CGRO/OSSE
Guest Investigator Guide

Version 1.1.1

Compton Gamma Ray Observatory
Oriented Scintillation Spectrometer Experiment

Naval Research Laboratory
Washington D. C.
and
Goddard Space Flight Center
Greenbelt, MD

Release Date: December 15, 1995

Editor: W. T. Bridgman, GSFC

0.1 Acknowledgements

This guide was the product of effort by many individuals, from the OSSE Principle Investigator Team, to the *Compton* Gamma-Ray Observatory Science Support Center, and even the guest investigators themselves.

The OSSE PI team provided most of the initial documentation from which this guide was constructed, particularly in regard to the Reference Guide. They also provided immeasurable assistance in the construction and validation of the exercises in the text. Among those I particularly wish to thank are Neil Johnson, Mark Strickman, Kellie McNaron-Brown, Byron Leas, Eric Grove, and Dick Kroeger, all at the Naval Research Laboratory.

The guest investigators were particularly helpful in providing insight on what they wish to know from the database. What are useful ways to examine the data for extracting real science from it. In many instances, I was performing the process for the first time working on their data. These tasks were later converted into exercises in this guide. I thank them for their tolerance as I also learned the system, particularly Edison Liang (Rice University), Greg Madejski (GSFC/USRA), Jim Ling and Evren Tumer (JPL), Paul Nandra (GSFC), and Luigi Piro and Marco Feroci (CNR).

Contents

Chapter 1

Overview

1.1 Introduction to OSSE Data Analysis

This manual is an introductory guide for Guest Investigators who wish to analyze data from the Oriented Scintillation Spectrometer Experiment (OSSE) aboard the *Compton* Gamma Ray Observatory. The primary CGRO/OSSE data analysis package is IGORE (Interactive GRO/OSSE Reduction Environment). This quide enables the Guest Investigator to examine and manipulate OSSE data on a basic level. It is not meant as a detailed tutorial for IGORE. The IGORE system is based on the commercially available software package IDL, and retains all the functionality of IDL while adding specific capabilities which are required when processing the large amounts of data necessary during GRO/OSSE data analysis.

The IDL basis of the IGORE analysis system provides IGORE with an interactive programming environment for the analysis and display of data. IDL also provides the ability to easily add user defined procedures and functions to the core IGORE system. In fact, many of the IGORE commands are procedures or functions written in IDL which have been incorporated into the core IGORE analysis system.

1.2 What's in this Manual

This manual is intended as a quick guide to extracting information from OSSE database. As such, it is designed for the inexperienced IGORE user, though experienced IGORE users may find many of it's exercises useful. Many of the exercises are based on actual Guest Investigator inquiries. If your analysis requirements cannot be met by this guide, contact the GRO/OSSE Instrument Specialist at the address below. Your analysis requirements may provide the basis for exercises in the next release of this guide. If your analysis requirements are highly specialized, you may need to expand your knowledge of IGORE. The *IGORE User's Guide* can familiarize you with the operational concepts of IGORE. The primary purpose of this manual is to provide a description of IGORE system procedures and functions which will be of use for typical OSSE data analysis. See the *OSSE Data Analysis User's Guide* for additional examples and more details regarding the use of these procedures and functions in the analysis of OSSE data. Beyond that is the*IGORE Users Utility Reference Manual* provides detailed documentation of the low-level functions and procedures in IGORE. As before, the GRO/OSSE Instrument Specialist is available for assistance.

This manual does not describe native IDL commands; a description of the IDL system and syntax can be found in the *IDL User's Guide* or using the IDL on-line help command by typing `?` at the `Igore>` prompt. On-line help for many of the commands described in this document can either be accessed via the command IGRHELP at the `Igore>` prompt, or using the IDL library documentation facilities `DOC_LIBRARY` or `XDL`. The latter requires X-Window support at the user's terminal.

1.3 Where to get this Manual

A copy of this manual can be obtained by contacting the CGRO/OSSE Instrument Specialist at the CGRO Science Support Center or can be downloaded electronically through the World Wide Web. The addresses are:

William T.(Tom) Bridgman
CGRO/OSSE Instrument Specialist
Goddard Spaceflight Center
Code 660.1
Greenbelt, MD 20771

E-mail: bridgman@grossc.gsfc.nasa.gov

World Wide Web OSSE Home Page
http://cossc.gsfc.nasa.gov/cossc/OSSE.html
(Look under High Level Products)

Please send feedback, comments, and suggestions to the above addresses as well.

1.4 Syntax Conventions

The following syntax conventions are used in this manual:

- The main IGORE prompt is printed as `Igore>`. The actual IGORE prompt may differ, depending on the local system configuration.
- IGORE commands, parameters and keywords are specified using San Serif font.
- IDL commands, parameters and keywords are specified using `Typewriter font`.

- Square brackets ([ ]) are used to indicate either that the enclosed items are optional parameters or, in the examples, as the IDL array concatenation operator. The intended use should be clear from the context.

- Curly brackets ({ }) are used in the examples to provide parenthetical comments and are not part of the command line.

Unless otherwise stated, all user inputs are assumed to be terminated with a carriage return.

1.5 Applicable Documents

The reader may find some of the following documents useful, or even essential for analysis of OSSE data:

- *OSSE Data Analysis User's Guide* – OSSE Document #926-225.
- *IGORE User's Guide* – OSSE Document #926-175.
- *IGORE User Utility Reference Manual* – OSSE Document #926-224.
- *IGORE Internal/System Utility Reference Manual* – OSSE Document #926-226.
- *Spectral Data Base Header Definition* – OSSE Document #926-227.
- *IGORE Programmer's Guide* – OSSE Document #926-176.
- *IDL User's Guide*[1]
- *IDL Reference Guide*[1]
- *IDL Internals*[1]

[1] Copyright Research Systems, Inc. IDL is a trademark of Research Systems, Inc.

1.6 Acronyms and Glossary

AU	Analyzable Unit
DTW	Digital Time Window
FDB	Fit Data Base
FDR	Fit Data Record
IDL	Interactive Data Language
IGORE	Interactive GRO/OSSE Reduction Environment
ODS	On-board Data Storage
OSSE	Oriented Scintillation Spectrometer Experiment
PDF	Pulsar Data File
PSF	Pulsar Selection File
SDB	Spectral Data Base
SDR	Spectral Data Record
TSF	Telemetry Scalar File
COSSC	Compton Observatory Science Support Center
GROSSC	Gamma-Ray Observatory Science Support Center
NRL	Naval Research Laboratory

Chapter 2

Introduction

The information and exercises described in Chapter 3 are intended for asking the basic question "Did OSSE see the object?". As part of this process, data files are created which provide a good "stepping-off" point for more detailed analyses. Some of these more detailed analyses are described in Chapter 4. These exercises make use of a number of high-level routines in IGORE.

The OSSE spectral analysis applications support SDB header version 7 only. Information on the most common SDB AUX records are given in Chapter A.

It is important to note that, for many of the spectral analysis applications, the indirect SDR fields (*e.g.*, the HDR, DAT and ERR fields, etc.) **need not be resident in memory** to successfully execute the application, as long as valid FILE and TAG fields are available in the SDR. If the required indirect field is not available in memory (*e.g.*, it has not been read into the SDR with either SDB$READ or SDR$GET), the application will attempt to read the required information directly from the SDB file using the FILE and TAG fields contained in the SDR. The search for the SDB file using the FILE field is performed in the following sequence:

- The current directory is searched using the string FILE.
- The current directory is searched using the string FILE and with the file type ".SDB".
- The directory path logical IGR$SDB is used with the string FILE.
- The directory path logical IGR$SDB is used with the string FILE and with the file type ".SDB".
- The message "Error opening SDB file." is reported.

Note that the applications SDB$OPEN and SDR$GET search for the SDB file in the same order. The application FILE_INFO may be used to determine which files are open and the associated logical unit.

IGORE creates, maintains, and uses records in a nested hierarchical system, allowing record structures that are more complex than the structures allowed by IDL. These IGORE records are not directly accessible to IDL. Instead, IDL variables are created as pointers to these IGORE records. It is therefore important to note the distinction between IGORE records and structures versus IDL records and structures.

A subset of the IGORE routines can be used to move data between IGORE and IDL variables. The routines include; GETFLD, PUTFLD, GETAUX, PUTAUX, GETAUXREC and PUTAUXREC.

Many of the routines that create and manipulate IGORE records accept optional keywords, including NAME and /UNIQUE. These keywords allow the user to explicitly specify the name of an IGORE record, or to force a unique name to be generated, respectively. By default IGORE record names are assumed to match the names of the IDL variables used as record pointers.

See the *IGORE User's Guide* for a full discussion of the IGORE philosophy and the related usage considerations.

Chapter 3

An Introduction to SPECANAL

3.1 SPECANAL: A First Line of Attack on the OSSE Database

3.1.1 What is SPECANAL?

SPECANAL is a high-level tool for performing spectral analyses from the OSSE database. At present, it is designed for use with observations of point sources which have corresponding background pointings. It can operate with a set of system-defined defaults to generate a preliminary analysis of the data for a specified target and viewing period. For the more advanced user, keywords and configuration files permit alteration of the default settings so that more customized analyses can be performed.

3.1.2 How does it run?

SPECANAL requires the specification of the Target and the time period to analyze. The time period is identified by viewing period or by start and stop dates.

