spacecraft is in daytime, the detectors are off, and no events are recorded. Almost inevitably, there are other smaller gaps for various reasons, ranging from data losses due to corrupted packets, down to the reporting interval of the monitors (discussed in Chapter 3.1). During these gaps, the values of the monitors are unknown. However, gaps do not necessarily indicate any malfunction or change in the monitor; they are just lapses in reporting.

The user must specify how long a data gap can be (in units of the reporting interval) before it is considered significant. This is done using the `gap` parameter, which can be set in the parameter file or using the local command `gap` in **dqselect**. When **dqselect** is constructing the valid times for a monitor, it compares the length of the time between reported monitor values, in units of that table's reporting interval, to `gap`. Then intervals between table times that are greater than (`gap` $\times$ report interval) are considered an invalid interval. However, because of sampling by the graphics device, gaps may not appear as blanks in the display unless they are fairly large.

As an example of how the gap setting and `gtgen` interact, assume limits of 2–40 have been placed on the monitor `Det4ADct` (the ADC counts for the SW spectrometer). This monitor is in `table1.tab`, for which the reporting interval is 1.024 seconds. `Gap` is set to 5. At some point, the monitor goes from a value of 50 to 35; **dqselect** considers the time of the row containing the value 35 as the start of a valid time interval. The monitor now stays between 2 and 40 for many rows. At some point, two consecutive rows are found to be separated by 3.072 seconds, rather than the expected 1.024. Since this is a gap of 3 reporting intervals, and `gap` is set to 5, this is not a major gap, so **dqselect** ignores the gap and continues. At a later point, a gap of $\sim$ 3600 seconds is encountered (the spacecraft entered daytime). This is clearly a major gap, so **dqselect** terminates the current valid time interval with the time of the last reported value before the gap. Then it starts a new valid time interval (assuming the value of the monitor is still in the range 2–40) with the time of the first reported value after the gap.

6.2 Specialized Data Quality Selection Tasks

6.2.1 MKPRIMBSCH makes deadtime correction tables

Mkprimbsch computes the correction for primbsching and TIF deadtime which should be applied to the data in order to get the most accurate count rates. (The origins of data loss due to primbsching and deadtime are discussed in more detail in Appendix B, § B.5.1.) There is no IRAF task available to accurately correct the data for these effects at present, but **mkprimbsch** supplies a limited capability for investigating the severity of the problem, determining time filters which can eliminate periods of excessive data loss, and/or calculating approximate corrections.

The task **euvred.mkprimbsch** computes the amount of deadtime and primbsching which has occurred, based on count rates reported by the detectors and TIF's. It reads information from the raw telemetry tables and from a table of TIF deadtime calibrations in the reference data. From these, it produces a table containing columns of reference data, and multiplicative correction factors for each of the detectors.

The parameters for **mkprimbsch** are shown in figure 6-2. Information from several sources is required for the task to run; the parameters given here will work correctly with the nominal arrangement of the data on disk (see figure 0-5 in Part I). Users should not change the value of `ctsoffset`. See the help page for a detailed description of the calculations and the output table columns.

Mkprimbsch can also print various messages on STDOUT to inform the user of its progress. In general, is it not crucial for the user to understand these messages. However, users should take note of the warning: "values not found in deadtime table". This message indicates that some values of the count rate on the detector exceeded the calibrated range of the TIF deadtime. The resulting output table will still be useful, but there will be values of INDEF on the lines where **mkprimbsch** could not compute a correction.

Once the primbsching/deadtime correction table is produced, the GO can use the results to apply an approximate arithmetic correction to count rates, or filter out frames with corrections higher than some limit. Each row of the `output` table records the timestamp, "TableTime", the telemetered and quadrant counts for each detector, correction factors, and errors. This amounts to a LARGE number of columns; users may find the list created by the task **ttools.tlcol** helpful in locating the interesting columns. Many of the the counts are trivial or uninteresting, as the quadrants may not be used,[1] or do not have the spectrum.

The output quantities of most interest to Guest Observers will be the columns `Det4Q0dpc`, `Det5Q1dpc`, and `Det6Q1dpc`, which contain the combined primbshcing and deadtime corrections, as functions of time, for the spectrum quadrants of the SW, MW, and LW spectrometers, respectively. For example, if some row contains the value 1.05 for `Det4Q1dpc`, this means that, at that time, the number of counts measured in the SW spectrum should be increased by five percent to get a true value.

Since **mkprimbsch** is run for each observation as part of the routine EGO Center data reduction, it may not be necessary for most GO's to use it. GO's who received their data under EGO Center software version 1.3 or earlier will need to run **mkprimbsch** themselves to obtain the table of correction factors. They may also wish to consider whether reprocessing their data with a more recent version of the reference data is appropriate.

6.2.2 Example: applying mkprimbsch and tfilter to a spectrum

Here is an example using the primbsching/deadtime correction table and the task **dqselect** to time-filter a QPOE file. Consider the case in which the user Dr. Ivana Ego is concerned about the effect of deadtime and primbsching on her data. Dr. Ego is assumed to be in the working directory with the usual arrangement of the data directories. She begins by inspecting the deadtime correction table supplied with her data, called *primbsch.tab*, in **dqselect**. Assuming the **dqselect** parameters are set as in figure 6-1, Dr. Ego starts the task **dqselect** with the command line:

```
eu> dqselect primbsch
```

which loads the primbsch table and then displays the command menu in line mode. Now Dr. Ego

[1] Quadrants 2,3 for detectors 5 and 6; 1,2,3 for detector 4 do not exist.

```
eu> lpar mkprimbsch

    output = ""               output table name
(detectors = "sw,mw,lw")      list of detectors to process
  (refdata = "reference/detector.tab") detector reference table
 (infomsgs = yes)             give informational messages
 (warnmsgs = yes)             give warning messages
 (diagmsgs = no)              give diagnostic messages
 (negprimb = no)              report negative primbsch values
  (readper = 100)             percent of input file to read
  (obsdata = "observation")   path to observation directory
   (events = "table0")        event table name
 (gaptimes = "table1")        gap times table
  (quadcts = "table2")        quadrant counts table
(ctsoffset = 1.024)           quadrant counts reporting offset in seconds
     (mode = "ql")
```

Figure 6-2: Parameters for the task **mkprimbsch** with nominal values.

displays the correction for the LW spectrometer by entering the **dqselect** command:

```
Command> display det6q1dpc
```

(recall that **dqselect** commands are case insensitive). The result is shown in figure 6-3.

