USING TELEMETRY IN ST TABLES

1.1 Data Quality Selection Using EUVE Telemetry

The first part of this chapter is about using the information in your telemetry tables, (mostly *table1.tab*) to create data quality selection filters. This method of data selection is based on the event times recorded in both the QPOE files and the telemetry tables. When a data quality selection filter is applied to the QPOE file, the time tag on each event is compared to a list of "good time" intervals, made from the telemetry tables, to determine if that event should be included in subsequent processing. The 2-d spectral images on each GO data tape are the result of this type of filtering; each event's time was compared to a time filter to determine whether it would be in the binned image. The criteria used in nominal processing are not especially stringent; every event from periods when the detector was operating at a normal voltage and the source was visible are included. This is a filtering known as choosing all events from "valid times"–the most basic criterion for usable data.

Some Guest Observers want to further restrict this choice of events, to eliminate periods of high background, at orbit dawn or dusk, or when the satellite crosses the South Atlantic Anomaly (SAA). Each GO's priorities may be different; where one investigator might accept a small loss of source data to omit periods of high background, another might wish to retain source counts at some cost to S/N.

In these cases, one must reselect the events from the QPOE file, using a new selection rule that passes only the events which meet the GO's criteria. This chapter should help GO's create filters in the form of tables of "good times" that select the best data set for their science. We will see how to pass those filters to the QPOE interface and create new images from the QPOE file in Chapter 2.

GO's who wish to use the 2-d spectral images created by the EGO Center nominal data reduction process without further filtering can proceed to Chapter 3, to determine whether they wish to re-extract their one-dimensional spectra.

1.1.1 DQSELECT creates and installs QPOE filters

The task **euvred.dqselect** is an interactive tool which creates plots of the data in ST table columns vs. time, constructs filter strings defining good time intervals based on the data, and writes the string in a QPOE file or table. The time-plots are like strip charts of each column, on which the user can set and mark limits on the values. Internal task commands create tables of valid times based on when the data transit those limits, and install the start and end times in a QPOE file as a data selection filter. We will describe using **dqselect** to set limits on the various columns of EUVE telemetry tables, which correspond to different monitors of the instruments and spacecraft, and constructing time filters which delimit intervals when the monitor values were "good" [1].

Please read the **dqselect** help page for a detailed description of the operation of the task. Refer to Appendix A of the *Data Products Guide* for definitions of the data in the telemetry tables and indications of their usefulness to GO's.

Do I need to use dqselect?

dqselect is made to facilitate data quality filtering, which actually takes place in the process of creating an IRAF spectral image from the QPOE file. All EUVE QPOE's have had a minimal data quality filter, the valid times (exposure) filter, applied when they are binned to make the 2-dimensional images, *sw.imh, mw.imh, lw.imh,* and *ds.imh.* GO's who wish to add criteria of their own to the definition of usable data will probably want to use **dqselect** to translate those criteria into good time intervals, or some other sort of event attribute filter.

Since the exposure filters are also installed in the QPOE files as macros (a process explained below), there is usually no need to duplicate creation of the default exposure filter. Even if the GO has re-run the event pipeline task **cep** to produce new QPOE files, the same exposure filters can often be reinstalled, from tables supplied on the tape. New exposure filters must be made only if the reprocessing changes the good time intervals, e.g. by subdividing or summing observations.

A sample parameter file for running **dqselect** is shown in figure 1-1.

```
eu> lpar dqselect


 tables = "observation/table1.tab" Monitor Tables
(limits = "")              Limits Tables
   (gap = 1)               Time Gap
 (imode = "line")          Input mode
(format = "13.3f")         Interval format control
(device = "stdgraph")      Graphics device
  (mode = "al")
```

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

[1] The term "good times" will always be used with the to imply a process of selecting the best data for the particular purpose at hand, while the word "valid" is used for the minimal data quality selection performed in the EGO Center's nominal reduction.

Preparing to run **dqselect**

- `tables`: the tables to be loaded into the task's buffer. The value in the example will load only *table1.tab*, which has all the most frequently used EUVE monitors.
- `imode`: the task can process input in either "line" or "cursor" mode. In line mode (the default), users enter commands to a command line cursor, choosing from a displayed menu. The alternative is the normal IRAF cursor mode; all line mode commands are still accessible as colon commands, as well as many commands that use cursor reads as input.
- `limits`: parameter used to load the name of a table of pre-defined limits, such as limits saved from a previous session with **dqselect**.
- `gap`: The parameter `gap` defines the minimum number of consecutive readings (lines) that must be missing from the table before **dqselect** considers the data stream interrupted. Gaps are shown in the display as breaks in the curve of displayed values. Make `gap` equal to 1 on the first viewing, so that **dqselect** notices all the gaps, major or minor. See the subsection "Gaps in the data" below, for an explanation of gaps and their consequences.

The next few sections describe the main functions of **dqselect**, in the order they are normally used. Most **dqselect** sessions cycle through four steps: `display` the monitors of interest, `edit` each monitor to set the desired upper and lower limits, run `gtgen` to generate the start and end times for the resulting good time intervals, and install the good times as a `filter` in the QPOE file. See Example 1.1 for a more detailed example, showing the **dqselect** display.

To begin running **dqselect** from the command line, type:

```
eu> dq observation/table1.tab
```

Monitor display

Once **dqselect** has been started with a table loaded in memory, the command:

```
Command> display monitor
```

produces a plot of the named `monitor` against time on the `STDGRAPH` device. The argument is case insensitive; you don't have to know whether the column name has any capitalization. The plot's time axis is numbered relative to the first time-tag in the loaded tables. Up to four monitors can be displayed on `STDGRAPH` at once. After that, one must remove a monitor from the display by using the command `hide` before a new monitor can be plotted. Hidden monitors can be resummoned to `STDGRAPH` later in the session by reissuing the `display` command with three or fewer monitors in the window.

Display sampling of ST tables

Due to the limitations on most STDGRAPH devices, **dqselect** must sample any table longer than about 500 lines. This means that most EUVE telemetry tables, which typically span tens of thousands of seconds, will be heavily sampled for display in **dqselect**. Sampled display means that high-frequency variations in the data may not be visible in **dqselect**, but this not usually problematic unless one is setting very narrow limits on the detector count rates (see section 1.1.2 below). See Appendix B if monitor plots at full resolution are required.

Since all EUVE observation data are selected from only nighttime telemetry, when the satellite is in the earth's shadow, the daytime parts of each orbit also appear as large gaps in the monitor displays. This is not a result of the the sampling, and is completely normal.

Setting limits

Limits can be set on each monitor displayed, establishing the upper and lower bounds of an acceptable range of monitor values. The capability to set limits on several monitors allows the user to apply several criteria at once. Use the command:

```
Command> edit monname
```

to set or edit the limits on the monitor `monname`. Issuing the command causes **dqselect** to enter cursor mode, and print a prompt:

```
Please pick the first limit value ...
```

Limit values can then be entered as cursor reads by placing the cursor and entering any keystroke[2], or typed in as colon commands with the syntax:

```
:y limit.decimal
```

```
and again...
```

```
:y limit.decimal
```

The limits can also be entered as a combination of colon commands and cursor reads. The order does not matter; the lower number will be treated as the lower limit, regardless of whether it was entered first or second.

Once limits are set, they appear on the monitor display as horizontal bars on the

[2]or by reading the cursor position from mouse clicks.

vertical axis. All the current limits can be saved as an ST table at any time, by using the command:

```
Command> limsave mylimits
```

to create the table *mylimits.tab* in the current working directory. Tables of saved limits can be reloaded into **dqselect** for a later session.

Defining good 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 the **dqselect** command:

```
Command> gtgen
```

This command causes **dqselect** to find all the times when each monitor transits one of its limits. Each time when a monitor are set enters the range prescribed by its limits becomes a "start time," and each time it goes above or below the range becomes an "end time." Each start time/end time pair makes a good time interval. All the sets of good times for all monitors with limits are intersected, producing the set of intervals when *all* the monitors are within their limits. If the tables are long, or many monitors are involved, `gtgen` may take a while to run.

The sampling that is implemented to display monitors does not affect the generation of good times. This means the list of intervals made by `gtgen` may show breaks in the valid time intervals not visible in the graphic display. In these cases it is the good time intervals that are accurate, not the display.

Saving the results of gtgen

The start and end times calculated with `gtgen` are stored in **dqselect**'s memory, but if they are to be used for selecting events from a QPOE file, they must be saved somehow before quitting from **dqselect**. **dqselect** can write the start and end times to an ST table, and can also format them into a filter string and write the string as a macro in a QPOE file.

The command `filter` is used to convert the list of start and end times into a standard QPOE I/O interface format, write this filter string in a named QPOE file, and give the filter string a name, making it a macro:

```
Command> filter QPOE_name macro_name
```

The start and end times can also be saved in a table in the current working directory called *mygtimes.tab* using the command:

```
Command> gtsave mygtimes
```

Saving the good times to a table is *always* a good idea; they don't usually take up much space, and it's often handy to have them around later for variations or fixes to your reduction. For example, if the user wants to intersect separate sets of good times using the task **euvtools.tfilter**, as in section 2.2.3, the good times must be present in ST tables like those made by `gtsave`.

Repeated use of gtgen

An important point to remember is that whenever `gtgen` is run in **dqselect**, *any set of good times that already exists in the internal buffer will be intersected with the newly computed good times.* **dqselect** always starts up with an empty good times buffer, but if you issue the `gtgen` command more than once before quitting, the good times buffer should usually be cleared each time by issuing the command:

```
Command> gtnew
```

Do not use the `gtnew` command until you have saved any useful good times as a table or a QPOE macro.

Likewise, limits which are not meant to be re-used when `gtgen` is re-issued must be removed with the command `delete`, prior to re-running `gtgen`. This must be done even though the monitors may not be visible on the screen, having been displaced by those displayed more recently. The limits still exist, and will still be used to generate start and end times. For this reason, it is a good idea to keep a written log of each session with **dqselect**, recording all major commands so that you can keep track of what limits and good times exist in memory.

The `gtgen` command merges time intervals by default, because this feature can be very useful. If the user wants to add a new criterion for good data to those used for a previous set of good times, they need not recreate the entire process. The command `gtload` can be used to load a previously saved table of the old good times into the buffer, after **dqselect** has been started with the data table containing the monitor to be "added." The user then displays and edits limits on the new monitor, and uses `gtgen` to intersect the good times for the new monitor with the old ones.