The front-end portion of SPECANAL creates a script of IGORE commands which are then executed either as a batch job or in the foreground. SPECANAL calls the five basic IGORE routines which an investigator would otherwise run manually, assigning file names in an organized fashion and documenting the run. This run can be described in five basic parts.

1) BLD_QUAD_DIF: This procedure performs quadratic background estimation and subtraction on the OSSE spectral memory data for the specified target and time range. It performs various types of data screening and generates the daily difference spectral files. Each file contains one day's worth of two-minute difference spectra.

2) DAY_SUMMER: This routine sums the two-minute difference spectra into longer intervals, typically orbital or daily totals. Two output files are generated, a day sumfile (DSM) and

a good sumfile (GSM). The good sumfile output handles error propagation due to some background correlations.

3) FFIT_PREP: This routine prepares the data for fitting, summing the spectra by detector and performing adjustments due to detector operation. In the default mode, it also builds a response matrix. Your particular IGORE installation may not have the routines for building the response matrix.

4) PLOTANAL: This routine generates plots of spectra and rate time histories from a variety of perspectives.

5) BOTTOMLINE: This routine generates a total count spectrum and deconvolves the count spectrum with a default model to produce a photon spectrum, plotting both results.

Required Inputs:

A Target Specification, either by name, (see Example 1) or a targdef description record including the name and position (in Right Ascension and declination) (see Example 2). If using a target name, it should conform to the format used in the OSSE planning catalog. This standard format has been chosen to conform to the designation in the SIMBAD database. Some alias names are supported from other catalogs.

Time Period, either by viewing period number (see Examples 1 & 2) or by start and stop time specification.

These specifications can be included in the SETUPFILE input parameter or can be identified in keywords, TARGET, VP, START_TIME, STOP_TIME. Setup files provide an easy method for rerunning the programs with identical settings if necessary. Keywords, on the other hand, can be easier to use and are more readily incorporated into an IDL program.

The next section gives examples for performing basic OSSE data analysis with SPECANAL and its friends. These examples are designed as an introductory tool and not meant as an exhaustive tutorial. You are encouraged to experiment with different parameter settings and namelist files.

For those with X-Window terminals, you may find it helpful to setup SPECANAL runs using it's graphical interface extension, OGRE. If you are running the testing version of IGORE at NRL, just type OGRE at the IGORE prompt.

Note: If you wish to run SPECANAL in background from your user account, your LOGIN.COM file must be set to initialize the appropriate version of IGORE, either IGRTST or IGRCM.

3.2 User Examples

3.2.1 Example 1: The Quick-n-Dirty Analysis

The first step to perform if you want to analyze OSSE data on a target is to determine if OSSE actually viewed the object. The easiest method to make this determination is to run XWHEN. Simply type:

```
Igore>XWHEN
```

On X-Windows, this will open a window with a scrolling window to select a target from a list. After the target is selected, the other scrolling window will list the viewing periods where the target was observed. Click on the desired viewing period, then click Display Current Selection. A window will open displaying the pointing strategy of all four detectors for the target. A window of target statistics will also be displayed. Click on the Done button when you are finished.

Now let's extract some data. Use the standard defaults. All you want to do is see some kind of result for a particular target and viewing period, for this particular example, Hercules X-1 in viewing period 208. This method assumes you have the ENTIRE OSSE database online. Generally this is only true at NRL.

```
Igore> SPECANAL, target = "HER X-1", vp = 208.0
```

This run generates the complete set of output files described in the reference section.

3.2.2 Example 2: My target isn't in the OSSE database

Suppose you hunt for your target in XWHEN as in Example 1a above and can't find it listed. All is not lost. OSSE has a *very* wide field of view, about 4 deg by 11 deg, and not all objects in that field of view for a given pointing are listed in the target catalog.

First, you must search the OSSE target catalog for official targets close to your target of interest. Use XWHEN as described above to examine the detector pointing strategy. Would the pointing strategy used provide acceptable target and background pointings for analysis?

If OSSE had any pointings where your target is in that field of view, you can find it by creating a dummy target record for SPECANAL.

```
Igore> targdef = {Name:'Fuzzy Spot',RA: 325.123, Dec: -54.32, SrcID: 007}
Igore> SPECANAL, target = targdef, start_time='92/090',stop_time='92/100'
```

This run generates the complete set of output files described in the reference section.

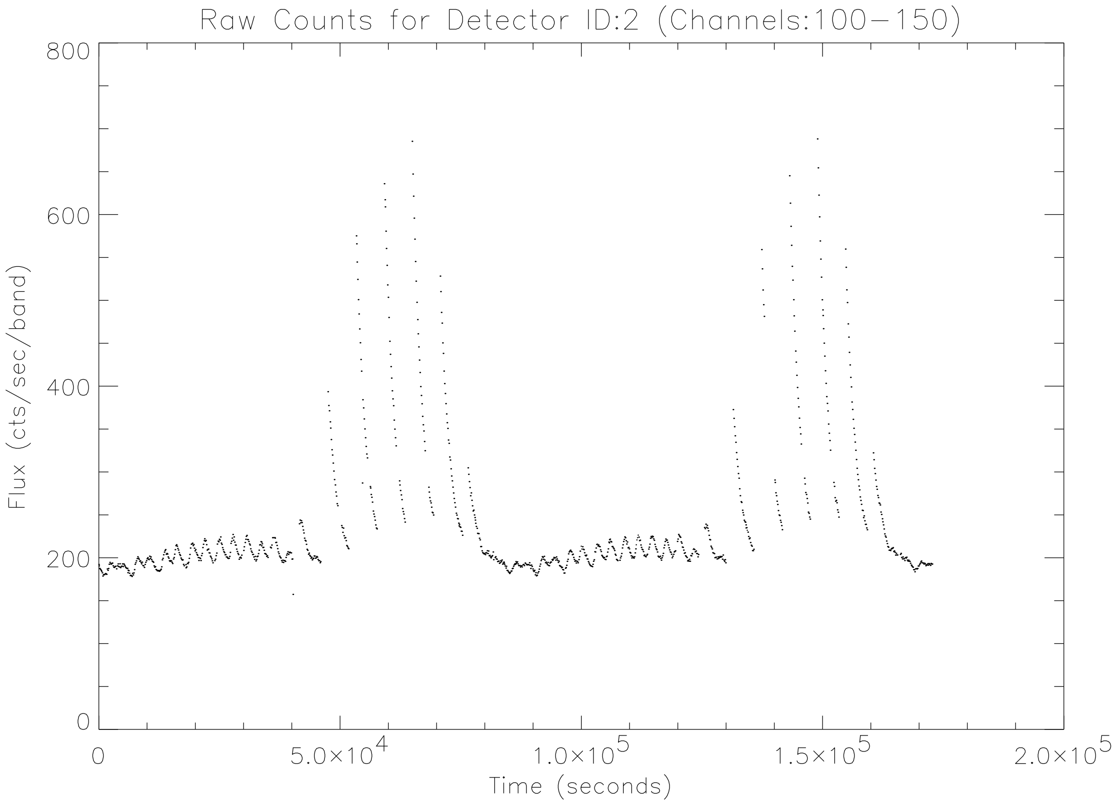


Figure 3.1: OSSE raw counts in channels 100–150 (approximately 551 – 857 KeV) over a period of two days.

3.2.3 Example 3: Screening out "bad" data

Occasionally a spectral observation may display features which defy explanation. You occasionally must ask — Is it the source, or is it some instrument problem? There are two common sources of error that should be checked on some observations:

- The South Atlantic Anomaly (SAA): When the spacecraft passes through this region, it is bathed in radiation which activates a number of components. This creates a high-than-normal background which decays with a half-life of about 25 minutes. It is readily visible some energy bands, such as shown in Figure 3.1.

- On-Board Data Storage (ODS): When the telemetry tape recorders failed, a work-around was developed where the data gathered were reformated in a compressed form and stored aboard the spacecraft's memory until the next contact with a ground station or TDRSS satellite enabled the data to be downloaded to the ground. In this compressed format, some instrument housekeeping data is lost so data with problems may make it into the SDB files improperly flagged. We can remove these data packets and process only non-ODS data which contains all the OSSE error flags.

To perform these two types of screening, we will use two different processes. For removing the ODS data, we will instruct SPECANAL to ignore SDR records with the ODS flag set. We will also write a user-screening routine which will pass only records where the spacecraft's orbit does not send it through the SAA.

For this outing, we will take a look at the Seyfert Type 1 galaxy NGC 4151 observed by OSSE during viewing period 222.0. If we perform a normal SPECANAL run and use FITIT with a power-law model , we obtain a result like the that displayed in Figure 3.2.

For whatever reason, we are curious as to whether the spectrum has suffered any contamination from such things as distorted background estimates from SAA passages or possibly mis-flagged data sets from ODS. How do we re-run SPECANAL to exclude this data?