For most of the observation, the correction is about 1.02, indicating the count rate is underestimated by 2%. The correction rises to 1.04 or more near the ends of each nighttime pass, and increases to values above 1.1 in several other short intervals. From a scientific point of view, deadtime/primbsching correction may not be necessary. If Dr. Ego does nothing to correct the data, she will underestimate the count rate by roughly 3–4 percent, which is acceptable in some cases. Unless of course, she is so unlucky as to have a highly variable source which has emitted many of its photons during very brief intervals. Because of data sampling by **dqselect** for display purposes, Dr. Ego cannot be sure that the correction is not larger in some very short intervals which are not shown. One might make a first-order correction with a minimum of effort by approximating the correction as a constant; after the analysis is complete multiply any derived flux by, say, 1.03.

However, let's suppose that Dr. Ego believes the source might have had sudden variations, and wants to proceed more cautiously. She recalls that whatever the limitations of her display device, **dqselect** will compute the good times for a given set of limits correctly from the table. She can eliminate all times when the correction was larger than 1.05 from inclusion in her data set by setting an upper limit of 1.05 and generating the good times table. Then she will have a guaranteed upper limit of 5 percent on the error in the derived flux, regardless of the nature of the source, and can estimate the true error as somewhat smaller.

Still in **dqselect**, Dr. Ego creates a QPOE set of valid times. First she sets limits on the long

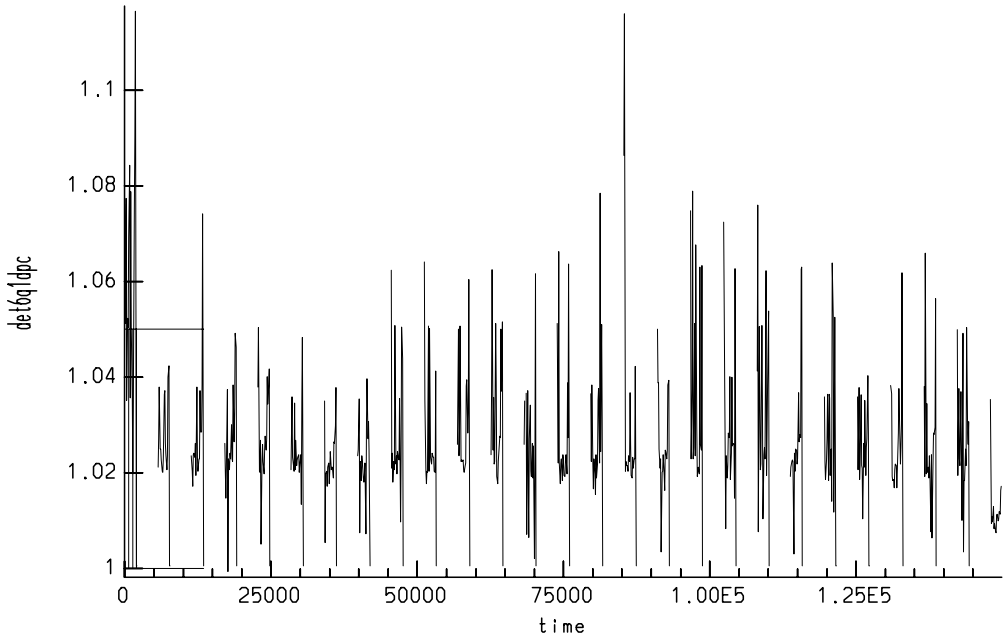

Figure 6-3: Deadtime/primbsch column displayed in **dqselect** with limits.

wavelength correction column, by giving the command:

```
Command> edit det6q1dpc
```

Dqselect requests that she enter the limits in the graphics window. Dr. Ego could click the mouse or press a key to set limits, but she uses colon commands to give more precise values. The commands:

```
:y 1
:y 1.05
```

set the limits to 1.0 and 1.05. **Dqselect** marks the limits on the display of the correction with horizontal bars at the left edge, as shown in figure 6-3.

Next Dr. Ego generates the valid times with

```
Command> gtgen
```

and saves the results to a table called "goodprim" with

```
Command> gtsave goodprim
```

Then she leaves **dqselect** with the command **"q"**. Let's look at the contents of the table *goodprim.tab*. Dr. Ego can view the table with **tdump**, **tprint**, or **tread;** the first few rows look like this:

```
        stime                   etime

780714437.0430151       780714439.0910149
780714617.2670169       780714619.3150171
780714715.5710188       780714717.619019
780714756.531019        780714758.579019
780714830.2590201       780714832.3070201
780714871.2190208       780714873.2670209
780714877.363021        780714879.411021
780714903.9870209       780714906.035021
780714910.131021        780714912.1790208
780714983.8590218       780714985.9070221
```

where each row represents one valid time interval and the columns are the start and end times in seconds, relative to some reference time. Notice that the time intervals are short; this is because the count rate, and with it the primbsch correction factor, varies a great deal during the observation, as is seen in the display in figure 6-3. Using the task **rowcount**, Dr. Ego finds that there are 1354 rows in the table.