Gaps in the data

The frequency of readings in EUVE telemetry is predetermined, but inevitably there are some gaps, or periods when there is no data, for each monitor's history. Lines may be missing from the table altogether, or there may be INDEF values in some of the columns. During these gaps, the values of the monitors are unknown, but gaps do not necessarily indicate any malfunction or change in the monitor; they are just lapses in reporting. Gaps between the time stamps of successive lines in the table occur for two main reasons:

1. First of all, the data included in EUVE GO data sets is selected from nighttime telemetry only, to obtain the best signal-to-noise. This means the data tables are composed of continuous records for each orbit night $\approx 1800 - 2000$ seconds), when the satellite is in the earth's shadow, interspersed with gaps the length of the orbit day (≈ 3600 seconds).
2. In addition, one or several values of any monitor may be missing because of telemetry transmission problems. The problem may be a single even upset (SEU) in the instrument or satellite electronics, or corruption of the telemetry at one of several stages in transmission to the ESOC at Berkeley.

The user must tell **dqselect** how long a data gap can be (in units of the reporting interval) before it is considered significant, by setting the `gap` parameter in the parameter file or using the command `gap` after starting **dqselect**. When **dqselect** is constructing the good times for a monitor, it compares the interval between the time tags of each pair of monitor values to `gap`. When the interval between successive times is greater than (`gap` × report interval), a good time interval is ended, or not begun, depending on what state **dqselect** is in at that time. This means that a small value of `gap` can affect the average length of good time intervals, even though the gaps may not appear as breaks in the display, because of sampling by the graphics display device.

As an example of how the value of `gap` interacts with `gtgen`, lets assume limits of 3–60 have been placed on the monitor `Det4ADct` (the ADC counts for the SW spectrometer) and that `gap` is set to 5. This monitor is in `table1.tab`, for which the reporting interval is one major frame, or every 1.024 seconds. As it is creating good times, **dqselect** encounters a line where the monitor jumps from a value of 80 to 55. Since no good time interval has been started, **dqselect** considers the time of the row containing the value 55 as the start of a good time interval. The monitor now stays within limits 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 ≈ 3600 seconds is encountered (the spacecraft entered daytime). This exceeds the value of `gap`, so **dqselect** terminates the current good time interval with the time of the last reported value before the gap.

Related topics

◇ The process of creating the nominal time filter used in EGO Center processing and applying it to the QPOE is described in Chapter 4 of the *EGO Center Data Products Guide*, "EGO Center Data Processing."

Example 1.1 : Creating the standard exposure filters

Each EUVE QPOE file is delivered with standard time filters, representing the basic "valid" exposure time for the observation[3]. The filters select times when the detector was turned on, the count rate was within operational limits, and when the earth was not in the field of view. The exposure filter is slightly different for each detector. This example describes the generation of the "detector on" part of the exposure filter, based on the ADC count rates for each of the DS/S detectors. The steps of making the filters are presented here to illustrate the use of **dqselect** and to give the user a basic recipe that can be modified to create other filters.

The detector-on filters are based on the monitor `Det#ADct` in *table1*. The filter for earth blockage is based on another monitor, `DSSZen` in *backmon*. Example 1.2 illustrates creation of an earth blockage filter. After starting IRAF, the GO first moves to the target directory, directly above the *observation* directory with the telemetry tables:

```
eu> chdir theobject
```

She then starts an interactive session with **dqselect**, loading *table1.tab* into memory, and setting the output format to print three digits after the decimal point:

```
eu> dqselect observation/table1 gap=1 format=13.3f
```

She displays plots of each Spectrometer detector count rate monitor on `STDGRAPH`:

```
Command> display det4adct

Command> display det5adct

Command> display det6adct
```

Figure 1-2 shows a display of the three Spectrometer count rate monitors for an average-length observation.

Limits representing valid operating conditions must be set on each count rate, and a separate set of good times generated for each detector. The value of each ADC count rate should be above 2 counts/sec, to exclude turned-off intervals with 1–2 spurious counts, and below about 5000, to include all times when the detector was turned on. The GO sets the detector count rate limits for the SW

[3] For a description of these filters, see the *EGO Center Data Products Guide*, version 1.5, section 4.3.

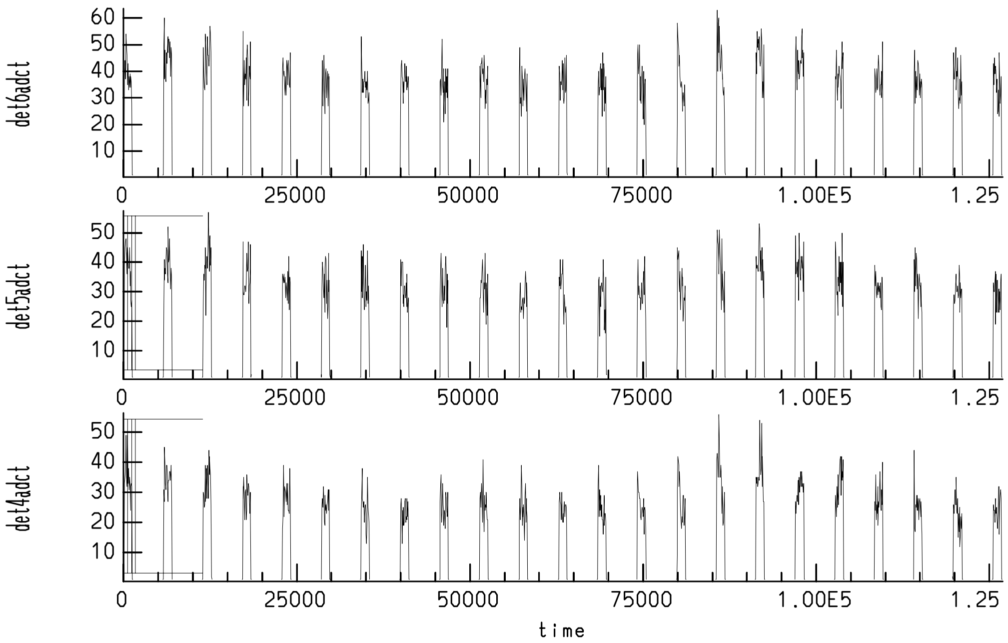

Figure 1-2: Spectrometer detector count rates for a 125 ksec observation, displayed in **dqselect**. The display can show up to four monitors from the loaded table(s) at once. The horizontal bars near the Y-axes of the bottom two plots show user-defined limits.

Spectrometer with the `edit` command:

```
Command>  edit det4adct

 Please pick the first limit value ...


 :y 3

 and again:

 :y 5000
```

Horizontal limit markers should appear on the vertical axis of the `Det4ADct` plot[4]. Now the GO generates the set of good time intervals associated with the short

[4]The markers will not be visible if limits are off the plot scale.

wavelength count rate:

```
Command> gtgen

 Generating good times for monitor, det4adct.
```

She places a filter string with the good time intervals in the QPOE file, *sw.qp*:

```
Command> filter sw sw_adct_gt

 Creating the filter string.
```

dqselect writes the filter string in the QPOE file header as the definition of the macro `sw_adct_gt`. The GO also saves the ADC count good times in a table, for possible future use:

```
Command> gtsave sw_adct_gt

 Good times saved to file, sw_adct_gt.
```

The GO repeats this process for the medium, long, and DS detectors. Before setting the same count rate limits on the medium wavelength ADC count rate, she removes the limits from the `Det4ADct` monitor:

```
Command> delete det4adct
```

and clears the SW detector good times from memory with the command:

```
Command> gtnew

 Ready for generation of new good times.
```

Now she is ready to edit the limits on `Det5ADct`, generate the MW count rate filter, and install it in *mw.qp*. The processing messages from **dqselect** always tells the user which monitors are being used to create the intervals.

```
Command> gtgen

 Generating good times for monitor, det5adct.
```

```
Command> filter mw mw_adct_gt

 Creating the filter string.


Command> gtsave mw_adct_gt

 Good times saved to file, mw_adct_gt.
```

And so forth, for the LW and DS detectors as well, saving each set of good times in an appropriately named ST table. After finishing the LW macro and quitting from **dqselect**, the GO uses the task **euvtools.qpmedit** (discussed in section 2.2.2) to list the macros in the QPOE files. An abbreviated version of the display generated from the SW QPOE file looks like this:

```
sw,time = d0
sw,x = s8
sw,y = s10
sw,lambda = r12
sw,theta = r16
sw,dx = s20
sw,dy = s22
sw,sw_adct_gt = 780800001.460:780802014.644,78080552.939:780807708.084,78081121
9.380:780813402.548,780816913.844:780819097.012,780822607.283:780824790.451,7808
28301.747:780829478.323,780829480.371:780830484.915...:780924686.475
```

with each pair of colon-separated numbers representing one orbit night. Each time in the macro `sw_adct_gt` is rounded to three decimal places. The accuracy of times in the telemetry is about 1 – 8 milliseconds, so the other digits in the *table1* times are not significant. **dqselect** truncated the times according to the parameter `format`, which was set to 13.3f. Also, the typical interval between two colon-separated times is roughly 2000 seconds. This is a normal length for a nighttime pass, so the only times that are being excluded are orbit days.

This example shows only the first stage of making the valid time filters. During EGO Center batch processing, both *table1.tab* and the table *backmon.tab* are loaded into **dqselect**. After the ADC valid times are found, the times free from earth blockage are computed from data in *backmon.tab* and merged with the ADC times to create the exposure filters called *<det>_gt* in each QPOE.

1.1.2 Count rate limits and signal-to-noise: the long table of very short time intervals

Some GO's may wish to use a more stringent definition of valid times for their observations. For example, a GO who notices a marked increase or fluctuation in the count rate of a non-varying source might infer that the background due to airglow was changing during the observation. If one wishes to remove those periods with the highest background levels, one can reformulate the valid times with a decreased upper limit on the ADC count rate. Any new spectral image created with the "pickier" filter will contain a new corrected exposure time. This strategy can increase signal-to-noise, at the cost of losing a few source counts, and can decrease or eliminate the need for primbsching/deadtime corrections in observations that are affected by primbsching for only brief intervals. However, over-tightening the count rate limits can have undesirable effects, whose origins may not be clear in **dqselect**.

Figure 1-3 shows a **dqselect** display of an SW detector count rate for a short observation, including two orbit nights and one day. The good times produced for this observation by the standard limits have only two intervals: one for each nighttime pass. Now in the figure, limits have been set at 20 and 40, which appear to exclude periods with higher count rates at the beginning of each nighttime pass. By contrast, the list of good times generated from these limits has 316 intervals, varying in length from a few to a few tens of seconds.