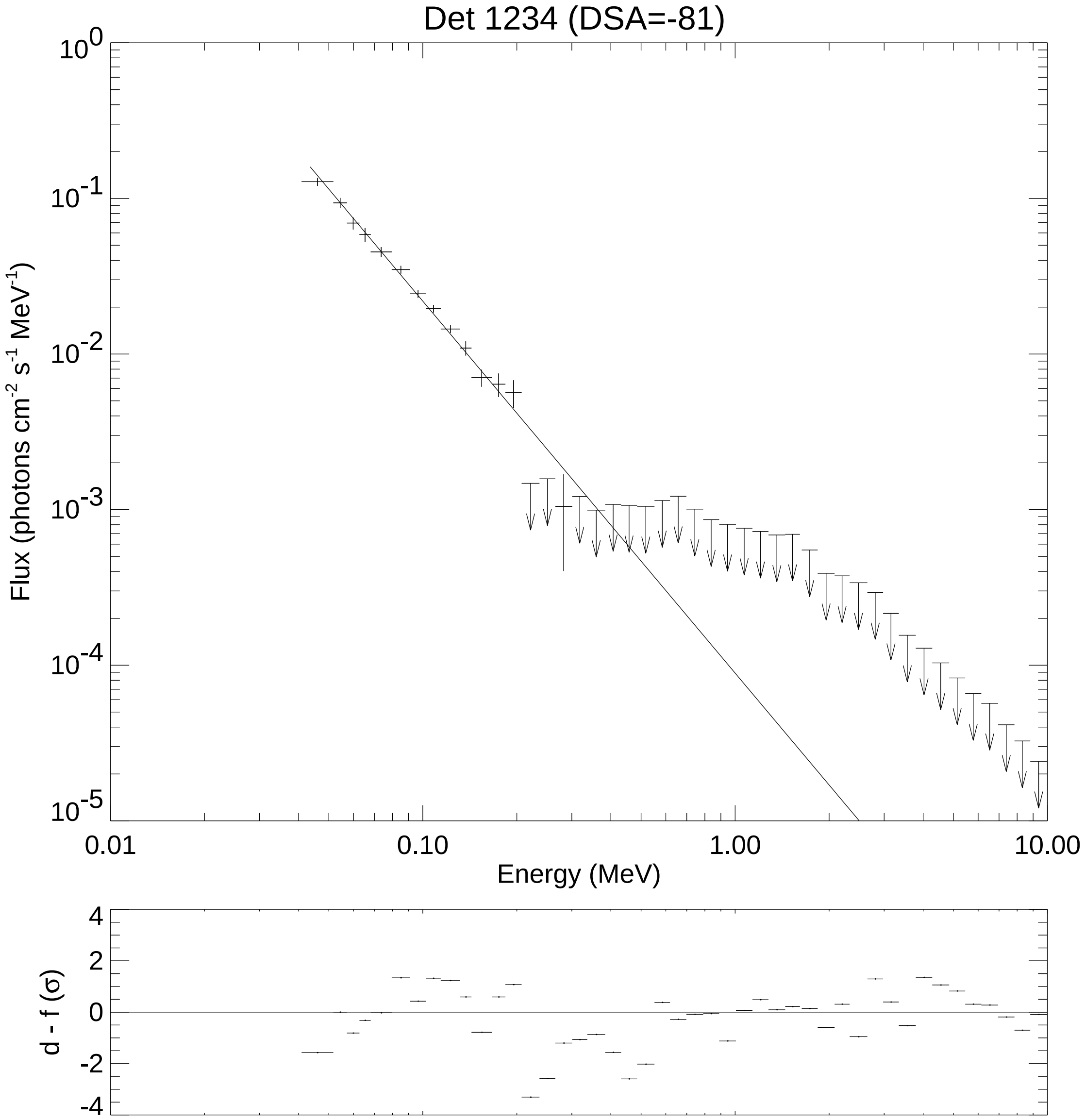
Since orbits with SAA passages is not a piece of information stored in each SDR, we cannot use the standard data flagging information. However, each SDR does contain information from which we can derive whether the spacecraft passed throught the SAA on a given orbit. Since the information must be derived from the SDR data, we need to write a User Screening routine. The basic User Screening routine takes pointers to the SDRs as input and returns and array of indices which point to SDRs *accepted* for further processing. In the snippet of code below, SAA_SCREEN.PRO, an array of indices is returned of SDRs which were collected more than 4000 seconds since the last SAA passage (enough to exclude a given orbit):

```
pro saa_screen,sdr,indxrs
; FILE: SAA_SCREEN.PRO
;
; Set SAA_limit to time from last SAA end want to discard
;
SAA_limit=4000.0  ; 4000.0 secs since last SAA exit
SAA_last = getfld('hdr.last_saa_exit',sdr)
indxrs=where(SAA_last ge SAA_limit) ; keep only if SAA_limit secs beyond SAA
return
end
```

Next, we need to tell SPECANAL to igore data where the ODS flag is set. This flag corresponds to bit 20 in the DSFStat status word and generates a value of 1048576 as a long integer. This is the value we set for DSFSTAT_MASK=1048576. We will use a total of three different namelist files to control these parameters.

To control the building of the background subtracted 2-minute spectra, we use the file NGC4151_SCREEN.BQD listed below:

```
; %NameList% BQD_STRUC % Structure created 12-Sep-1995 16:21:11.00
VIEWPERIOD =        222.000
START_TIME = "93/144 17:10:34"
STOP_TIME = "93/151 12:14:37"
DSFSTAT_MASK =      1048576
PROD_DIR = "$PROD:"
TARGDEF = "NGC 4151"
USERSCREEN = "saa_screen"
/END ; of setup element definition
```




```
1995 Sep 12 15:44
Config: NGC4151_NOSCRN_CFG.FIT      Model: NGC4151_NOSCRN.MDL
SDR_SUM: Det 1234    93/144 17:24:14 - 93/151 12:12:26 Accum:   269811.7 sec
SDR_SUM: NGC 4151 VP 222 Det 1234
   ChiSqr/Dof:  1.032  NDof =    446     (Prob =  0.312278)
   En Range:    0.041 -  9.977 MeV
   Powerlaw Function for NGC 4151
No       P            dP          Pmin-P        Pmax-P   Cnstr  Par_Name
 1   2.1826e-02   5.0356e-04                             SIGN+  Amplitude
 2  -2.3896e+00   5.0898e-02                             SIGN-  Pwrlaw Index
 3   1.0000e-01                                          FIXED  Normalizing Energy
```

Figure 3.2: A Spectrum of NGC 4151 from Viewing Period 222.0. The model is a power-law fit. Only standard screening has been performed on the data.

We use the file NGC4151_SCREEN.dsm to control the DAY_SUMMER (as a technical point, we don't really need to specify DSFSTAT_MASK and USERSCREEN in this paramter file since the selection and screening performed by BLD_QUAD_DIF has already extracted the appropriate SDRs.)

```
; %NameList% DAYSUM_STRUC % Structure created 12-Sep-1995 16:21:53.00
OUTFILE = "NGC4151_screen_DSM.SDB"
GOOD_SUMFILE = "NGC4151_screen_GSM.SDB"
DSFSTAT_MASK =      1048576
PROD_DIR = "$PROD:"
USERSCREEN = "saa_screen"
START_TIME = "93/144 17:10:34"
STOP_TIME = "93/151 12:14:37"
/END ; of setup element definition
```

Finally, the file NGC4151_SCREEN.fp control FFIT_PREP (the major reason we must write a namelist file for FFIT_PREP is to specify the proper good-summed SDB file for input.)

```
; %NameList% FIT_PREP_STRUC % Structure created 12-Sep-1995 16:23:14.00
FILEROOT = "ngc4151_screen"
SDBFILE(0) = "NGC4151_screen_GSM.SDB",
DAYFILE = "NGC4151_screen_DAY.SDB"
OUTFILE = "NGC4151_screen_TOT.SDB"
/END ; of setup element definition
```

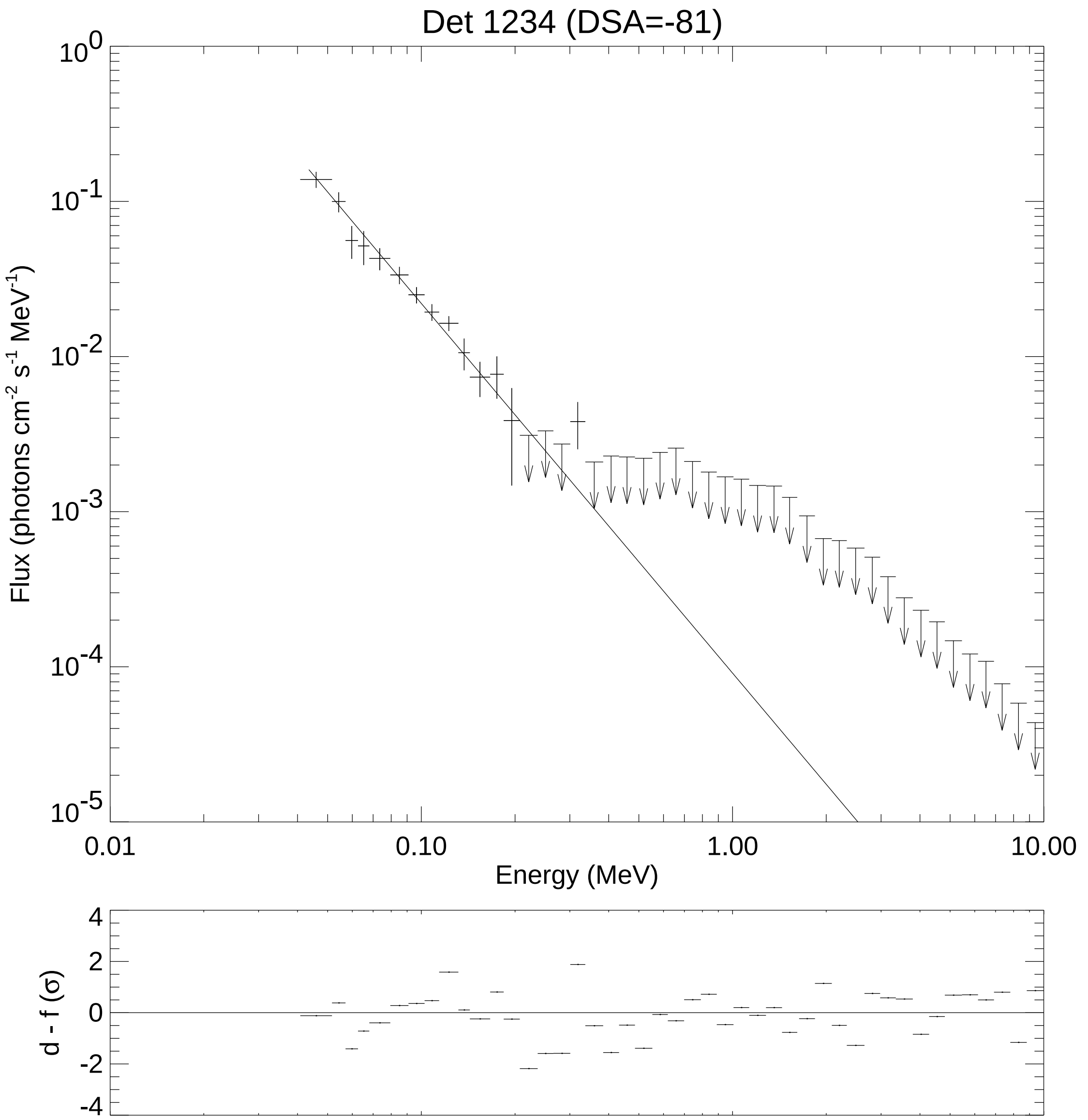
Then start SPECANAL running...

```
Igore> specanal,target='ngc 4151',vp=222.0,fileroot='ngc4151_screen',$
Igore>         bqd_config='NGC4151_SCREEN.BQD',$
Igore>         fitprep_config='ngc4151_screen.fp',$
Igore>         daysum_config='NGC4151_SCREEN.DSM',$
IGORE>         /nobln,/noplotanal,$
Igore>         que='sys$ossep'
```