In order to use these valid times to filter the QPOE file, Dr. Ego needs to construct a QPOE filter expression from them, and install them as a time filter in the QPOE file. As mentioned in § 2.2.3, the QPOE files already contain time filters made from tables of valid times for the exposure. The tables themselves are also delivered with the data. Since multiple filters on an attribute are currently (IRAF 2.10.2) unreliable (see the warning box in section 2.2.3), the primbsch filter must be merged with the exposure time filter. The merge can be done using the task **tfilter**, as discussed in section 5.1.2. Dr. Ego sets up the parameters as shown in figure 5-2 and types:

```
eu> tfilter goodprim,lw_gt output=lw.qp name=goodprim
```

where *lw_gt.tab* is the pre-supplied table with the valid exposure times. **Tfilter** intersects the two sets of time intervals and writes one filter expression into the QPOE file as a local macro named `goodprim`.

Now Dr. Ego applies this filter to make an image from the QPOE file with only photons from correct exposure times *and* the allowed primbsch states.

```
eu> imcopy 'lw.qp[time=(goodprim)]' lw_clean.imh
```

The image can then be used for spectral extraction and other analysis steps.

Though this example has focused on investigating the primbsch/deadtime correction, similar steps would be used for constructing any other time filter:

- The user can run **dqselect** on the tables that contain the monitors of interest,
  If the user has time intervals from some outside source, and wishes to make a filter, the start and end times can be entered into a table of the same format as the exposure table.

- Then **tfilter** can be used to combine all new interval tables with the exposure time table, and install the new filter in the QPOE.

The possibility that this application of **dqselect** may produce a very large number of very short valid times is considerable. Unfortunately, there seems to be a limit to the length of filters that can be parsed reliably in some situations, but the conditions are not well known, and there is no sure workaround for this problem at this time. This limitation, combined with the fact that only a single filter for each attribute can be parsed when the QPOE file is accessed by an image-handling task in IRAF (version 2.10.2 or earlier), may pose problems with data selection according to deadtime values.

6.2.3 MKASPECT makes aspect correction tables

The task **euvtools.mkaspect** creates a table of time-tagged sky coordinates from the aspect quaternions in an ST table (normally *table1.tab*). See the IRAF help page for a complete list of the parameters.

The output table has four columns: time, RA, Dec and spectrometer roll angle. In addition, if `distance` is "yes", the off-axis angle from the target coordinates `ra` and `dec` to the telescope boresight is calculated as a positive angle. No azimuthal information is computed directly.

The coordinate aspect table `output`(sw,mw.lw).tab can be read and the coordinates and/or distance angle used to generate time filters using **dqselect**. As with all other filters they must be incorporated into the QPOE file by using **tfilter** to merge all time filters into one (IRAF 2.10.2 or earlier). An example display of such a table is shown in figure 6-4.

6.2.4 BACKMON makes tables of geographic orbit parameters

Sometimes the information contained in the data tables is indirect and cryptic at best. Notably, the aspect and ephemeris of the spectrometer and satellite are recorded as quaternion components. The task **backmon** (so named because it can be used to help monitor and reduce sky background) was written to aid GO's who wish to limit their data set according to geophysical parameters. Given the location of the ST table containing the aspect reports (*table1.tab*) and a set of reference data, **backmon** calculates a list of selected orbit parameters as a function of time and puts them out in a new ST table. A sample parameter file is shown in figure 6-5.

Backmon uses the parameter `skip` to determine how often it should sample the aspect table (every 50 lines in the example above). It calculates instrument look direction, satellite position and velocity vectors, sun position, angles between the DS/S and anti-sun, and the DS/S and anti-sun with the satellite zenith, and the satellite magnetic latitude which correlates with the intensity of the particle environment. The time of each sampled line is recorded, and all latitudes and longitudes can be converted to RA and Dec using the Greenwich Meridian location.

Starting with data processed under EGO Center software version **1.4**, **backmon** is run for each observation as part of the standard data reduction. GO's whose data were processed under earlier software versions may need to run the task themselves. The output table can be loaded into **dqselect** like any of the standard telemetry tables, and used to create time filters on QPOE's in exactly the same manner.