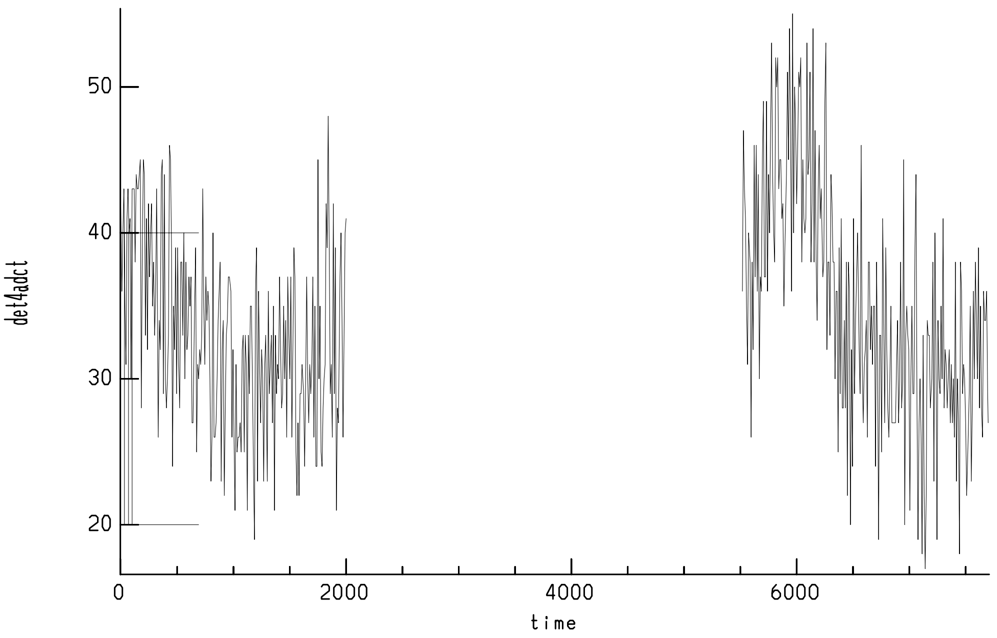

Figure 1-3: Spectrometer detector count rate for a short observation, displayed in **dqselect**.

If this result is multiplied over the 25 – 40 orbits of an average EUVE observation, the number of intervals could be somewhere between 3000 and 7000, and the number of characters in the macro could be from 84,000 to nearly 200,000. This is cumbersome at best, and doesn't really improve the signal to noise. Also, there are limits both on the length of QPOE macros and on the number of intervals they can specify. With the **euv** package installed the limit on the length of a single QPOE macro is 65536 characters[5].

The GO may be puzzled at the number of good times generated with the new limits, until recalling that the display of most monitors in **dqselect** is heavily sampled. This prevents the user from seeing just how often the ADC count rate transits the limits they set, or indeed from seeing whether fluctuations visible in the display are really significant within the actual range of data values.

Users who wish to see every line of the table in their plot can use the **igi** (for **interactive graphics interface**) in the package **tables.tbplot**. Its use is described briefly in Appendix B; please note the instructions on dealing with the double precision time stamps in EUVE tables **igi** can be very helpful for setting count rate limits and finding the exact times of outbursts and other transient effects.

Figure 1-4a) shows the 2-orbit **dqselect** display of the SW ADC count rate monitor, zoomed in X to show the spacing of the sampled monitor readings for about 200 seconds. The vertical scale is from 16.5 to 55.3 Figure 1-4b) shows the same SW ADC count monitor displayed in **igi** with about the same vertical scale. One can see from the **igi** display that transits of the value 40 cps are quite frequent, and that

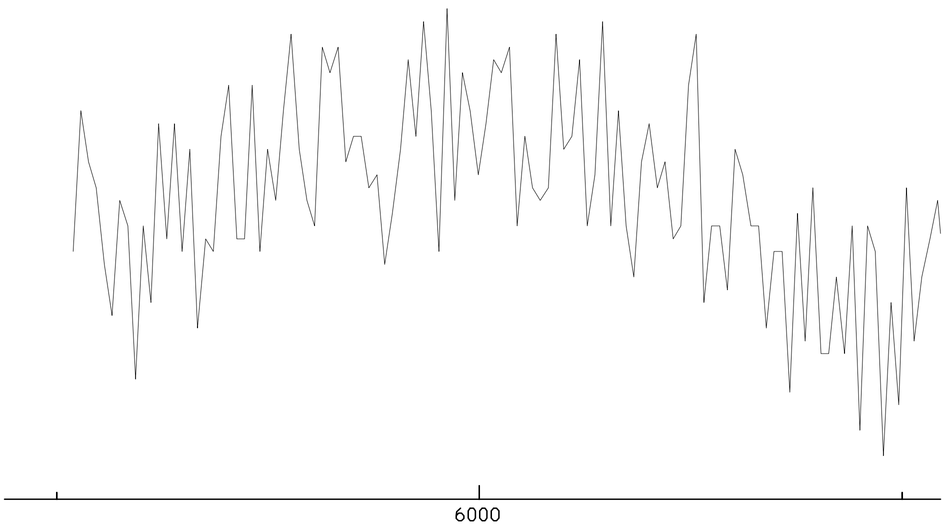

Figure 1-4: a) Detector count rate enlarged in **dqselect**. Plot spans approximately 200 seconds with a vertical range of 16.5 – 55.3.

[5] Otherwise, the limit is usually 8192 characters.

the range of values present is larger than in the sampled **dqselect** display. The moral of the story is that placing very tight limits on detector count rates in **dqselect** may NOT significantly increase signal-to-noise, may make for unwieldy exposure filters, and may exclude a large portion of the valid exposure time.

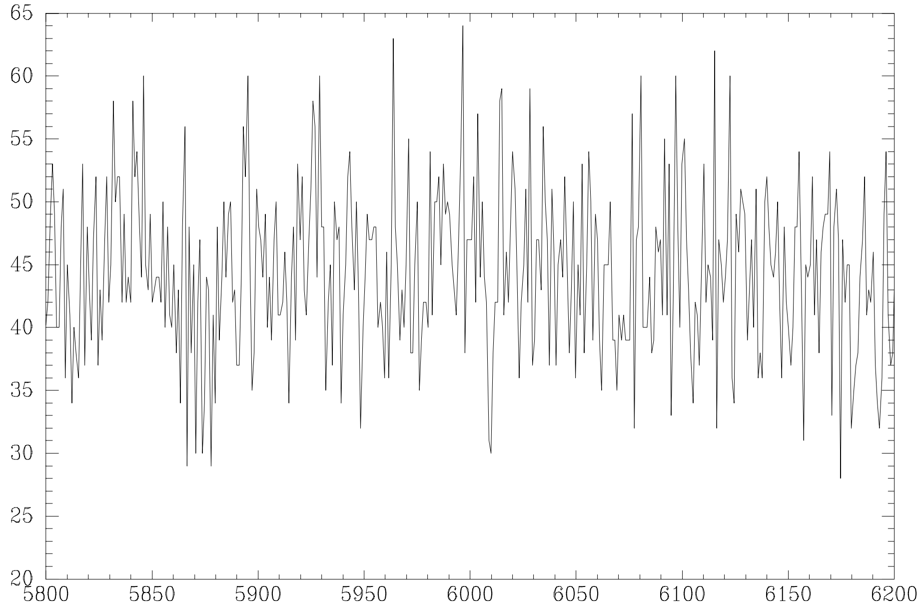

Figure 1-4: b) Display of detector count rate in **igi**. **igi** reads every line of the selected table column, without sampling. The window has been "Zoomed" to show the fluctuations due to Poisson noise.

Related topics

◇ If for some other reason you want to create a longer QPOE macro than is allowed by your system, there are ways to work around the size limitations. See Appendix A, section A.4.1 for more information on setting up your IRAF environment to deal with long QPOE macros.

1.2 Specialized Data Quality Selection Tasks

This section is about several specialized tasks that address particular problems that may occur in some data sets. All these tasks create new ST tables from existing ones, usually with the purpose of rendering the telemetry data into a more usable form. The tables are used for data selection, and correction of exposure times for effects called "primbsching" and TIF (Telescope InterFace) deadtime. GO's can refer to Chapters 4 and 5 of the *EGO Center Data Products Guide* for more background on the problems these tasks address. Here, we've tried to include material to help users determine whether and when they need to use these tasks.

1.2.1 MKPRIMBSCH makes deadtime correction tables

The task **euvtools.mkprimbsch** computes corrections for primbsching and TIF deadtime which can 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 section 5.1 of the *EGO Center Data Products Guide*. **mkprimbsch** supplies the capability for investigating the severity of the problem, and applying corrections in the process of rebinning a QPOE file into a two-dimensional spectral image.

mkprimbsch reads information from the raw telemetry tables and from a reference table of TIF deadtime calibrations, and computes the amount of deadtime and primbsching which occurred during each pair of telemetry frames (2.048 seconds). The computations are based on count rates reported by the detectors for each filter quadrant before TIF processing, and the numbers of events actually recorded in *table0.tab*. From these, it produces a table containing columns of reference data, and multiplicative correction factors for each of the detectors during each two-frame interval. Once the primbsching/deadtime correction table is produced, one can use the results to determine the magnitude of the correction required, and to selectively exclude frames with corrections above some limit, or create spectral images with corrected exposure times, using the task **qpmkim** (section 2.3).

Do I need to run mkprimbsch?

mkprimbsch is run for each observation as part of the routine EGO Center data reduction, starting with version **1.4** of the IRAF/EUV software package. The output table *primbsch.tab* is supplied on the standard data tape, and it should not be necessary for most GO's to run **mkprimbsch**. Beginning with data processed under version **1.5** of the IRAF/EUV package, the standard two-dimensional spectral images supplied on each tape have recorded exposure times that have been corrected using the table *primbsch.tab*. If new images with specialized filters are desired, they can be produced with primbsch/deadtime corrected exposures from any QPOE file for which there is a primbsch table, by using **qpmkim**. Even if the QPOE files are remade by rerunning the pipeline, the event times, and thus *primbsch.tab*, are not changed. Primbsching/deadtime correction can be applied to the new QPOE's in the process of creating new spectral images with **qpmkim**. See section 2.3 for more detail on making corrected spectral images.