The final results from FITIT are plotted in Figure 3.3. Note that the error bars are larger and the accumulated live time is smaller. This is the kind of tradeoff one must make when screening data. It is up to the investigator to determine if screening actually improves the results.

3.2.4 Example 4: How do I get There from Here

You are at a remote location with your own complete copy if the IGORE system and you want to analyze data from the COSSC. The first problem you'll encounter is that since you do not have a complete installation of the OSSE database at your site, you must be more selective when you run SPECANAL. Basically, you must tell SPECANAL exactly where to look for the OSSE data files.




```
1995 Sep 12 17:23
Config: NGC4151_SCRN_CFG.FIT      Model: NGC4151_SCRN.MDL
SDR_SUM: Det 1234    93/144 18:35:13 - 93/151 01:47:46 Accum:    60899.3 sec
SDR_SUM: NGC 4151 VP 222 Det 1234
   ChiSqr/Dof:  1.058  NDof =    446     (Prob =  0.192715)
   En Range:    0.041 -  9.984 MeV
   Powerlaw Function for NGC 4151
No       P            dP          Pmin-P        Pmax-P   Cnstr  Par_Name
 1   2.1959e-02   1.0831e-03                             SIGN+  Amplitude
 2  -2.3823e+00   1.0877e-01                             SIGN-  Pwrlaw Index
 3   1.0000e-01                                          FIXED  Normalizing Energy
```

Figure 3.3: A Spectrum of NGC 4151 from Viewing Period 222.0. The model is a power-law fit. Only data from orbits without an SAA passage and non-ODS data are included in this spectrum.

First you must transfer and convert the SDB files from the COSSC archive to your system. For this exercise, say we wish to examine three days worth of data on the Crab Nebula. We need to download the files:

```
P921770000_SPM.SDB
P921780000_SPM.SDB
P921790000_SPM.SDB
```

and place them in a directory, which, for the purpose of this example, we'll call $disk7:[username.crab].

Next, we must tell SPECANAL precisely where to find these data files. We do this using the namelist files for BLD_QUAD_DIF and DAY_SUMMER. We can start by making a copy of the BQD and day_summer configuration files from the default directory:

```
$ copy igr$igore_data:nom_cfg.bqd $disk7:[username.crab]crab_cfg.bqd
$ copy igr$igore_data:nom_cfg.dsm $disk7:[username.crab]crab_cfg.dsm
```

Now let's examine the configuration files. First look at the BLD_QUAD_DIF configuration file. In this case, we can retain all the normal processing and just inform SPECANAL the location of the production directory. We do this by setting the PROD_DIR keyword to $disk7:[username.crab] as done below.

```
; %NAMELIST%  BQD_STRUC % - Build_Quad_Dif Structure
;  This es an example structure for standard processing/ no special
screening
START_TIME = ""
STOP_TIME = ""
BACKGROUND =        0
DQHAZ_MASK =           0
DQINFO_MASK =           0
DSFSTAT_MASK =           0
EARTHSEP =       72.0000
FILE_EXT = ".SDB"
FILE_MOD = "_DIF"
MIN_LTL =      0.000000
NOPRI =        0
NOSEC =        0
NOSOL =        0
NOTRANS =        0
NOSTANDARD =        0
OUT_DIR = ""
PROD_DIR = "$disk7:[username.crab]"
SOURCE_OFFSET =       1.00000
SPM_PREFIX = "P"
SPM_SUFFIX = "_SPM"
TARGDEF = ""
USERSCREEN = ""
/END ; of setup element definition
```

And here's the configuration file for DAY_SUMMER. We have made the same change to PROD_DIR.

```
; %NameList%  Structure created 18-Aug-1994 18:10:44.00
; No. of Functions =  1

; Function #        1
DQHAZ_MASK =            0
DQINFO_MASK =            0
DSFSTAT_MASK =            0
NOSTANDARD =        0
PROD_DIR = "$disk7:[username.crab]"
USERSCREEN = ""
E_RANGE(0) =   4.5000002e-02,  2.5000000e-01,
THRESH =      0.900000
NODSA =        0
NOSTANDARD =        0
ORBIT =        0
START_TIME = ""
STOP_TIME = ""
DSA_SCREEN =        0
BYCMDSTATE =        0
EARTHSEP =       72.0000
COVAR =        1
VERBOSE =        0
/END ; of setup element definition
```

Now we run SPECANAL with the following parameters:

```
Igore> specanal, target='crab', start_time='92/177',$
Igore>   stop_time='92/179', bqd_config='crab_cfg.bqd',$
Igore>   daysum_config='crab_cfg.dsm'
```

Alternatively, we can create a file, crab_cfg.setup with the setup configuration built in:

```
; %NameList% ANAL_STRUC % Structure created 31-Mar-1995 12:59:30.00
TARGET = "{NAME: 'CRAB PULSAR', RA:   83.521, DEC:   22.020, SRCID: 372 }"
START_TIME = "92/178 00:00:00"
STOP_TIME = "92/179 00:00:00"
OUT_DIR = "$DISK7:[username.CRAB]"
FILEROOT = "CRABPULSAR_32"
LOGFILE = "CRABPULSAR_32.LOG"
BQD_CONFIG = "$DISK7:[username.CRAB]CRAB_CFG.BQD;3"
BQD_DIR = "$DISK7:[username.CRAB]"
DAYSUM_DIR = "$DISK7:[username.CRAB]"
DAYSUM_OUT = "CRABPULSAR_32_DSM.SDB"
DAYSUM_GOOD = "CRABPULSAR_32_GSM.SDB"
DAYSUM_CONFIG = "$DISK7:[username.CRAB]CRAB_CFG.DSM;2"
```

```
FITPREP_OUT = "CRABPULSAR_32_TOT.SDB"
FITPREP_DIR = "$DISK7:[username.CRAB]"
FITPREP_CONFIG = "OGRE$DAT:NOM_CFG.FP"
PLOTANAL_DIR = "$DISK7:[username.CRAB]"
PLOTANAL_CONFIG = "OGRE$DAT:NOM_CFG.PA"
BLN_OUT = "CRABPULSAR_32_BLN.SDB"
BLN_DIR = "$DISK7:[username.CRAB]"
BLN_CONFIG = "OGRE$DAT:NOM_CFG.BLN"
/END ; of setup element definition
```

Then SPECANAL is executed with the simple command:

```
Igore> specanal,'crab_cfg.setup'
```

Chapter 4

Beyond SPECANAL

4.1 Applications

SPECANAL is merely a filter between the OSSE database and the researcher. It gives you control over the data you wish you view and gives you a rough idea of just what OSSE saw. It does not, however, do any real science and generally does not present the data in the most desirable form. This requires much more interaction from you, the researcher.

This section gives examples of how you can extract more information from the data extracted by SPECANAL. These files can be generated by either performing your own SPECANAL run on the low-level OSSE data or you may retrieve this files already generated from the COSSC High- level archive. As mentioned before, this is not meant as an exhaustive tutorial. You are encouraged to experiment.

4.2 User Examples

4.2.1 Example 1: I Don't Like 'The Standard Model'

Suppose you don't like the default E^{-2} model used by SPECANAL for deconvolving the spectrum from count- to photon-space. It is simple enough to generate outputs with a different model and generate a fit at the same time. Using the basic data files generated by SPECANAL, use the IGORE routine FITIT to examine the data using a different model.