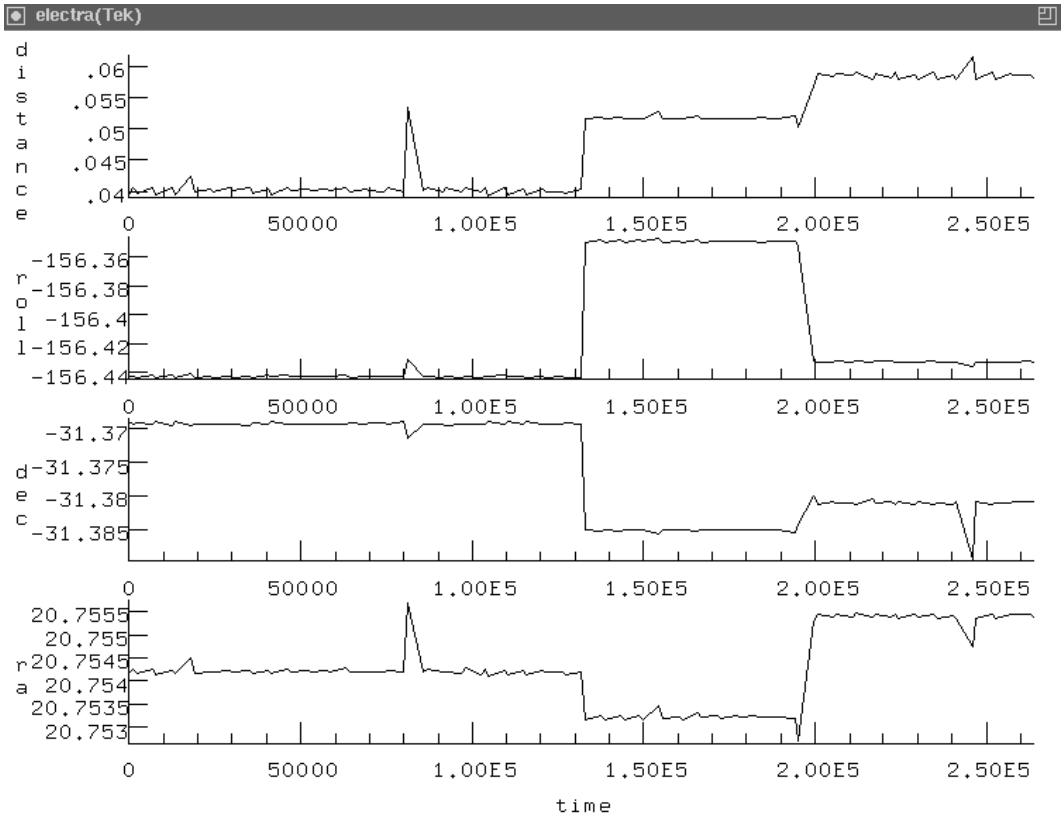

Figure 6-4: Aspect table columns produced with **mkaspect**.

6.3 Other Table Utilities

This section describes additional IRAF/EUV tasks that handle tables, and can be useful in some applications.

6.3.1 ROWCOUNT determines table length

The task **euvtools.rowcount** allows the user to count the number of rows in a data table in which a selected column matches (or does not match) a certain value to within a specified range. If the `range` parameter has the default value `INDEF` the value match assumes the local machine precision. **Rowcount** can be used to find the number of rows (events) in *table0.tab* within some small range of times.

6.3.2 QCALC calculates quaternions

The task **qcalc** is a quaternion calculator that recognizes some common forms of astronomical coordinates. As many as 26 quaternions (4-vectors) can be loaded into registers named `"a,`

```
eu> lpar backmon
```

```
      output = "mysource_bmons" Name of output ST table
       (skip = 60)              Input skip factor
 (maxtimegap = 600)             Max gap in time (secs) for Sat. Vel. Vectors
   (dodsspos = "yes")           Make cols for DSS Look Direction Long & Lat
   (dosatpos = "yes")           Make cols for Satellite Long, Lat & Height
   (dosunpos = "yes")           Make cols for Sun Long & Lat
   (doangles = "yes")           Make cols for ASunZen, DSSZen, DSSASun angles
    (dogmpos = "yes")           Make col for Greenich meridian position
   (domaglat = "yes")           Make col for Magnetic Latitude
     (dommjd = "yes")           Make col for Modified Modified Julian Date
   (dosatvel = "yes")           Make cols for Satellite Velocity Vectors
    (obsdata = "observation")   directory of ST observation tables
     (aspect = "table1")        Name of input aspect/ephemeris ST table
    (refdata = "reference/detector.tab") Name of ST table for reference data
  (detectors = "sw")            Name of detector (for boresight quaternions ref
    (hdrkill = "RATE_???")      list of header parameters not to copy from inpu
       (mode = "al")
```

Figure 6-5: Typical parameter file for **backmon**

`b, c...` and manipulated through a number of vector operations. Numbers enclosed in angle brackets (<>) are interpreted as celestial coordinates in decimal degrees, and those in curly brackets ({}) as coordinates in hours, and are converted to quaternions automatically. The user can enter the calculator in interactive mode by typing the task name on the command line:

```
eu> qcalc
```

and begin loading vectors and typing commands after the prompt:

```
qcalc>
```

perform calculations, and exit with a "control-d", as in the following example:

```
qcalc> a = .92 0 .73 0

qcalc> b = 0 1 0 0

qcalc> c = a * b

qcalc> c
```

```
 0. 0.92 0. -0.73

qcalc> d = 12 54

qcalc> &d

 -0.5877852522924731 7.198055546404513E-17 0.8090169943749474

qcalc> !d

 180. 54.

qcalc> ^D
```

Typing "&" or "!" before a register name signals the task to print the contents in vector or degree form, respectively, with vector as the default. Output in hours is not yet supported.

6.3.3 ECLIPTIC converts equatorial to ecliptic coordinates

This task converts between equatorial or celestial coordinates and ecliptic coordinates. **Ecliptic** reads `input` from the STDIN or from a file, using the "@file" notation. The `in_coords` may be in either decimal RA and Dec or Latitude and Longitude. If ecliptic coordinates are input, it will convert to equatorial coordinates. Be sure to use the correct `epoch` for the coordinates. Set `print_coords` to "yes" to print both the input and output coordinates.

DOING YOUR OWN DATA REDUCTION

7.1 The EUVE Event Pipeline

There are two main reasons why GO's will want to redo the data reduction process. By "redoing the reduction", we mean converting the time-ordered event list in *table0* to a set of three QPOE files with distortion and wavelength correction, using the event pipeline **cep** in IRAF. Some GO's will find that the binary format of the QPOE files provided on their tape cannot be read by their home system. Running the pipeline on your own IRAF system will produce a set of locally compatible QPOE files. In addition, as new knowledge about the operation of the spectrometer becomes available, the reference data products and or the pipeline may be revised, and some GO's may profit by reprocessing their data with the new reference data set or pipeline.

7.1.1 CEP produces QPOE files from telemetry tables

The task **cep** (for comprehensive event pipeline) implements the steps of wavelength -correction, aspect-correction, and makes a linearized two-dimensional spectral "image" from the event list in *table0*. It processes by reading each event as a line from the table, processing the event, and writing to a separate QPOE file for each of the detectors. **Cep** also uses aspect reports and other information from other telemetry tables and instrument data from the reference data set on the tape, or the EGODATA product from the CEA ftp site.

The parameter file for **cep**, shown in figure 7-1 is deceptively simple. This complex task uses its two parameters as pointers to a number of ASCII files which contain descriptions of the telemetry tables, the reference data, and the sub-programs **cep** uses at each stage of the reduction. **Cep** makes a number of assumptions about the names and locations of the tables, and files it needs, so it is important to have the standard directory setup described at the beginning of Part I for running the pipeline. Figure 7-2 shows a schematic of the directory structure before **cep** is run, with the locations of required tables and ASCII files. This figure should be used as the map to check your setup before running **cep**, as not all file locations are the same as when they are first read from the tape.