The parameters for **mkprimbsch** are shown in figure 1-5. The task requires information from several sources; the parameters given here will produce a correction table for all three Spectrometer detectors and the Deep Survey detector, assuming the nominal arrangement of the data on disk (shown in figure 0-3 of the *Data Products Guide*). See the help page for a detailed description of columns in the output table.

```
eu> lpar mkprimbsch
```

```
      output = "primbsch"      output table name
  (detectors = "sw,mw,lw,ds")  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 1-5: Sample parameter file for the task **mkprimbsch**.

Notes on running **mkprimbsch**

- The parameters `diagmsgs` and `readper` are mostly for diagnostic purposes, and should not normally be changed from the defaults.
- When the parameter `negprimb` is set, the program reports the number of "negative primbsching" events which occur in the data. These are intervals for which the telemetered counts *exceed* the quadrant counts. The detector quadrant counts are not changed.
- The parameters `events` and `quadcts` point to the tables used for the calculations. *table0* is scanned to find the number of telemetered events in a detector quadrant during each interval, and *table2* is scanned for the detector quadrant counts
- *table1* should be used for `gaptimes`, to find gaps where a major frame is missing from the telemetry.
- Users should not change the value of `ctsoffset`.

To run **mkprimbsch** from the command line, producing a table of correction values for all of the DS/S detectors, simply type:

```
eu> mkprim primbsch sw,mw,lw,ds
```

mkprimbsch can print various messages on `STDOUT` to inform the user of its progress. The flags `infomsgs`, `warnmsgs`, `diagmsgs` and `negprim` to report various conditions. In particular, the warning: "values not found in deadtime table", indicates that some values of the detector count rate were not in the calibrated range of the TIF deadtime table. This happens only for extremely bright sources, such as the moon.

Out-of-range detector count rates cause **mkprimbsch** to use the largest fractional correction in the calibration table *tifdeadtime.tab*.

Each row of the `output` table includes the time stamp for an interval of two major frames (the 2.048 seconds which *end* at the time stamp), the telemetered and quadrant counts for each Spectrometer detector and the Deep Survey detector, correction factors, and errors. The **tables** task **ttools.tlcol** can be used to list the columns[6].

The most interesting columns are the ones named `Det4Q0dpc` and `Det[5-7]Q1dpc`. These are the combined deadtime and primbsching corrections for the spectrum quadrants and the central Lexan filter of the Deep Survey. For example, if some row contains the value 1.05 for `Det4Q0dpc`, then the number of counts detected in the SW in that interval should be multiplied by 1.05 to get the true value. Figure 1-6 shows a typical deadtime/primbsch correction column for a bright source in the LW detector, as plotted by **dqselect**. Remember that because of the table's length and the limitations of the graphic display, the plot shows a sampling of the data.

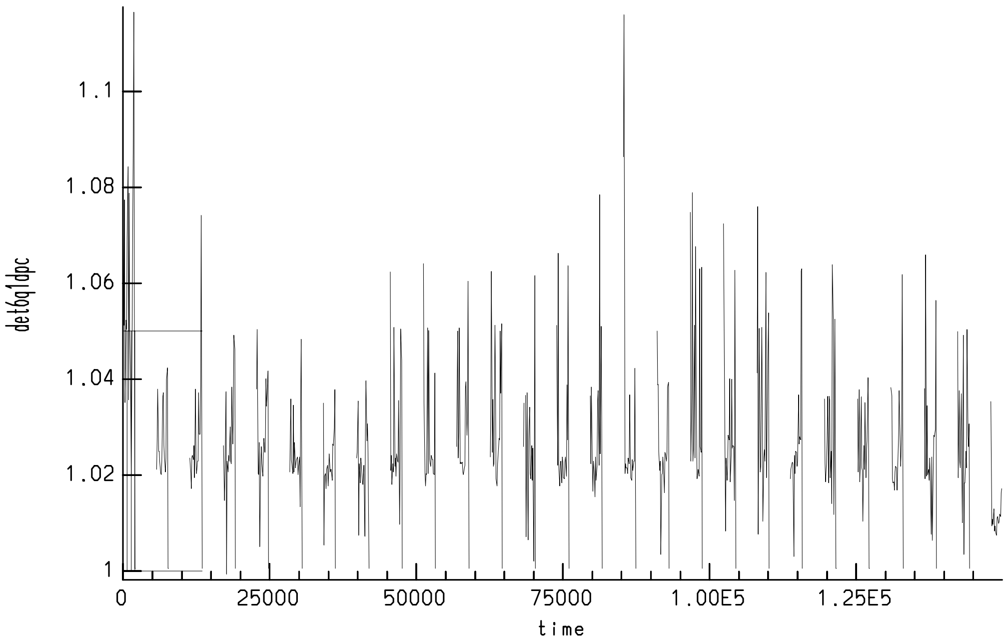

Figure 1-6: Deadtime/primbsch column displayed in **dqselect** with limits.

Gaps in the data affect the results in the primbsch table, and their occurrence is identified at run time if the task is run with `diagmsgs` on. The presence of gaps should be noted and understood by the user, but the messages are not signals of catastrophic failure. The resulting output table will still be useful; the image creation task **qpmkim** will apply a unit correction factor (no correction) for intervals

[6]Some of the columns are trivial or uninteresting. Quadrants 2,3 for detectors 5 and 6; 1,2,3 for detector 4, and quadrant 3 for detector 7 do not exist, and quadrant 0 in detectors 4-6 does not have the spectrum.

where the correction factors are missing or INDEF.

While the ADC counts are recorded every major frame, the filter quadrant counts are only recorded every other frame. If, for any pair of major frames, the major frame with the quadrant counts is missing, then the correction table will have a 2-frame gap. If a major frame is missing *between* the quadrant count reports, only the primbsch correction will be affected. This correction will be INDEF, and this will make the combined deadtime+primbsch correction column INDEF for that detector in both those frames. These two types of data loss normally occur in about one two-frame interval in every 5000; if they occur more frequently, something may be wrong with the input tables or data paths. Also, if for any interval the telemetered detector counts actually *exceed* the quadrant counts (negative primbsching), a primbsch correction factor of 1.0 is entered.

Related topics

- ◇ Correction for deadtime and primbsching is applied when a spectral image is created from the QPOE file. When you are ready to make a new corrected spectral image, go on to section 2.3.
- ◇ The task **qpmkim** has two different correction modes. Correction can be applied by adjusting the exposure time (the default), or by adjusting the number of counts in each pixel. See section the items in "Notes on running **qpmkim**" in section 2.3.1.

1.2.2 BACKMON makes tables of geographic orbit parameters

The EUVE telemetry contains quite a bit of geophysical data, but the format of the data may be unfamiliar, or difficult to integrate with the GO's analysis plan. For example, the aspect and ephemeris of the Spectrometer and satellite are recorded as quaternion components referenced to a celestial coordinate system, while investigators who wish to use data on the satellite's physical orientation may find this format inconvenient. The task **euvtools.backmon** (so named because its output can be used to monitor sky background) was written to help GO's create data quality filters based on geophysical parameters. GO's who wish to convert only the aspect quaternions into celestial coordinates can skip to section 1.3.1.

Given the location of the ST table containing the aspect and ephemeris reports (e.g. *table1.tab*) and a set of reference data, **backmon** calculates a list of selected orbit parameters for each major frame and puts them out in a new ST table. It calculates instrument look direction, satellite position and velocity vectors, and sun position in geocentric coordinates. A selection of reference angles can also be produced, as well as the satellite magnetic latitude, which often correlates with the local density of ions. The data can also be sampled at some user-specified interval. 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. See the IRAF help page for a complete listing of the output columns available and their units.

Do I need to run backmon?

The task **backmon** has already been run on *table1.tab* for all standard EUVE data sets, starting with IRAF/EUV version **1.4**. The output is in the table *backmon.tab* in the tarfile labeled *Processing* on the data tape listing. Most GO's will never need to rerun **backmon**. However, it is advisable to have some understanding of the quantities **backmon** produces and their role in the EGO Center nominal data reduction, so we suggest that all GO's inspect *backmon.tab* and read example 1.2, on creating the earth blockage filter.

A sample parameter file is shown in figure 1-7.

```
eu> lpar backmon


      output = "backmon"       Name of output ST table
       (skip = 10)             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 Greenwich 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")           detector (for boresight quaternions)
       (mode = "al")
```

Figure 1-7: Sample parameter file for **backmon**.

Notes on running **backmon**

- **backmon** uses the parameter `skip` to determine how often it should sample the aspect table. For example, if run with the parameters above it will calculate all the selected quantities for the first line of the aspect/ephemeris table, and every 10 lines thereafter. This can save space, at the expense of resolution.
- The parameter `maxtimegap` is the longest time in seconds that is allowed between successive sampled lines when computing the satellite velocity vector and the velocity view angle (angle in degrees from the DS/S look direction to the spacecraft ram direction). If successive sampled lines are separated by an interval larger than `maxtimegap`, these columns have the value INDEF. It is normal for the daytime gaps in telemetry to produce INDEFs.

- ◻ Specify only one detector as a boresight reference; all DS/S detectors are equivalent, as they share the same telescope.

Like **mkprimbsch**, **backmon** is usually run with standard parameters, and can easily be run from the command line:

```
eu> backmon backmon skip=10
```

Related topics

- ◇ One of the uses of *backmon.tab* is to test for earth blockage during a pointed observation. See section 4.3.2 in the *EGO Center Data Products Guide* and the example below.

Example 1.2 : Creating a new earth-blockage filter

The conditions used to make the earth blockage filter used in EGO Center processing are described in the *Data Products Guide*, section 4.4.2. It is installed by a process like the one described in example 1.1 above. The standard filter is conservative, in that it filters out events from paths up to a relatively high altitude above the earth's surface: 450 km. This was done to be sure that none of the data passed are from lines of sight with significant atmospheric absorption, which would skew any flux values that were derived. GO's whose images show that a significant number of events have been removed by the earth blockage filter may wish to use a lower altitude to define earth blockage, and to make a new filter that lets more events through.

The condition for earth blockage can be stated in terms of one of the angles calculated by **backmon**: `DSSZen`, which is the angle between the DS/S look direction and the spacecraft zenith vector.

$$DSSZen > 180 - \arcsin\frac{(R_e + A)}{(R_e + H)}$$

where R_e is the earth's radius, H (km) is the average height of EUVE's orbit above the earth's surface, and the signal is considered blocked for purposes of EUVE science when it must penetrate the atmosphere at an altitude $\leq A$ (km).

The GO must first determine, by some physical argument (and possibly considering in what part of the bandpass he expects the source flux), the altitude in the earth's atmosphere at which the source EUV photons are subject to enough absorption to decrease the detected count rate. Suppose he arrives at an altitude of 200 km above the earth's surface.

The GO next looks at the column `SatHt` of table *backmon.tab*, which gives the height of the satellite. The GO gets the min, max, and mean of this column using **ttools.tstat** [7].