First, create a model with initial parameters using BUILDMODEL:

```
Igore> BUILDMODEL, ms
```

This prompts the user for model types and parameters. You can add multiple models through this routine. Next you must save the model to a file on disk:

```
Igore> status=MODEL_WRITE('filename.mod', ms)
```

Next you can either create a namelist file for FITIT, or call the routine specifying keywords.

For the namelist file option:

```
Igore> status=fitit(file='namelistfile.NML')
```

For the keyword option or the namelist file, there are at the very least, three entries necessary. First, you need the filename of your model. Second, you need to specify the name of the SDB file which contains the totalled difference spectra and the response matrix. You create the model file yourself, the SDB file <fileroot>_TOT.SDB is created by SPECANAL and contains both the difference spectra and the response matrix. Finally, to save any output to files, you need to specify a FILEROOT name which becomes the prefix for the output files. Therefore your absolute minimum namelist file might look like this:

```
; Namelist file for FITIT
;   Minimum file model for Cygnus X-1
;
SDBFILE         =       "cygx-1_2_TOT.SDB"
FILEROOT        =       "cygx-1_2_tbrem"
MODELFILE       =       "thermalbrem.mod"
```

Using keywords, running FITIT would look like:

```
Igore> status=fitit(sdbfile='cygx-1_2_tot.sdb',$
Igore> modelfile='thermalbrem.mod', fileroot='cygx-1_2_tbrem')
```

When FITIT first begins running, you are prompted if you wish to read a new setup file. Answer no (N) to this prompt.

The FITIT routine would choose a set of "reasonable" defaults for the other parameters. Of course, IGORE's definition of reasonable and yours may vary. Let's take a look at a more detailed namelist file:

```
; Namelist file for FITIT
;   Thermal Bremsstrahlung model for Cygnus X-1
;
TITLE           =       "Cygnus X-1: Thermal Bremsstrahlung Model"
SDBFILE         =       "cygx-1_2_TOT.SDB"
MODELFILE       =       "thermalbrem.mod"
FILEROOT        =       "cygx-1_2_tbrem"
PLOT_ERANGE(0)  =       0.01,   10.0
PLOT_FRANGE(0)  =       0.00001,        10.0
PLOT_CHANMOD    =       "log"
PLOT_NCHAN      =       25
ERANGE(0)       =       0.05,   1.0
```

where TITLE gives a title to present at the top of a plot; SDBFILE, FILEROOT and MODELFILE, as before, give the names of the input data file, the default output file name, and the model to be used for fitting; PLOT_ERANGE is used to expand the plotting range in energy out to 0.10 to 10.0 MeV; PLOT_FRANGE gives minimum and maximum values to plot in photon flux ($photons/cm^2/s$); PLOT_CHANMOD specifies that the standard spectral bins in the SDRs should be combined into PLOT_NCHAN channels of logarithmically scalled widths; and finally ERANGE specifies the minimum and maximum energy range over which to perform the model fitting.

As for output files, FITIT generates an SDB file with the photon spectrum, <fileroot>_PHOT.SDB; a text file table of the photon spectrum (suitable for installing in your own plotting routine), <fileroot>_PHOT.DAT; and a model file with the fitted results, <fileroot>.MDL. The .MDL file is in standard ASCII model file format, appropriate for input in routines such as PHOTONMODEL and FOFX. The graphs generated are in <fileroot>.ps and show a flux spectrum with residuals and parameters from the model fit.

For more details on the keywords available for customizing a FITIT run, run the IDL doc_library and ask to view the FITIT procedure.

4.2.2 Example 2: Processing Individual Days of Data

Let's say you've examined the light-curve reports from the standard SPECANAL products and see some hints of variability in the source. You want to examine the spectra for individual days or a range of days to discern the nature of the variability. How do you go about it?

First, you need an SDB file with the daily summed SDRs. Next, you need the appropriate response matrix. Daily summed SDRs reside in the Good-Sum (GSM) SDB file. The response matrix resides in the total (TOT) SDB file. FITIT requires a link between the SDRs and the response matrices in order to function. How do we do this?

It is possible to run SPECANAL so that it will generate the required file in its initial run, but we don't need to go back that far. It is also possible to generate the appropriate daily file if you have the regular output files from SPECANAL, particularly the GSM (Good-Sum) SDB file.

In IGORE, run the FFIT_PREP command with the /daily flag set. This will generate a <fileroot>_DAY.SDB file which contains pointers to the response matrices in <outfile>.

```
Igore> status=ffit_prep(file='ogre$dat:nom_cfg.fp',$
Igore>      SDBfile='$disk9:[osse.hilev]cygx-1_2_5_GSM.SDB',$
Igore>      outfile='cygx-1_daily_tot.sdb',$
Igore>      fileroot='cygx-1_daily',$
Igore>      filesetup='NONE',$
Igore>      logfile='cygx-1_daily_prep.log',$
Igore>      /daily)
```

Once the day file is generated, it is ready to use.

```
Igore> sdb=select_files('*day.sdb')
```

If more than one *DAY.SDB file is available you will be presented with the option to select the file desired. Otherwise this function will return with the value of SDB set. Next, let's load in the SDRs and examine their detectors and starting times.

```
Igore> sdr_load,sdb,sdr                   ; load all SDRs
Igore> start=getfld('hdr.strttim',sdr)    ; get the SDR starting times (
optional)
Igore> det=getfld('hdr.detid',sdr)        ; get the SDR detector IDs (
optional)
```

Next, generate an index to select the SDRs which will be summed. Sum them, making sure to attach the response matrix. Let's say for this example we wish to sum the first four (indices 0, 1, 2, and 3) SDRs

```
Igore> sdr$create,r_out,1                 ; create an output record
Igore> sdr_sum,sdr(0:3),r_out,/matrix     ; add the SDRs into r_out
```

At last, you have an SDR (r_out) with an appropriate averaged response matrix. Now what?

At this point you have two options. You can continue the analysis in IGORE, or you can convert the SDR into a FITS format suitable for reading in XSPEC. To continue in IGORE, you can run FITIT on the individual SDR you've created, much like the previous example

```
Igore> status=fitit(sdr=r_out,modelfile='pwrlaw.mod',$
Igore>    fileroot='cygx-1_2_5_day1')
```

Again, you answer N to the 'Read new Setup File?' prompt.

Alternatively, you can easily convert the new SDR into a count spectrum and response matrix file for use in XSPEC with:

```
Igore> igore_to_xspec,r_out,'cygx-1_2_5_day1'
```

This will create two files, the count spectrum with a .PHA file extension and the response matrix with a .RMF file extension.

4.2.3 Example 3: Hard Analysis of Hard Sources

If your target has significant emission up in the several MeV range (targets such as the Crab pulsar or the Sun), problems can arise due to the finite size of the detector response matrix. The default settings for SPECANAL generates a response matrix covering the nominal energy range of the OSSE detector: 0.05–10 MeV in count space. Photons with energies above the detector's nominal energy range can generate counts at lower energies so generally the response matrix is created covers energies from 0.02–12.0 MeV with 460 logarithmically-spaced channels.

For this exercise, we will take the daily good-summed (GSM) count spectra and regenerate the response matrix expanding the photon energy range. To respecify the generation of the response matrix, we need to use the MTRX_NPHOT and MTRX_PRANGE parameters in FFIT_PREP. In this example, we will expand the energy range in photon space to cover 0.02 to 20.0 MeV with 600 channels. We will also select new names for the output files to separate them from the normal SPECANAL run.