The user's entry point to **cep** is through the file named by the parameter `database`. A copy of the *pipeline* file which which was used to process the data at the EGO Center is located in the tarfile *Reference* with the instrument calibraiton data, and is put into the directory *reference* when the tape contents are transferred to disk. If you have not already done so, move the file to the workding directory where the QPOE files will be made. You may wish read the comments at the top of the file for additional information on **cep**'s requirments.

The contents of a typical *pipeline* file are shown in figure 7-3.

Referring back to the parameter file, we see that the only parameters are `database`, which points to the *pipeline* file and `pipeline`, which indicates where the processing script begins in the *pipeline* file. In our example, `pipeline` is set to `defpipe`. Looking at the sample pipeline file, there is a line reading

```
begin   defpipe
```

Cep will start reading the pipeline file, and will expect processing directions when it encounters this word. The name `defpipe` is just a short name for `Spectrometer Wavelength Calibration Plots`, which can be found two lines below. That is the name of the standard pipeline "script" which **cep** will follow in processing the data.

There are a few points to be noted in the pipeline script. The entries for `RefPath` and `Reference`, when concatenated together, must resolve to the correct path to the *reference* file in the reference data set. In our example, **cep** will look for the file *reference/reference*, which is the correct path for the assumed arrangement of the data on disk. Likewise, the entries `ObsPath` and `Observation` must resolve to the path to the *observation* file, in the directory containing the telemetry tables. Our example entry points to *observation/observation*, which is correct for the assumed directory setup.

Looking quickly over the sample script, one can see the general flow of the processing: events are read from the telemetry; various processing steps are done, including subpixel randomization (to remove binning distortion effects, described in section B.5.4), wavelength correction, and aspect correction; then events are written to the QPOE files. Each of these steps is represented by lower-level scripts which reside in the *reference* file. These scripts in turn point to the subroutines which are executed by **cep**.

Of course the source position on the sky is needed to perform aspect correction. The source position is specified by constructing a table called *catalog.tab* containing the RA and Dec of the source and then pointing to the table in the section of the pipeline file starting at `SourcePosition`. In figure 7-3, under `SourcePosition` the entry `Catalog` specifies the table name (which happens to be "catalog"). IRAF automatically supplies the *.tab* extension. **Cep** is instructed to look in the first row (`Column` = ROW and `Value` = 1) and get the RA from the column "RA" and the Dec from the column "DEC". The RA can be in either hours or degrees; **cep** looks at the units field of the table column to determine which is used. Using **tdump** on the copy of *catalog.tab* that was supplied on the data tape will show the coordinates and units that were used in the nominal reduction of the data.

The telemetry tables are described by the file *observation* which is kept in the same directory

```
      eu> lpar cep

database = "pipeline"      Processing text database name
pipeline = "defpipe"       Pipeline record name
   (mode = "al")
```

Figure 7-1: Parameters for the task **cep** with nominal values.