```
tt> tstat backmon SatHt

 # backmon SatHt


 # nrows  mean          stddev    median    min      max
    4099  6883.719085   2.49885   6883.01   6880.83  6890.19
```

This shows a negligible variation in height over the observation, and supplies a value for the height H. Since `SatHt` gives the height from the center of the earth, the relation to the height in the earth blockage condition is $H = \overline{SatHt} - R_e = 6883.01 - 6378 \approx 505$ km. Using this value and 200 km for A, the GO calculates the limiting DS/S-Zenith angle above which the line of sight is blocked:

$$\text{Blockage} \equiv \texttt{DSSZen} \quad > \quad 180 - \arcsin \frac{(R_e + 200km)}{(6883km)}$$

$$DSSZen \quad \leq \quad 107.12$$

Notice the sense of the condition is changed– the GO is getting ready to make a QPOE filter, which requires a range of *acceptable* values for the attribute in question (time), so the limiting values of `DSSZen` should the times *without* earth blockage. To skip over earth-blocked data, he must select data with values of `DSSZen` less than 107.12 degrees.

Having determined this limit, the GO can easily make the appropriate filter string with **dqselect**. With the table *backmon.tab* loaded, **dqselect** displays a time plot of `DSSZen` on `STDGRAPH`:

```
Command> display dsszen
```

He sets limits on `DSSZen` at 0 and 107.12:

```
Command> edit dsszen

 Please pick the first limit value ...


 :y 0

 and again:

 :y 107.12
```

[7] The values shown are examples, and do not correspond to any particular GO data set.

The GO then generates the earth blockage good times from the `TableTime` column with the `gtgen` command, and saves them in a table:

```
Command> gtgen

 Generating good times for monitor, dsszen.


Command> gtsave myear_gt


 Good times saved to file, myear_gt.
```

The new earth blockage filter should be merged with the existing ADC count rate filter in each QPOE file to make a new exposure filter, so the GO quits **dqselect**and uses **euvtools.tfilter** to create and install the new filter macros:

```
eu> tfilter sw_adct_gt,myear_gt output=sw.qp name=losw_gt cmdline-
format=13.3f

eu> tfilter mw_adct_gt,myear_gt output=mw.qp name=lomw_gt

eu> ^mw^lw^g

 tfilter lw_adct_gt,myear_gt output=lw.qp name=lolw_gt
```

Once the **tfilter** parameters `cmdline` and `format` are set, they need not be reset each time the task is used.

1.3 Aspect and Pointing Problems

The pointing of the DS/S telescope is constantly subject to small drifts. While the pointing of the satellite is updated periodically, measurements of the satellite orientation, or aspect, are recorded in the telemetry more often: every 1.024 seconds. The readings are based on readings of the on-board gyros, star trackers, and the digital fine sun sensor.

The aspect correction algorithm in the EGO Center processing pipeline **cep** (Chapter 5) uses these satellite aspect reports to determine the true instrument pointing direction at the time of each detector event. The aspect is compared to the source position and correction factors are calculated that should restore each event to the position on the detector associated with a perfect, on-boresight pointing. If a stationary target's position is known with enough accuracy, the main limit on the spectrum focus is the quality of the aspect reports in telemetry.

However, targets for EUV spectroscopy include some objects that move too rapidly for EUVE's multiple attitude control system (MACS) to track continuously. These include planets, such as Mars and Jupiter, comets, and the moon. Observations of such objects have the source moving through the field of view, so that the relationship of the satellite pointing to the target position changes with time. The resulting spectra are smeared out in both the imaging and spectral directions on each detector, even after **cep** has performed its aspect correction, because **cep** assumes a constant source position on the sky. Occasionally, stationary targets are also subject to problems which cause inaccurate aspect reporting, with similar results.

The next two sections consider how to determine whether an observation is affected by any of these problems. We introduce tasks which GO's can use to evaluate the quality of pointing and aspect reports for their observation, and to reprocess the data from observations of sources with changing parallax.

1.3.1 MKASPECT creates coordinate tables from aspect records

The task **euvtools.mkaspect** creates a table of time-tagged celestial coordinates from time-tagged aspect quaternions in an ST table (e.g. *table1.tab*). This can be useful for checking the accuracy and stability of the spacecraft pointing during an observation, and for determining the positions of objects in the field of view. The output table has columns of times, Right Ascension, Declination, and roll angles. Because roll angles for the various detectors differ by multiples of 60 and 120 degrees, a different table is generated for each DS/S detector specified in the input. In addition, a column with the angular distance between the reported aspect and a user-supplied position on the sky (usually the target coordinates), can also be calculated.

Do I need to run mkaspect?

The task **mkaspect** is run for the Deep Survey detector as part of standard EGO Center processing. The resulting aspect coordinate table is called *mkaspectds.tab*,and is in the fourth tarfile on the data tape. This table contains the reported DS/S pointing direction in RA and Dec for all the DS/S detectors, with roll angles for the DS detector. All GO's should inspect the **mkaspect** table, but most will not need to rerun **mkaspect** unless they have some specific need, such as identifying the sources of multiple spectra. See the subsection titled **Source orientation in Spectrometer images** in section 1.3.1. If the DS detector was turned off because of high count rates, the task is run for the LW spectrometer, producing *mkaspectlw.tab*.

An example parameter file for running **mkaspect** is shown in figure 1-8. The target is Beta CMa, and the task is to be run from the source directory in a standard EUVE reduction setup. Separate output tables will be produced for the SW and MW Spectrometers. Angles from the DS/S optical axis to the source coordinates will also be computed for every line of the aspect table *table1.tab*.

```
eu> lpar mkaspect

      input = "observation/table1"Name of input aspect ST table
     output = "aspcoords"     Prefix for output aspect ST tables
      (skip = 10)             Number of rows to skip over
   (refdata = "reference/detector.tab") Name of ST table for reference data
 (detectors = "sw,mw")        Detector list
  (distance = yes)            Compute distance to boresight?
        (ra = 6.3783)         Central RA in hours
       (dec = -17.956)        Central DEC in degrees
      (mode = "al")
```

Figure 1-8: Parameter file setup for the task **mkaspect**. The parameters `ra` and `dec` are set to the target coordinates to produce the off-axis angles in the `DISTANCE` column.

Notes on running **mkaspect**

- The values of the parameters `ra` and `dec` should be set to the target coordinates desired if distance angles (see below) are to be computed.

- Although the aspect quaternion is reported every major frame, the calculations can be done less frequently by setting the parameter `skip`, which causes the task to pass over the given number of records after each calculation. This can shorten the resulting table if the aspect is known to be sufficiently stable over intervals of `skip` $\times 1.024$ seconds.

- If `distance` is "yes", the angle from the target coordinates `ra` and `dec` to the telescope boresight is calculated as a positive angle. Azimuthal information is not provided directly, but the distance angle can be combined with the roll information to determine the exact orientation of the telescope boresight to the source position given by `ra` and `dec`.

To run **mkaspect** in IRAf command line mode for the same target, and with `distance` on, one would type:

```
eu> mkasp observation/table1 aspcoord detectors=sw,ds distance+ ra=6.3783
dec=-17.956
```

The output tables will be called *aspcoordsw.tab* and *aspcoordmw.tab*, and each will have five columns: `TableTime`, `RA` and `DEC` of the detector boresight, detector `ROLL` angle, and the `DISTANCE` in degrees between boresight and the specified RA and Dec. Roll angles are given in degrees from the direction south to the positive X-axis of the *un-remapped detector image* (`dx`). Positive angles are counter-clockwise (ccw) from south to `dx`, while east is the direction 90 degrees *clockwise* from north[8].

[8] $\mathrm{Roll}_{sw} \approx \mathrm{Roll}_{ds} - 180, \mathrm{Roll}_{mw} \approx \mathrm{Roll}_{ds} - 60, \mathrm{Roll}_{lw} \approx \mathrm{Roll}_{ds} + 60$

Using mkaspect's output to verify pointing

All GO's should inspect the contents of *mkaspectds.tab* in **dqselect** or **tbplot.igi**, along with the DS detector image, to familiarize themselves with the behavior of the satellite pointing during their observation. Since **mkaspect** produces the reported pointing directions in celestial coordinates, this table may be more helpful than the raw quaternions in *table1.tab*. Plotting columns of *mkaspectds.tab* in **igi** is most effective, as the full range of values can be inspected at any resolution. See Appendix B for a tutorial. A plot of `RA` vs. `DEC`, such as that in figure 1-10, shows literally where the telescope was pointed on the sky.

Some of the irregularities that may affect pointing and aspect reporting are listed below; most pointing errors are accurately reported in the aspect and corrected for in the standard data reduction, but GO's should be aware of them. If the aspect report is in error, this must be evaluated and corrected by the GO.

- Looking at any coordinate column plotted against time will show any unexpected data gaps, such as a slew to another target. Figure 1-9 shows an **igi** plot of `RA` vs. `TableTime` for a 300 ksec observation that was interrupted for about two orbits to perform a pointing test[9]. The GO may want to check the gap times and exposure for the target.

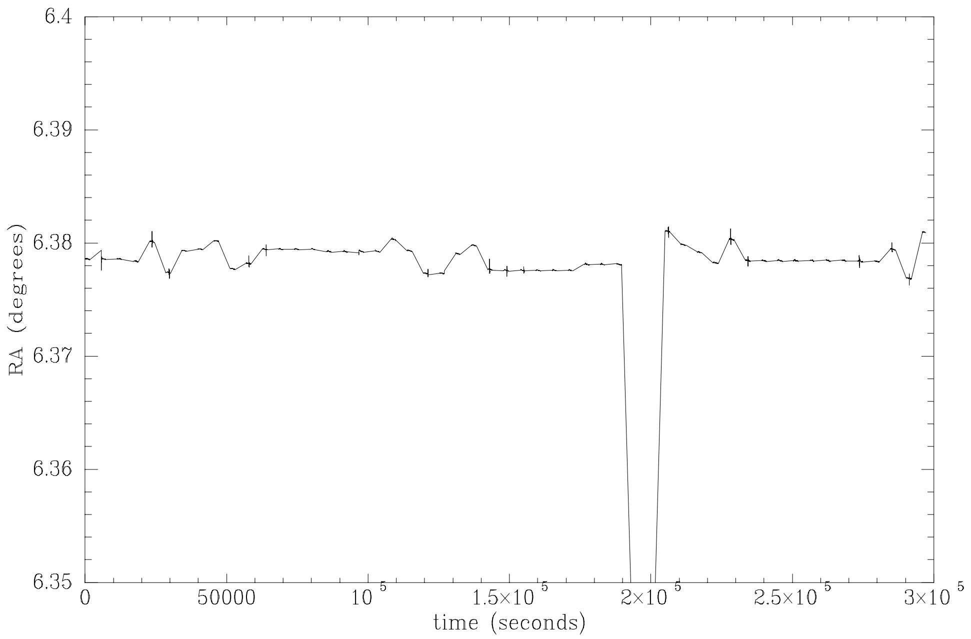

Figure 1-9: `RA` vs. `TableTime` from *mkaspectds.tab*, showing a data gap of 10,000 seconds. The observation was interrupted by a test, which lasted for 2 orbits.

[9] The telemetry for slews and time observing other targets is not included in the GO data set. This time is not counted against the integration time in scheduling or processing.

- The boresight RA and Dec in *mkaspectds.tab* correspond to the un-remapped coordinates of the telescope boresight, which may not coincide to the target coordinates. Most observations are pointed off-axis by 1–2 arc minutes to avoid the DS detector dead spot. The event positions are corrected using the aspect report during processing, and do not affect spectrum focus.
- Observations that have been dithered will show a series of systematic variations in RA and Dec, (and distance angle, if calculated), as shown in both figures 1-10 and 1-11. These deviations are also removed during processing by using the aspect report to remap the event positions.

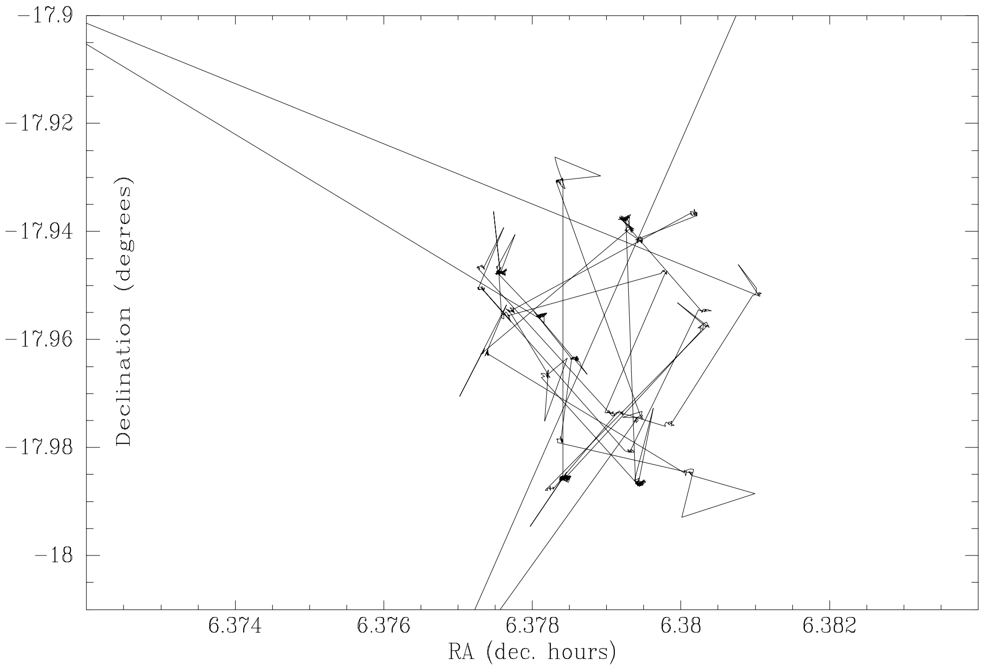

Figure 1-10: `RA` vs. `DEC` from *mkaspecds.tab*, plotted with **tbplot.igi**. The pointings of a dithered observation are small dots within 2 arc minutes of the of the target. The observation was interrupted twice.

- If the source image in the remapped DS detector image *ds.imh* is not at the detector center, then an unreported pointing error may have occurred. The aspect reports may be inaccurate, and the GO should take steps to find and correct the problem. Unreported pointing errors may be caused by things that "fool" the MACS, like telescope flexure or guide star confusion in the star trackers. Aspect errors will not be confirmed by **mkaspect**'s `distance` output. See section 1.3.2, on correcting for aspect errors.
- Inspecting a plot of roll vs. time can help to identify occasions when the roll angle was changed in mid-observation. This usually occurs when the available guide stars change, and is part of normal operations. However, there is always a possibility that one of the two positions in such an observation is subject

to telescope support flexure, which alters the pointing although the change is not reported in the aspect. The DS image and spectra may be blurred in such cases. Constant pointing offsets are addressed in section 1.3.2.

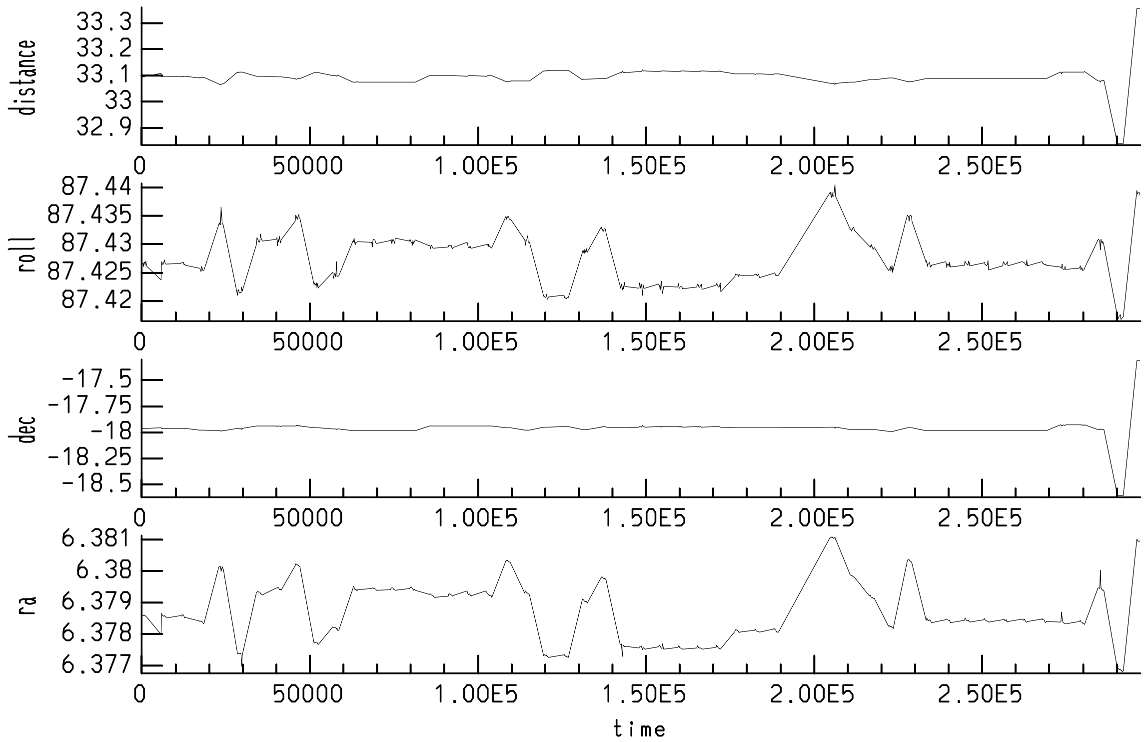

Figure 1-11: Coordinates from **mkaspect** for a dithered observation, plotted with **dqselect**. This plot shows RA, Dec, and roll for the DS detector.

Source orientation in Spectrometer images

Another use for the data from **mkaspect** is to determine the locations and relative orientations of multiple sources, or of different parts of an extended source like a supernova remnant. The DS QPOE file and the remapped DS image are produced so that north is always along the positive Y axis, with east increasing to the right. This is not practical for the Spectrometer QPOE's, as half of the position information for each event is embedded in the wavelength solution. In Spectrometer QPOE's, only the Y axis has position information, and the imaging angle has a different orientation on the sky for each channel.

The imaging angle of each detector event corresponds to the angle from boresight in a plane tangential to the circumference of the telescope aperture plane. This can be seen from figure 1-12, which shows that each detector Y axis is oriented tangential to its respective aperture section, while the X axes are arranged radially. This picture can be compared directly to the raw detector images, with appropriate rotations. The Z-axis points out of the page.

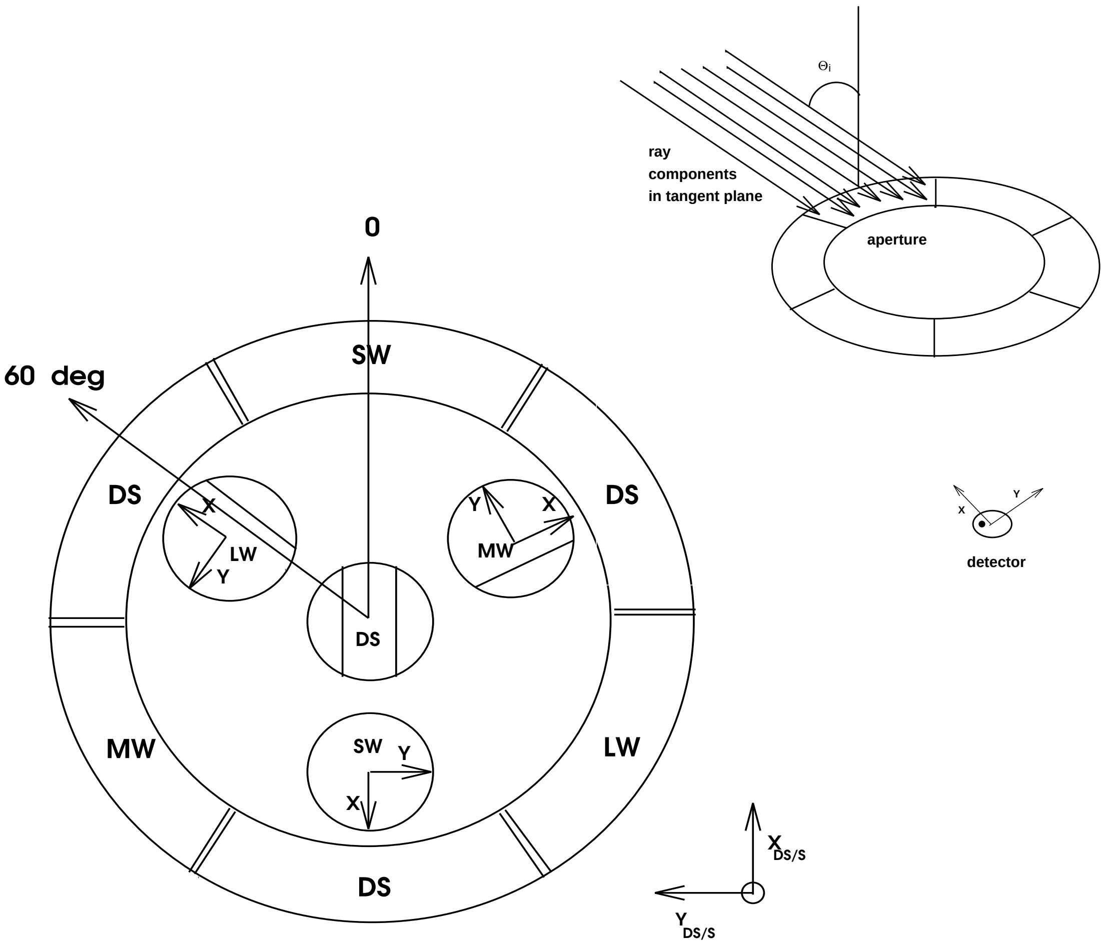

Figure 1-12: Orientation of detector coordinates in the DS/S. In the larger diagram, the observer is facing the detectors, with the optical axis (Z) pointing out of the page. The smaller diagram illustrates the definition of the imaging angle, Θ_i.

The radial geometry of the instrument means the vector displacement of a point off the boresight is resolved with a different fractional component along the on the imaging axis in each detector. The orientation of north to the Y axis depends on to the telescope roll angle and the angular separations of the Spectrometer detectors' Y axes (120 deg). To find the orientation of multiple sources in one of the Spectrometer images, it is recommended that the GO run **mkaspect** for that particular Spectrometer channel. Although the relation between the roll angles for the different DS/S detectors is simple, this saves the trouble of worrying about sign changes with an odd number of reflections, and whether offsets are to be added or subtracted from the values in *mkaspectds.tab*.

The roll angle in *mkaspect*w.tab* is unique to the observation. When applied as stated above, the roll angle tells how the Spectrometer Y axis is related to the directions north, south, east and west for that particular detector during that observation. The difference in orientation for detector images (*_det.imh*) and remapped spectral images (*.imh*), is small enough to be ignored in a first approximation. Once the relation of the Y axis to north is established, this defines the direction on the sky associated with the imaging angle. But one must bear in mind that the imaging angle is only one component of the vector displacement between the sources.