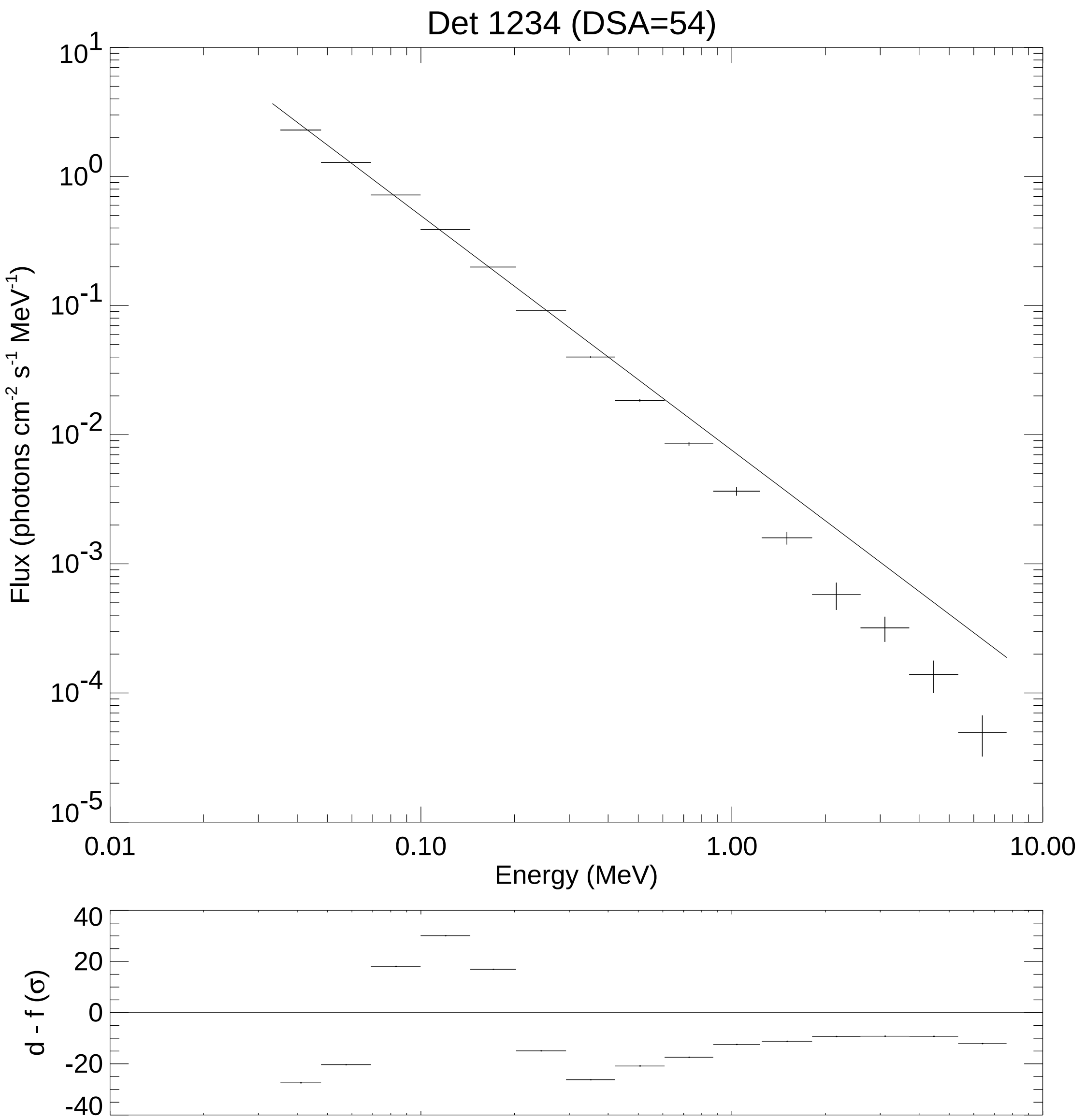
```
Igore> status=ffit_prep(file='ogre$dat:nom_cfg.fp',$
Igore>     sdbfile='od10:[gimgr.crab]CRABPULSAR_221_GSM.SDB',$
Igore>     outfile='crabpulsar_hard_tot.sdb',$
Igore>     fileroot='crabpulsar_hard',$
Igore>     filesetup='none',$
Igore>     mtrx_nphot=600,$
Igore>     mtrx_prange=[0.02,20.0])
```

Once the new total file is generated, it is ready to use.

```
Igore>; now combine the individual detectors
Igore> sdr_load,'CRABPULSAR_HARD_TOT.SDB',sdr
Igore> sdr$create,r_out,1
Igore> sdr_sum,sdr,r_out,/matrix

Igore>; and fit the resulting SDR
Igore> stat=fitit(fileroot='CRABPULSAR_HARD',$
Igore>          sdr=r_out,$
Igore>          modelfile='PWRlaw.MOD')
```

The result of this fit is shown in Figure 4.1. How much does this expanded response matrix alter the fit? We can examine the contents of the text output files CRABPULSAR_SOFT_PHOT.DAT and CRABPULSAR_HARD_PHOT.DAT. From these we notice that the flux in a given energy band is different with the expanded response matrix, though for this observation, it is differs only by a small amount. That will be true for many OSSE observations. The effect is only strong for bright, hard sources , such as solar flares.

```
OD10:[GIMGR.GI_GUIDE_DEV]CRABPULSAR_HARD_PHOT.DAT;1

;    [Mon Aug 14 14:38:27 1995]
;columns: 1 = E_low(MeV), 2 = Width(MeV), 3 = Flux(pho/MeV/cm2/s),   4 = Sigma
15     rows
4         columns
   3.532876e-02   1.239002e-02   2.304310e+00   7.621887e-03
   4.771877e-02   2.134042e-02   1.287403e+00   3.693308e-03
```




```
1995 Sep  5 09:30
Config: CRABPULSAR_HARD_CFG.FIT      Model: CRABPULSAR_HARD.MDL
SDR_SUM: Det 1234    93/133 18:23:16 - 93/144 11:18:01 Accum:   227033.1 sec
SDR_SUM: CRAB PULSAR VP 221 Det 1234
   ChiSqr/Dof: 18.476  NDof =    446     (Prob =  0.000000)
   En Range:    0.031 -  7.677 MeV
   Powerlaw Function for CRAB PULSAR
No       P            dP          Pmin-P        Pmax-P    Cnstr  Par_Name
 1   4.9737e-01   2.3601e-03                              SIGN+  Amplitude
 2  -1.8164e+00   8.1665e-03                              SIGN-  Pwrlaw Index
 3   1.0000e-01                                           FIXED  Normalizing Energy
```

Figure 4.1: A Spectrum of the Crab Nebula & pulsar from Viewing Period 221. This was performed with the expanded response matrix.

```
   6.905919e-02   3.071807e-02   7.201910e-01   1.284941e-03
   9.977726e-02   4.433932e-02   3.891529e-01   1.063650e-03
   1.441166e-01   5.824104e-02   1.992370e-01   6.661410e-04
   2.023576e-01   9.044579e-02   9.211867e-02   4.396134e-04
   2.928034e-01   1.284606e-01   4.004344e-02   4.065113e-04
   4.212640e-01   1.863276e-01   1.845548e-02   3.678837e-04
   6.075916e-01   2.640166e-01   8.496242e-03   2.870063e-04
   8.716082e-01   3.596396e-01   3.656742e-03   2.816125e-04
   1.248917e+00   5.628425e-01   1.592188e-03   1.803348e-04
   1.811760e+00   7.825322e-01   5.771935e-04   1.376720e-04
   2.594291e+00   1.125891e+00   3.194376e-04   7.037019e-05
   3.720183e+00   1.622687e+00   1.388980e-04   3.919802e-05
   5.342870e+00   2.302968e+00   4.960678e-05   1.742393e-05

OD10:[GIMGR.GI_GUIDE_DEV]CRABPULSAR_SOFT_PHOT.DAT;1

;    [Mon Aug 14 14:35:35 1995]
;columns: 1 = E_low(MeV), 2 = Width(MeV), 3 = Flux(pho/MeV/cm2/s),   4 = Sigma
15     rows
4          columns
   3.532876e-02   1.239002e-02   2.315879e+00   7.661720e-03
   4.771877e-02   2.134042e-02   1.287981e+00   3.695189e-03
   6.905919e-02   3.071807e-02   7.202391e-01   1.285031e-03
   9.977726e-02   4.433932e-02   3.891836e-01   1.063726e-03
   1.441166e-01   5.824104e-02   1.991939e-01   6.659965e-04
   2.023576e-01   9.044579e-02   9.205822e-02   4.393271e-04
   2.928034e-01   1.284606e-01   4.003363e-02   4.064146e-04
   4.212640e-01   1.863276e-01   1.842380e-02   3.671801e-04
   6.075916e-01   2.640166e-01   8.467456e-03   2.859893e-04
   8.716082e-01   3.596396e-01   3.632584e-03   2.794470e-04
   1.248917e+00   5.628425e-01   1.572489e-03   1.779186e-04
   1.811760e+00   7.825322e-01   5.666818e-04   1.351443e-04
   2.594291e+00   1.125891e+00   3.122509e-04   6.881129e-05
   3.720183e+00   1.622687e+00   1.357605e-04   3.831739e-05
   5.342870e+00   2.302968e+00   4.871346e-05   1.710554e-05
```

4.2.4 Example 4: Combining Separate Observations

It often happens that an OSSE observation of a given target may be split into two or more parts. This can occur due to interruption by a target of opportunity, a scheduling constraint, or any one of a number of other possibilities. If the observations are not greatly separated in time, it can be desirable to concatenate the multiple observations into a single file. An excellent example for this is a Phase 2 observation of the QSO 3C 273 which was observed across three different viewing periods: 204, 205, and 206. The observation actually shared some of the OSSE pointings with 3C 279.

We'll begin this exercise by assuming you already have generated the high level products for the targets and three viewing periods in question, either by downloading them directly from the archive or generating them yourself in IGORE:

```
Igore> specanal,target='3C273',vp=204.0
Igore> specanal,target='3C273',vp=205.0
Igore> specanal,target='3C273',vp=206.0
```

(In reality, if you have full access to the low-level data products, it is more worthwhile to run SPECANAL over the entire time period in question for the desired target. For this exercise, we will assume it is most convenient to use the high-level products.)

If you are only interested in the time-averaged spectrum accumulated over the entire time of observation, we can utilize the total spectra (by detector) and response matrices in the *TOT.SDB files. Basically, we will extract the SDRs in each *TOT.SDB file and sum them together before passing the result to FITIT. The listing below illustrates this process.