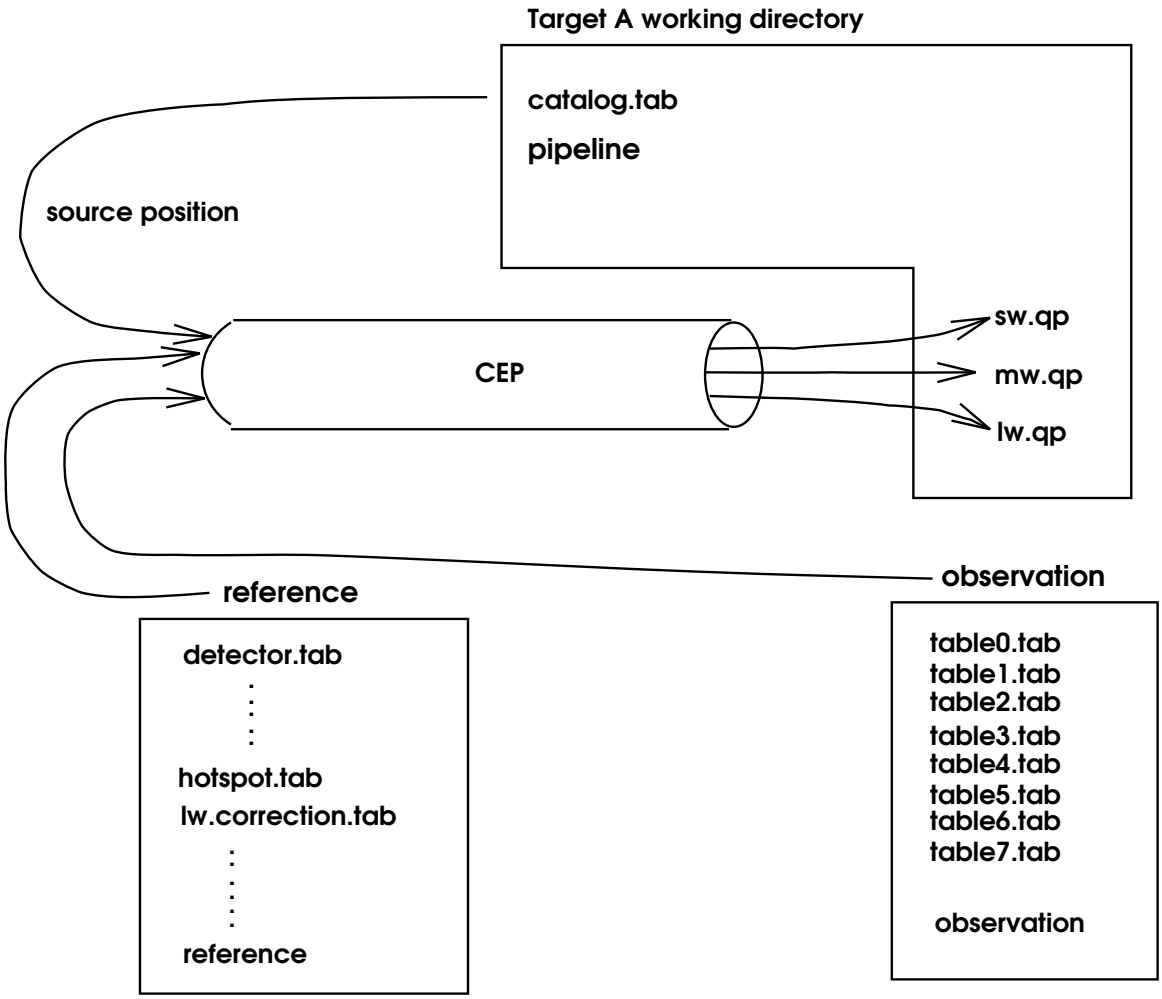

Figure 7-2: Locations of directories and files for running the event pipeline **cep**. Make sure no old versions of QPOE files are present in the working directory.

Figure 7-3: A typical *pipeline* file.

```
begin   SourcePosition
        Type    Starcat
        Catalog catalog
        Column  ROW
        Value   1
        ColumnRA        RA
        ColumnDEC       DEC

begin   defpipe
        Type    Indirect
        Name    Spectrometer Wavelength Calibration Plots

begin   Spectrometer Wavelength Calibration Plots
        Type            Pipeline
        RefPath         reference
        Reference       reference
        ObsPath         observation
        Observation     observation
        ProcessList
                Read Spectrometer Photon Events
                Convert Spectrometer WSZ to XY
                Copy Spectrometer X to S
                Copy Spectrometer Y to Z
                Randomize Spectrometer X SubPixel Position
                Randomize Spectrometer Y SubPixel Position
                Correct Spectrometer Wavelength
                Source Spectrometer Aspect
                Aspect Spectrometer Correction
                Correct Spectrometer Wavelength Aspect
                Correct Spectrometer Imaging Aspect
                Copy Spectrometer S to TYPE
                Copy Spectrometer Z to FLAT
                Copy Spectrometer X to W
                Copy Spectrometer Y to S
                Write Spectrometer Photon Events
                EOL
```

with the tables. Users should not change this file; it is created by the program that builds the tables and is not meant to be modified.

The reference data is normally kept in a directory of its own (see figure 0-5). This directory also contains a file named *reference* which **cep** uses to get information *about* the reference data. The file also contains descriptions of standard processing steps for **cep** to follow.

We will not attempt to describe pipeline scripts in any detail here. While some alternative scripts are included at the end of the reference file, it is not recommended that users attempt to write new pipeline scripts at this time. Descriptions of other available scripts and their uses is deferred to a later edition of this Guide.

Any time a new version of the reference data set is released, a new pipeline script is included. GO's should always use the script that accompanies the reference data set in reprocessing data, as it may contain changes in the processing directions. Requests and suggestions for new **cep** capabilities may be forwarded to the EGO Center[1] mail server.

7.1.2 Example: rerunning the pipeline

We will describe a situation which is expected to be fairly common: Guest Observer Dr. Ivana Ego has been informed that a new release of the reference data is available through the EGO Center ftp site at the CEA, and this new data contains important changes which the GO can take advantage of by rerunning the pipeline.

Dr. Ego installs any new software first**!** It is accompanied on the ftp site by installation instructions.

Dr. Ego is already familiar with the description of **cep** in § 7.1 and the terminology defined there. She has copied her data tables to disk in the recommended arrangement in figure 0-5. There is a "working" directory with two subdirectories: one named *observation* containing the telemetry tables and the *observation* file, and one named *Reference* containing the new reference data set. She either moves the existing QPOE files to the names **w_old.qp*, to prevent the old and new versions from being confused at a later date, or skips over them in the tape-to-disk transfer.

The most important step in running **cep** is making sure one has enough disk space available. Dr. Ego checks the size of the file *table0* in the *observation* subdirectory. She needs enough free disk space for files totaling about twice that size. Half of that space will be used in the working directory for the QPOE files, the other half for temporary files and are created in her IRAF *imdir* directory, pointed to by the environment variable *imdir*. If necessary, Dr. Ego uses the `set` command to change *imdir* to an appropriate path (don't forget the terminating "/").

Dr. Ego then copies the file *pipeline* from the *reference* subdirectory into the working directory. She uses the **tread** or **tdump** commands to look at the contents of the file `catalog.tab`, which is in the working directory, and makes sure that the values in the columns RA and DEC are correct.

At this point she is ready to run the pipeline. She types:

[1] *egoinfo@cea.berkeley.edu*

```
eu> cep pipeline defpipe
```

to start. **Cep** prints a message to the screen as it finishes processing each 10% of the telemetry. From this Dr. Ego can estimate how long it will take to finish running. **Cep** will usually take up to several hours to run for a typical observation. Temporary files continue to appear from time to time in her *imdir* directory as the task runs. When **cep** finishes, it deletes all the temporary files and Dr. Ego has three finished QPOE files named *sw.qp*, *mw.qp*, and *lw.qp*, in her working directory.

7.2 Special Problems in Aspect Correction

The aspect correction algorithm in the EGO Center processing pipeline **cep** uses the satellite aspect reports from telemetry to determine the instrument pointing direction, or attitude. The attitude is then compared to the source position and correction factors are calculated that should restore each event to its nominal boresight position on the detector. If a stationary target's position is known with enough accuracy, the main limit on the spectrum focus is the quality of the the aspect reported in the telemetry.