For example, suppose two bright spots in a supernova remnant produce spectra separated by an angle of .05 degrees in the SW image, and the roll angle produced by **mkaspect** for the DS/S is 57.5 degrees. Since roll is given in degrees ccw from south to `dx`, this means that `dy` points 23.5 degrees ccw (west) of due north. The two sources must be separated by .05 degrees on a line 23.5 degrees ccw from north, with the target nearest boresight to the south and east (see figure 1-13). A check on the relative shift in X between the two spectra might supply 0th order information about their separation along the perpendicular direction.

Related topics

- ◇ Sections 3.3 and 4.3.4 of the *EGO Center Data Products Guide* describe the table *mkaspectds.tab* that is delivered on the GO data tape.
- ◇ When two spectra are present in an image, the EGO Center reduces the data for only the requested pointing. If the second source position can be found, e.g. from a DS image, the GO may wish to rerun the event pipeline on the data to create QPOE files with the second spectrum, as the target. See section 5.2.1 on correcting constant pointing offsets and aspect errors.

1.3.2 ASPCORR creates new aspect tables from source positions

euvtools.aspcorr is a task designed to help GO's process observations with moving targets or aspect reporting errors which change with time. While telescope pointing cannot be corrected after the fact, unreported pointing errors can be addressed in the EGO Center reduction pipeline using the task **aspcorr** if some reliable source of information on the target's apparent position in the field of view (based on the aspect report) is available.

aspcorr requires the satellite aspect reports in *table1.tab*, a fixed target position, and an ST table with the apparent celestial coordinates of the target at regular intervals as input. **aspcorr** produces a new table of aspect readings which can be used to rerun the event pipeline, **cep**. The new table will have one line for each line in *table1.tab*; if necessary, the apparent positions are interpolated between the times given in the input to produce aspect reports with the same times as the original aspect table.

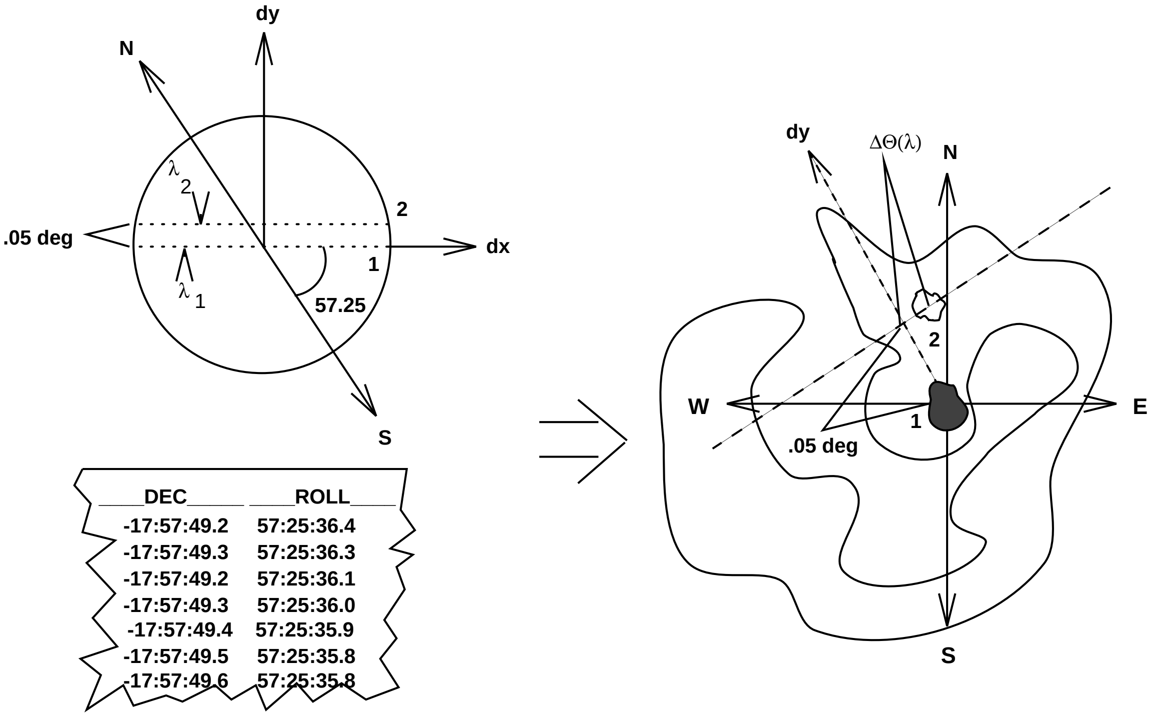

Figure 1-13: Using **mkaspect** output to locate off-boresight sources in the Spectrometers. Cartoon compares the roll angle data from **mkaspect** with the orientation of the detector imaging axis on the sky. The position of the source along the line perpendicular to `dy`, labeled $\Delta\Theta(\lambda)$, is related to the shift in X position of a spectral feature which appears in both spectra, as indicated by the $\wedge$, $\vee$'s on the spectral image.

aspcorr uses the apparent positions to calculate a compensating factor for each of the original aspect reports. Figure 1-14 illustrates the calculation for a source moving through the DS/S field of view. For the location $\vec{t}$ of the target at some moment, $\vec{t}'$ is the direction of the actual pointing, which is offset by some factor $\Delta\vec{t}$. The aspect is reported as $\vec{a}$; the true aspect is $\vec{a}'$. The dashed circle shows the radius of normal fluctuations in the aspect between pointing updates. It may or may not include the target location, depending on $\Delta\vec{t}$.

The difference $\Delta\vec{a}$ is the correction which **cep** calculates from the aspect report and *catalog.tab*, and does not include the offset $\Delta\vec{t}$. **cep** assumes it to have the relation $\vec{a} = \vec{t} + \Delta\vec{a}$ to the target. When **aspcorr** is run, the apparent locations in the input table correspond to the target position $\vec{t}$ at different times. For each aspect report, **aspcorr** calculates the factor $\vec{\delta} = \Delta\vec{a} - \Delta\vec{t}$, which includes the offset between the pointing and the target direction for that time. If $\vec{t}$ and $\Delta\vec{t}$ are rapidly changing functions of time, e.g. for the moon, the compensation must be calculated for every aspect report ($\approx$1/second).

In other words, each line in the aspect table created by **aspcorr** has a compensating factor added to the original satellite aspect quaternion, which is equal and opposite

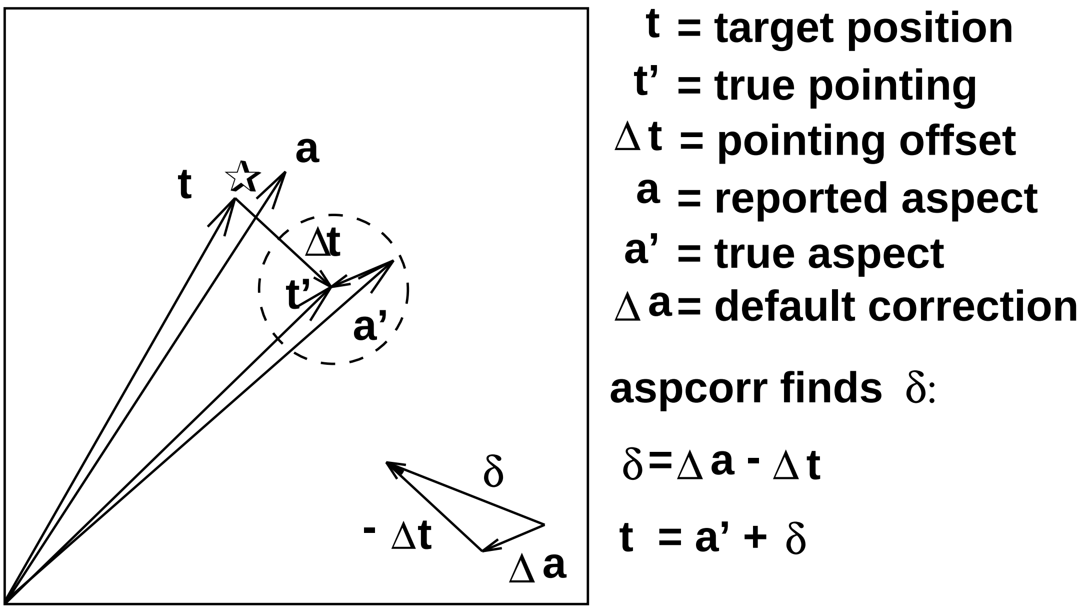

Figure 1-14: Vector geometry for correcting changing aspect error. The dashed circle represents the range of normal aspect variation – about $\widehat{30''}$. If **cep** uses the apparent location $\vec{t}$, it will calculate the compensating factor $\vec{\delta}$.

to the difference between the fixed target position and the apparent target coordinates. When the new aspect table and the same fixed target position are input to **cep** the pipeline produces a corrected QPOE file. The new aspect table "fools" **cep** into compensating for the aspect errors, with the result that each event position is corrected to the current target position for that time, and the resolution of the spectrum is improved. However, this process depends on the precision with which the user knows the apparent coordinates of the source in the DS/S field of view.

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 satellite could not track the target accurately, the target's apparent RA and Dec can be calculated from known target positions and the satellite aspect and ephemeris in *table1*. The output table from **backmon** produces several vectors, referenced to common geocentric coordinate systems, which may be useful.

2. In cases where the aspect reports for a distant target are not accurate, centroids can be obtained from the Deep Survey data if the source has sufficient flux in the Lexan bandpass. EUVE GO data tapes processed under version **1.5** include all events from the DS Lexan filter in *table0.tab* and in a DS QPOE file. The frequency with which centroids can be produced depends on the count rate.

Do I need to run aspcorr?

This task is for solving a particular kind of problem: when the actual pointing offset is not equal to the difference between the target coordinates and the reported aspect, AND the difference is changing in time. As indicated above, the two main reasons this occurs are that the target was actually moving in the instrument field of view, or that the star trackers were not properly locked onto guide stars, causing unreported aspect drift. Unless you know that your observation is subject to one of these two conditions, you probably *don't* need to run **aspcorr**. Other aspect problems, like flexing of the telescope support structure or use of inaccurate target coordinates in the reduction pipeline, produce constant aspect offsets which are simpler to correct. These are discussed in section 5.2.1, on rerunning **cep** for constant offsets.

In situations where the apparent position or aspect error are changing rapidly, it is best to supply positions at a frequency of about 1 row per second. Once the apparent coordinates are obtained, the procedure is the same: pack the coordinates into a standard format ST table using a task like **ttools.tcreate**, run **aspcorr** to create the new aspect table, substitute it for *table1*, and rerun **cep** with the corrected aspect table to produce the corrected QPOE file(s).

A typical parameter file for running **aspcorr** with a table of positions for a moon observation is shown in figure 1-15.