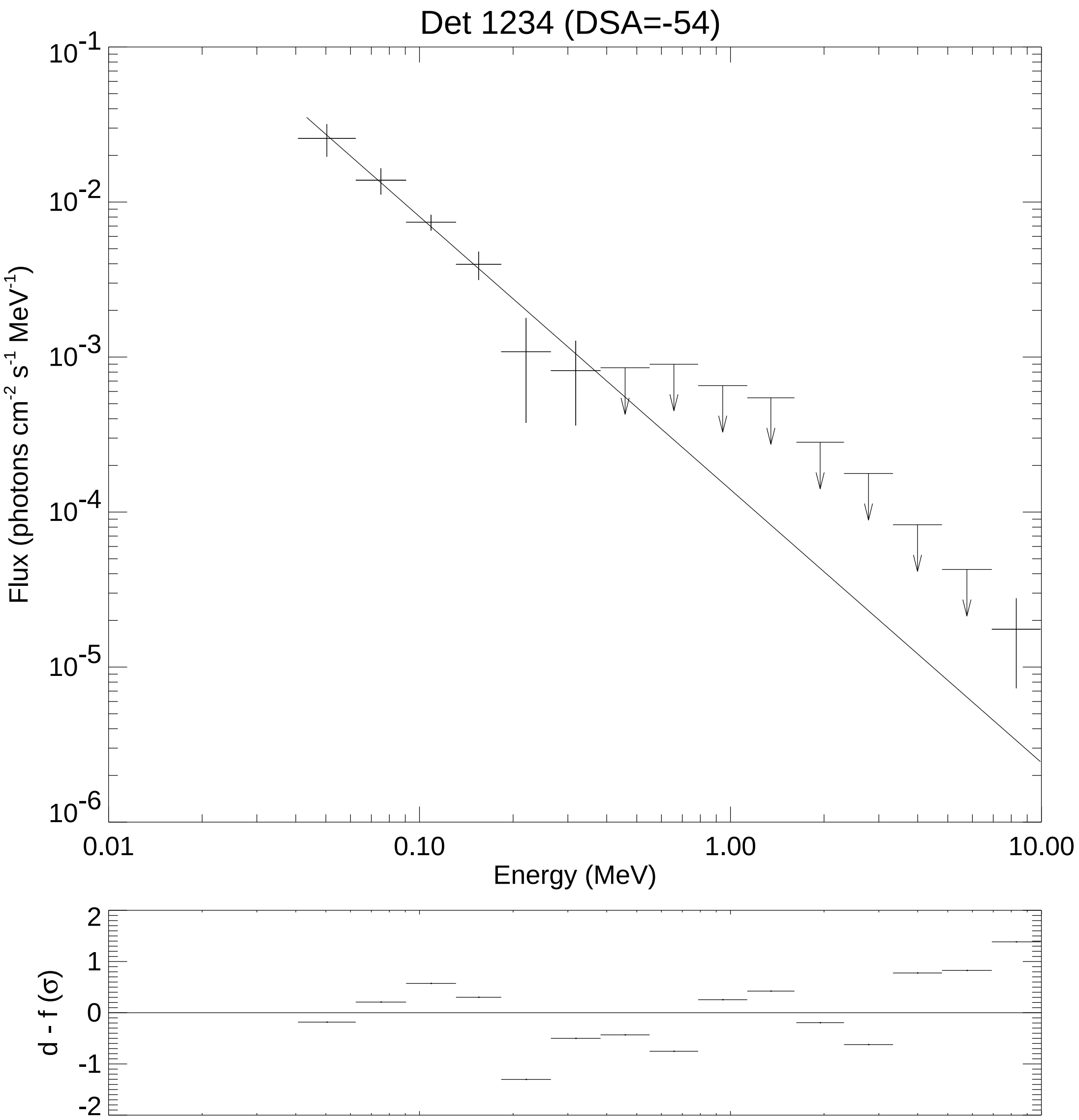
```
; extract the SDRs from the files desired
Igore> sdr_load,'3C273_204_TOT.SDB',sdrvp204
Igore> sdr_load,'3C273_205_TOT.SDB',sdrvp205
Igore> sdr_load,'3C273_206_TOT.SDB',sdrvp206

; now combine them into a single set of SDRs by detector
;  and keep the response matrices
Igore> sdr_sum,[sdrvp204,sdrvp205,sdrvp206],$
             sdrtot,$              ; output SDR
             /bydet,$              ; summing by detector
             /matrix               ; and retaining response matrices

; now generate a fit to a power-law model
Igore> status=fitit(sdr=sdrtot,$        ; input SDR
Igore>        modelfile='pwrlaw.mod',$  ; model name
Igore>        fileroot='3C273_pwrlaw_bydet')  ; saving files to...
```

This will generate a plot and fit for each detector used in the observation and a totaled spectra. For the above exercise, we have assumed a power-law model template, pwrlaw.mod, has already been generated.

Suppose were not interested in the spectra for each detector which is generated by FITIT. In this case, simply sum all the detectors using SDR_SUM and call FITIT again:

```
; now combine them into a single SDR
;  and keep the response matrice
Igore> sdr_sum,[sdrvp204,sdrvp205,sdrvp206],$
             sdrtot,$              ; output SDR
             /matrix               ; and retain response matrix

; now generate a fit to a power-law model
Igore> status=fitit(sdr=sdrtot,$        ; input SDR
Igore>        modelfile='pwrlaw.mod',$  ; model name
Igore>        fileroot='3C273_pwrlaw')  ; saving files to...
```

A sample output from this process is presented in Figure 4.2.




```
1995 Aug  8 17:39
Config: 3C273_PWRLAW_CFG.FIT      Model: 3C273_PWRLAW.MDL
SDR_SUM: Det 1234    92/357 16:07:18 - 93/012 11:14:44 Accum:   216694.8 sec
SDR_SUM: Data from 1 targets, 2 VPs
SDR_SUM: 3C 273 VP 204 Det 1234
SDR_SUM: 3C 273 VP 205 Det 1234
   ChiSqr/Dof:  0.992  NDof =    446     (Prob =  0.539435)
   En Range:    0.041 -  9.944 MeV
   Powerlaw Function for 3C 273
No       P            dP          Pmin-P        Pmax-P   Cnstr  Par_Name
 1   8.0509e-03   6.5352e-04                             SIGN+  Amplitude
 2  -1.7608e+00   1.5497e-01                             SIGN-  Pwrlaw Index
 3   1.0000e-01                                          FIXED  Normalizing Energy
```

Figure 4.2: Power Law Spectral fit for 3C 273 in periods 204, 205 & 206.

Suppose you wish to check this extended observation for time variability. In this case, we wish to keep the individual days separated by detector (to ensure the variability is consistent across each detector). In this case, we can accumulate the SDRs from the Good-Summed SDB file and send them through PLT_ENE_BANDS which generates the EBAND output file from SPECANAL. Here's how to do it:

```
; extract SDRs from the files in desired
Igore> sdr_load,'3C273_204_GSM.SDB',sdrvp204
Igore> sdr_load,'3C273_205_GSM.SDB',sdrvp205
Igore> sdr_load,'3C273_206_GSM.SDB',sdrvp206
```

Now we have a problem. We could combine the SDRs into one long array by the simplest means possible, i.e.:

```
; combine them into a single SDR
Igore> sdr=[sdrvp204,sdrvp205,sdrvp206]
```

However, this immediately runs into problems. The transition from one viewing period to the next doesn't occur at some nice convenient time, like midnight, GMT. As a result, there are common days between viewing period 204 and 205. This is easy to illustrate by examining the starting time of the SDRs:

```
print,getfld('hdr.strttim',sdrvp204)/86400.0
;       8978.67      8978.86      8978.67      8978.68      8979.04      8979.04
;       8979.05      8979.04      8980.06      8980.00      8980.00      8980.00
;       8981.02      8981.02      8981.03      8981.02      8982.04      8982.04
;       8982.05      8982.04      8983.00      8983.00      8983.01      8983.06
;       8984.02      8984.02      8984.02      8984.02      8985.04      8985.04
;       8985.04      8985.04
print,getfld('hdr.strttim',sdrvp205)/86400.0
;       8985.82      8985.81      8985.81      8985.81      8986.01      8986.06
;       8986.00      8986.02      8987.04      8987.08      8987.04      8987.04
;       8988.04      8988.04      8988.04      8988.04      8989.06      8989.06
;       8989.12      8989.06      8990.02      8990.01      8990.02      8990.01
;       8991.04      8991.04      8991.04      8991.04      8992.01      8992.02
;       8992.01      8992.01
```

where the division by 86400.0 converts the value to truncated Julian days. It's easy to see that the last four entries in VP 204 are on the same day as the first four entries in VP 205. There are four entries for each day corresponding to each of the four OSSE detectors used in this observation.

What do we do? We prefer to combine the last four SDRs in VP 204 with the first four SDRs in VP 205, making them into a single day. Similarly, we find that the last four SDRs in VP 205 are on the same day as the first four SDRs in VP 206. We finally want to merge the entire set into a single array of SDRs, properly separated by detector and sorted by day. Simple enough:

```
; merge the connecting day for VP 204 & 205
sdr_sum,[sdrvp204(28:31),sdrvp205(0:3)],sdrda,/bydet

; merge the connecting day for VP 205 & 206
sdr_sum,[sdrvp205(28:31),sdrvp206(0:3)],sdrdb,/bydet

;and now slip those outputs into the appropriate sequence...
sdrlc2=[sdrvp204(0:27),sdrda,sdrvp205(4:27),sdrdb,sdrvp205(4:*)]
```

An finally we plot the results.

```
; now plot them in the 0.05-0.15 MeV energy band
Igore> bandedge=0.05
Igore> bandwidth=0.10
Igore> plt_ene_bands,'',bandedge,bandwidth,insdr=sdrlc2
```

A sample output of the resulting light-curve is presented in Figure 4.3.