Targets for EUV spectral observations, however, include some objects that move too rapidly through the field of view the field of view for the satellite to keep them on boresight for extended periods. These include planets, such as Mars and Jupiter, comets, and the moon. Although observing conditions for these targets are being improved by more frequent slewing, most observations of such objects from Cycle I will have the source moving in the field of view. The result is that spectra are smeared out in both the imaging and spectral directions on each detector, creating fuzzy images and spectral offsets that change with time.

Besides this, stationary targets are sometimes subject to problems which cause inaccurate aspect reporting, with similar results. These problems include pointing with an insufficient number of guide stars (< 2), interference in the star trackers from sources near guide stars, and flexing of the telescope support structure, which is not detected by the satellite attitude system. These conditions can produce blurring in both spectral and imaging angle directions, and spectral shifts of several Å, which remain even after **cep** has performed its aspect correction.

7.2.1 ASPCORR creates new aspect tables from source positions

While nothing can be done to correct aspect reporting after the fact, both types of error can be addressed in the pipeline using the task **aspcorr** if some reliable source of information on the target's position in the field of view (hereafter called the 'apparent coordinates') is available. **Aspcorr** uses the satellite aspect data, a fixed target position, and a table with the apparent celestial position of the target at regular intervals, and produces a new table of aspect readings. Each line in the new aspect table makes a "correction" to the satellite aspect equal and opposite to the off-axis pointing angle, which, when input to **cep** with the nominal position, produces a corrected QPOE image file.

In effect, the new aspect table fools **cep** into compensating for the aspect errors, with the result that the photon positions are corrected more accurately, and the resolution of the spectrum is improved. However, like any other aspect correction, this process relies on the precision with

```
      eu> lpar aspcorr

position = "moon.tab"      apparent positions table
asptable = "table1"        satellite aspect table
  output = "new_table1"    output table
   alpha = 0.              fixed position ra(hours)
   delta = 0.              fixed position dec(degrees)
     (ra = "RA")           ra column name
    (dec = "Dec")          dec column name
   (time = "Time")         time column name
 (aspect = "AspectT,AspectW,AspectX,AspectY,AspectZ") Time,W,X,Y,Z
   (mode = "al")
```

Figure 7-4: Sample parameter file for running **aspcorr** on observed lunar positions.

which the user knows the location of the source. Calculating the apparent coordinates of the source is done in one of two ways, depending on the type of error to be corrected:

1. For moving objects where the aspect reports are accurate, but the telescope was not pointed directly at the target, the apparent target RA and Dec in the field of view are calculated from the satellite aspect and ephemeris in *table1*, and known target positions.
2. In cases where the aspect reporting is poor, if the source has sufficient flux in the Lexan bandpass, centroids can be obtained with some frequency from the Deep Survey data. Centroids and photon arrival times can be made available to GO's for their observations by the EUVE Data Analysis and Support System (DASS) upon request, but they are not among the standard data products supplied by the EGO Center.

Once the corrections are obtained, from whatever position information, the procedure is the same: pack the target positions into a standard format ST table using a task like **tedit** or **tcreate**, run **aspcorr** to create the new aspect table, substitute it for *table1*, and rerun **cep** with the corrected aspect table.

A typical parameter file for **aspcorr** is shown in figure 7-4.

The parameter `position` should point to a three-column ST table with the celestial coordinates of the object at frequent intervals, of 1 second or more. The next three parameters allow the user to specify the column names used in the position table, which in this example, happen to be called 'RA', 'Dec', and 'Time'. The times associated with the calculated apparent positions must span the observing time, and include times both prior and posterior to the exposure.The aspect is reported every 1.024 seconds.

The original satellite aspect table, and the corresponding column names are supplied in `asptable` and `aspect`. The default values are the standard table and column names used on all GO data tapes. After the name of the output table, a fixed source position in celestial coordinates is supplied in `alpha` (RA, Hrs:min) and `delta` (Dec, dec.degrees). Note that the command line parameters are `position`, `asptable, output, alpha`, and `delta`. Thus, the task could be run in the *observation* directory using the default hidden parameter values as follows:

```
eu> aspcorr moon.tab table1 new_table1 0.0 0.0
```

This produces a new aspect table, with the same column names for time and the four components of the aspect quaternion that were used in the `asptable`. Since aspect is the only information from *table1* that is used by **cep**, the new aspect table can be substituted by moving *table1* to a new filename, and renaming *new_table1.tab* as *table1.tab*. Then **cep** must be rerun with the new aspect table to produce QPOE files with more accurate event locations.

One can also use **aspcorr** with any record of the target's apparent position as a function of time to create an aspect table that compensates for poor aspect reporting. Knowledge of the Deep Survey boresight position can be used to calculate the sources apparent position.

7.2.2 Example: rerunning cep on an observation of the moon

A 1600-second test observation in which the full moon scanned through the DS/S field of view has been reduced by this method, using apparent coordinates for the moon calculated by Dr. Igor Ego (AKA Dr. Randy Gladstone, of Austin Research Institute). The detector image in the LW spectrometer before correction, is shown in figure 7-5. The accumulated image of the uncorrected data looks like a number of stripes, nearly perpendicular to the spectral direction.

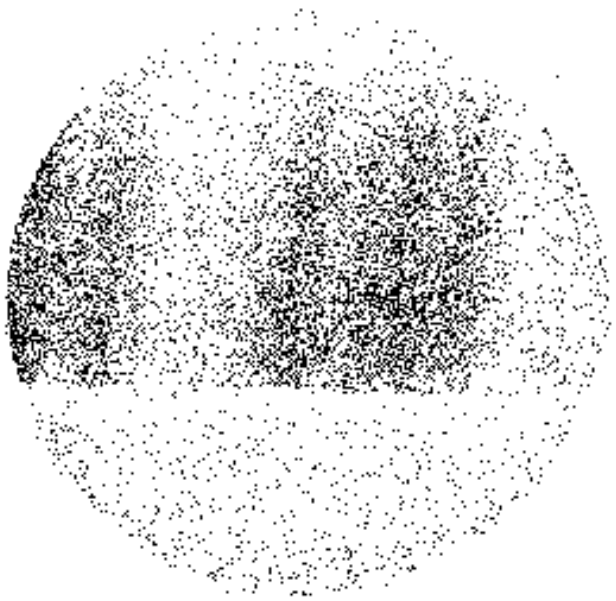

Figure 7-5: Image of the moon in the long wavelength detector, after processing with the satellite aspect table and the standard pipeline, but before processing with **aspcorr**. The streaks show the paths of spectral features across the detector during the observation.