```
     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 1-15: Sample parameter file for running **aspcorr** on a lunar observation. The fixed source coordinates, `alpha` and `delta`, are arbitrary when dealing with a moving source like the moon.

Notes on running **aspcorr**

- The parameter `position` should point to a three-column ST table with the apparent celestial coordinates of the object at intervals of 1 second or more. The table should have the format shown in figure 1-17. The frequency of the positions may determine the accuracy of the correction for fast-moving sources, such as the moon.

- □ If target coordinates were obtained from an ephemeris, they may need to corrected for parallax. The position must reflect the RA and Dec of the target as seen from the position of the EUVE satellite. The satellite ephemeris can be obtained directly from the columns `PositionX/Y/Z` in *table1.tab* (in meters), or from the output of **backmon**(in Lat/Long/Ht).
- □ The times in the `position` table must *span* (i.e. include times both before and after) the reported times `Table1Time` in *table1*.
- □ The name of the original satellite aspect table is supplied in `asptable`, and will generally be *table1* for EUVE data.
- □ Fixed source coordinates in the field of view are supplied in `alpha` (RA, Hrs:min:sec) and `delta` (Dec, degrees.dec). The same position will be used to run **cep** with the corrected aspect table. For moving targets, the choice of coordinates is arbitrary, *as long as it is correct for the coordinates in the* `position` *table.*
- □ The three parameters `ra`, `dec` and `time` are used to specify the column names used in the position table, which in this example, happen to be called `RA`, `DEC`, and `Time`. Column names are case sensitive.
- □ The parameter `aspect` supplies the column names of the aspect quaternion in `asptable`. The default values are those for EUVE telemetry tables, and need not be changed.

Note that the command line parameters are `position`, `asptable`, `output`, `alpha`, and `delta`. Thus, the task could be run using the position table *moon.tab* 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 as those in *table1.tab*.

Since aspect is the only information from *table1* that is used by **cep**, the new aspect table can be substituted directly for *table1.tab*. The next step is to run **cep** with the new aspect table to produce QPOE files with more accurate event locations.

Related topics

- ◇ See Chapter 5 and especially section 5.2 to correct aspect problems by rerunning the event pipeline.
- ◇ Events in an area around the boresight can be selected for various intervals to create a record of the pointing offset in time, as described in section 5.2.2.
- ◇ Section 5.3 of the *Data Products Guide* also describes the effects of some aspect reporting errors, and the procedure called "dithering."

Example 1.3 : Making a new aspect table for 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 processed using **aspcorr**, using apparent coordinates for the moon, as calculated by the GO. The detector image in the LW spectrometer, after uncorrected processing with **cep**, is shown in figure 1-16. The accumulated image of the uncorrected data looks like a number of stripes, nearly perpendicular to the spectral direction.

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 $\widehat{20'}$ FWHM of the collimator transmission. The image shapes are elongated in Y and shortened in X because the optical speeds along the dispersion and imaging axes are different. 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. The GO wrote a program which

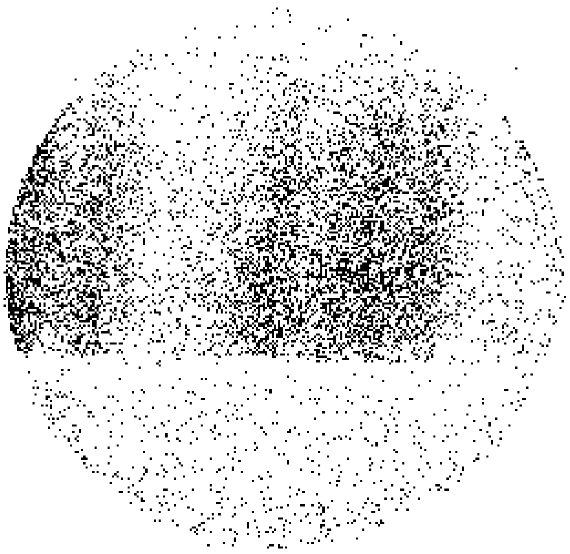

Figure 1-16: The moon in the LW Spectrometer, after standard processing with **cep** and the satellite aspect. Streaks show the paths of spectral features across the detector during the observation.

uses the satellite ephemeris from *table1.tab* and the satellite look direction in *backmon.tab* to calculate the apparent RA and Dec of the lunar disk every 2.048 seconds, and created a table with columns of times, RA, and Dec. The first few lines of *moon.tab*, containing the apparent celestial coordinates of the moon, are shown in figure 1-17.

The GO then ran **aspcorr** using the parameter settings shown in figure 1-15 to produce the ST table *new_table1.tab*. The new table contains the "corrected"

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 1-17: Part of an ST table input to **aspcorr**, with apparent coordinates of the moon for alternate major frames.

aspect quaternions, each with an offset equal and opposite to the difference between the apparent coordinates and the reported aspect. A few lines of *new_table1.tab* are shown in figure 1-18.

Column	1	2	3
Label	_________AspectT_________	_________AspectW_________	_________AspectX_________
1	774678133.1390079	0.6644102135026706	-0.3740989780195041
2	774678134.163008	0.6644057080561128	-0.3741017126098966
3	774678134.7840004	0.6644029725503054	-0.3741033700186317
4	774678135.187008	0.6644011547295496	-0.3741044181743463
5	774678136.2110082	0.6643962913513004	-0.3741070834119812
6	774678136.8320002	0.6643933395028272	-0.3741088446477844
7	774678137.2350078	0.6643914734961072	-0.3741100191197195
8	774678138.2590079	0.664386962646536	-0.3741125250967082

	4	5
Label	_________AspectY_________	_________AspectZ_________
1	-0.5232151820760572	0.3805978845892577
2	-0.5232145074152104	0.3806041354529378
3	-0.5232139558862321	0.3806080750180024
4	-0.5232136479903312	0.3806106641431434
5	-0.5232131065275788	0.3806170105491646
6	-0.5232129243487749	0.380621007662806
7	-0.5232127482948769	0.3806235635017167
8	-0.523212066339116	0.380629823501779

Figure 1-18: Columns of an output table from **aspcorr**.

Note that the first time in figure 1-17 is slightly *before* the first time `AspectT` in the aspect table; **aspcorr** requires that times in the input position table span those in the aspect table. The new aspect table was then substituted for *table1* in re-running **cep** (see directions and example 5.2 in section 5.1).

The resulting spectral image is shown in figure 1-19. 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.

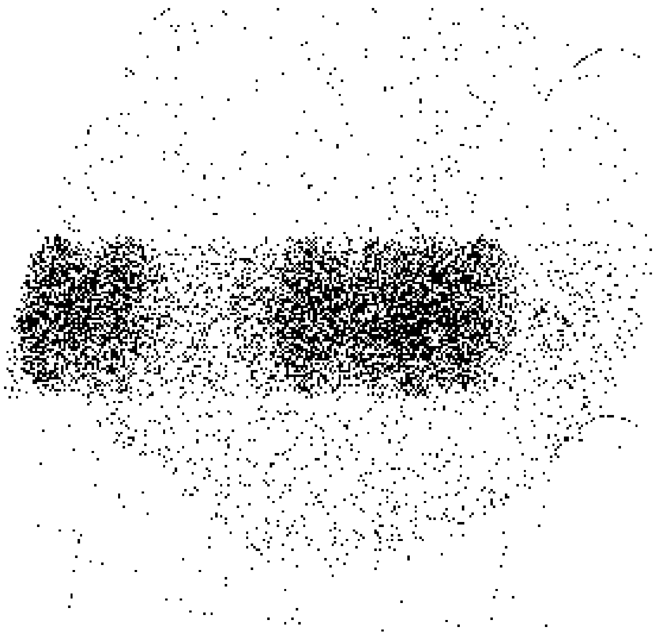

Figure 1-19: Lunar observation after processing with the standard pipeline and corrected aspect table created by **aspcorr**.

The aspect correction for a distant source with an inaccurate aspect table is improved by essentially the same method. Any record of the object position can be converted to apparent coordinates in an ST table, and used to calculate the corrected aspect table with **aspcorr**, before rerunning the pipeline. If positions from DS data are to be used, they may be interpolated to achieve more frequent positions, if another source of information, such as a source ephemeris, is available.

1.3.3 QCALC calculates quaternions

The task **euvtools.qcalc** is a quaternion calculator that recognizes some common forms of astronomical coordinates. As many as 26 quaternions can be loaded into registers named `a, 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>
```

Exit the task when finished, 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. 0.9200000000000001 0. -0.73

qcalc> d = 12 54

qcalc> &d

 -0.5877852522924731 7.198055546404513E-17 0.8090169943749474

qcalc> !d

 180. 54.

qcalc> ^D

 12:00:00.0 54:00:00.0
```

Typing "&" or "!" before a register name signals the task to print the contents in vector or degree form, respectively, with vector as the default.