The spectrum is actually composed of overlapping images of the full lunar disk at a number of wavelengths. Each image is a single spectral feature, its width determined by the 20′ collimator transmission. The moon is obviously very bright (remember this is only 1600 seconds of data), but the spectrum is badly spread out in both imaging and spectral directions. However, Dr. Ego is not discouraged. He writes a program which uses the satellite ephemeris from *table1* to calculate the apparent RA and Dec of the lunar disk every 2.048 seconds, and packs the output into an ST table. The first few lines of *moon.tab*, containing Dr. Ego's apparent celestial coordinates of the moon, are shown in figure 7-6.

Column	1	2	3	4
Label	Index	_____Time_____	______RA_____	______DEC_____
1	0	774678132.7360001	81.0991821	23.7006283
2	1	774678134.7840004	81.0980301	23.7021084
3	2	774678136.8320002	81.0968246	23.7035847
4	3	774678138.8800002	81.0956726	23.7050686
5	4	774678140.9280001	81.0944824	23.7065468
6	5	774678142.9760003	81.0933304	23.7079926
7	6	774678145.0240002	81.0921249	23.7094669
8	7	774678147.072	81.0909271	23.7109451
9	8	774678149.1200001	81.0897369	23.712431
10	9	774678151.1680003	81.0885468	23.7138958
11	10	774678153.2160001	81.0873489	23.7153378

Figure 7-6: ST table input to **aspcorr** with lunar positions for every other major frame

He then runs **aspcorr** using the parameter settings shown in figure 7-4 to produce the ST table *new_table1.tab*. The new table contains aspect quaternia, a few lines of which are shown in figure 7-7.

Dr. Ego then moves *observation/table1.tab* to *observation/table1old.tab* and copies the new aspect table into *table1.tab*. Since the pipeline uses only the aspect and time columns of *table1*, this change will be transparent to **cep**. He then returns to the working directory and carefully *renames his original QPOE files* before setting up and running **cep** as described in section 7.1.2. This creates a new set of QPOE files with the generic names (*sw.qp, mw.qp, lw.qp*) used by **cep**.

The image created from the new LW QPOE is shown in figure 7-8.

The corrected picture shows each spectral feature centered in y at the position of a boresight spectrum. The height of the image is determined by the size of the moon in the field of view.

The aspect correction for a distant source with a bad aspect table can be improved by virtually the same method. Any record of the object position with greater accuracy than the aspect report can be converted to effective coordinates in RA and Dec, and used to calculate the corrected aspect table with **aspcorr** before rerunning the pipeline. Positions may have to be interpolated to achieve the required frequency, depending on how often a good centroid can be obtained.

Column	1	2
Label	_____AspectT_____	_____AspectW_____
1	774678133.1390079	0.6644102135026706
2	774678134.163008	0.6644057080561128
3	774678134.7840004	0.6644029725503054
4	774678135.187008	0.6644011547295496
5	774678136.2110082	0.6643962913513004
6	774678136.8320002	0.6643933395028272
7	774678137.2350078	0.6643914734961072
8	774678138.2590079	0.664386962646536
9	774678138.8800002	0.6643842227585436
10	774678139.2830078	0.6643824170016941

	3	4
Label	_____AspectX_____	_____AspectY_____
1	-0.3740989780195041	-0.5232151820760572
2	-0.3741017126098966	-0.5232145074152104
3	-0.3741033700186317	-0.5232139558862321
4	-0.3741044181743463	-0.5232136479903312
5	-0.3741070834119812	-0.5232131065275788
6	-0.3741088446477844	-0.5232129243487749
7	-0.3741100191197195	-0.5232127482948769
8	-0.3741125250967082	-0.523212066339116
9	-0.3741140449760684	-0.5232116542559422
10	-0.3741150087346472	-0.5232114243782759

Column	5
Label	_____AspectZ_____
1	0.3805978845892577
2	0.3806041354529378
3	0.3806080750180024
4	0.3806106641431434
5	0.3806170105491646
6	0.3806210076622806
7	0.3806235635017167
8	0.380629823501779
9	0.3806334790965606
10	0.3806358702917326

Figure 7-7: Columns of an output table from **aspcorr**.

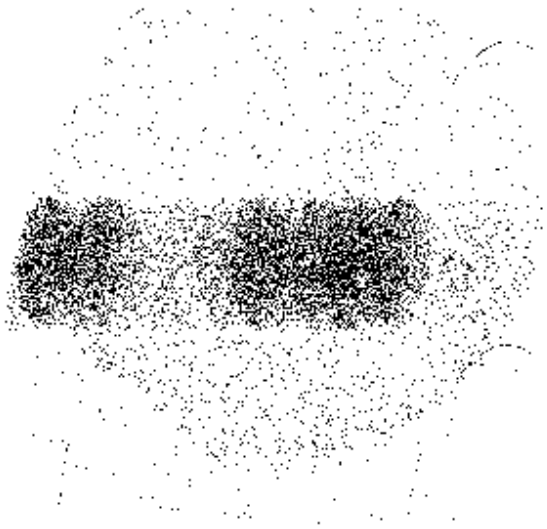

Figure 7-8: Lunar observation after processing with the standard pipeline and corrected aspect table created by **aspcorr**. Each oblong image of the lunar disk is a spectral line whose width is determined by the spectrometer optics. The height of the spectrum shows the size of the moon in the Spectrometer's 2-degree field of view.