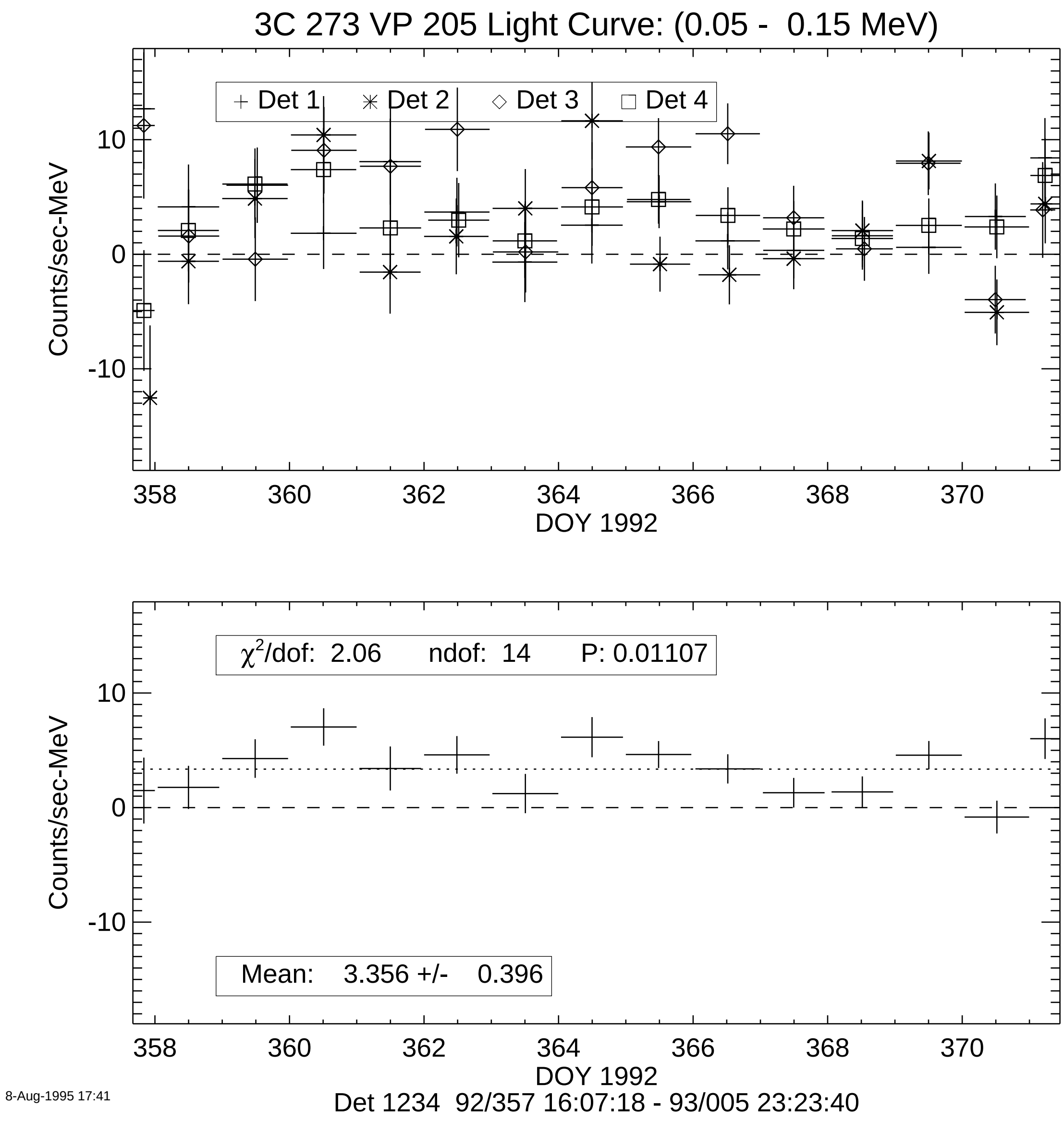


Figure 4.3: Light Curve at 0.05-0.15 MeV for 3C 273 in periods 204, 205 & 206.

Detector	χ^2/dof vs Avg. of Other Dets
1	1.60 (ndof: 14; Prob: 0.07079)
2	2.51 (ndof: 14; Prob: 0.00141)
3	2.11 (ndof: 14; Prob: 0.00870)
4	0.49 (ndof: 14; Prob: 0.94065)

Chapter 5

Advanced Exercises

5.1 User Examples

5.1.1 Example 1: Processing a Spectrum for Each Orbit

Imagine you have a really bright source that varies hourly (okay, okay, I did say *imagine*) and you wish to generate spectra on timescales less than the daily totals that are stored in the Good-Sum (GSM) SDB file but longer than the 2-minute sums stored in the daily difference (DIF) SDB files. Can it be done? Absolutely!!

Given a bright source like Cygnus X-1, a Galactic X-ray nova, or (oh, please, please, please!) a Galactic supernova. it can be practical to generate spectra on time-scales as short as a single orbit.

Our first goal is to create an SDB file summed by orbit, rather than by day, as is normally performed by DAY_SUMMER. Another important issue is that for part of the observation we wish to consider (Viewing Period 15), the OSSE detectors were set at x2 gain. Therefore we need to keep this data distinguished from the x1 gain observations by separating the instrument command states.

To setup for running SPECANAL, we first need to specify the special commands for the DAY_SUMMER which will be in the file CYGX-1_15_ORBIT.DSM:

```
; %NameList% DAYSUM_STRUC % Structure created  5-Sep-1995 16:53:00.00
OUTFILE = "CYGX-1_15_DSM.SDB"
GOOD_SUMFILE = "CYGX-1_15_GSM.SDB"
PROD_DIR = "$PROD:"
ORBIT =        1
START_TIME = "91/332 12:55:30"
STOP_TIME = "91/346 16:33:40"
BYCMDSTATE =        1
/END ; of setup element definition
```

The flags ORBIT and BYCMDSTATE are set to 1 to generate separate SDRs (still separated by detector) for each orbit and instrument command state. The names of both the Day-Summed (DSM) and Good- Summed (GSM) files are specified.

We also need to inform FFIT_PREP how to total the spectra and link in the response matrices. This is done in the file CYGX-1_15_ORBIT.FP:

```
; %NameList% FIT_PREP_STRUC % Structure created  5-Sep-1995 16:54:24.00
SDBFILE(0) = "CYGX-1_15_GSM.SDB",
BYDET =        1
DAILY =        1
DAYFILE = "CYGX-1_15_DAY.SDB"
OUTFILE = "CYGX-1_15_TOT.SDB"
/END ; of setup element definition
```

where DAILY=1 specifies to generate the *DAY.SDB file which contains links to the response matrices in the OUTFILE. Note we still want the SDRs separated by detector (BYDET=1).

At this point, we have two possible methods for invoking SPECANAL. One way is to directly call the routine and use keywords to specifiy the new parameter files:

```
Igore> specanal,target='CYG X-1',vp=15.0,$
Igore>     fileroot='cygx-1_15_orbit',$
Igore>     daysum_config='CYGX-1_15_ORBIT.DSM',$
Igore>     fitprep_config='CYGX-1_15_ORBIT.FP',$
Igore>     /nobln,/noplotanal,$
Igore>     que='sys$ossep'
```

where we have also used the NOBLN and NOPLOTANAL keywords to skip running the PLOTANAL and BOTTOMLINE routines (why waste CPU time and disk space when we'll be generating our own products which replace them).

Alternatively we can use an analysis setup file which we will name CYGX-1_15_ORBIT.ANL:

```
; %NameList% SPECANAL_STRUC % Structure created  5-Sep-1995 17:05:47.00
CFG_FILE = "OGRE$DAT:nominal_cfg.anl"
TARGET = "{NAME: 'CYG X-1', RA:  299.590, DEC:   35.202, SRCID: 120 }"
VP =       15.0000
START_TIME = "91/332 12:55:30"
STOP_TIME = "91/346 16:33:40"
OUT_DIR = "ODT5:[BRIDGMAN.GUIDE_SAMPLES.OPTIONS]"
FILEROOT = "CYGX-1_15_orbit"
LOGFILE = "CYGX-1_15_ORBIT.LOG"
BQD_CONFIG = "IGR:[DAT]NOM_CFG.BQD;1"
BQD_DIR = "ODT5:[BRIDGMAN.GUIDE_SAMPLES.OPTIONS]"
DAYSUM_DIR = "ODT5:[BRIDGMAN.GUIDE_SAMPLES.OPTIONS]"
DAYSUM_OUT = "CYGX-1_15_DSM.SDB"
DAYSUM_GOOD = "CYGX-1_15_GSM.SDB"
```

```
DAYSUM_CONFIG = "ODT5:[BRIDGMAN.GUIDE_SAMPLES.OPTIONS]CYGX-1_15_ORBIT.DSM"
FITPREP_OUT = "CYGX-1_15_ORBIT_TOT.SDB"
FITPREP_DIR = "ODT5:[BRIDGMAN.GUIDE_SAMPLES.OPTIONS]"
FITPREP_CONFIG = "ODT5:[BRIDGMAN.GUIDE_SAMPLES.OPTIONS]CYGX-1_15_ORBIT.FP"
NOPLOTANAL =        1
PLOTANAL_DIR = "ODT5:[BRIDGMAN.GUIDE_SAMPLES.OPTIONS]"
PLOTANAL_CONFIG = "IGR:[DAT]NOM_CFG.PA;1"
NOBLN =        1
BLN_OUT = "CYGX-1_15_ORBIT_BLN.SDB"
BLN_DIR = "ODT5:[BRIDGMAN.GUIDE_SAMPLES.OPTIONS]"
BLN_CONFIG = "IGR:[DAT]NOM_CFG.BLN;3"
/END ; of setup element definition
```

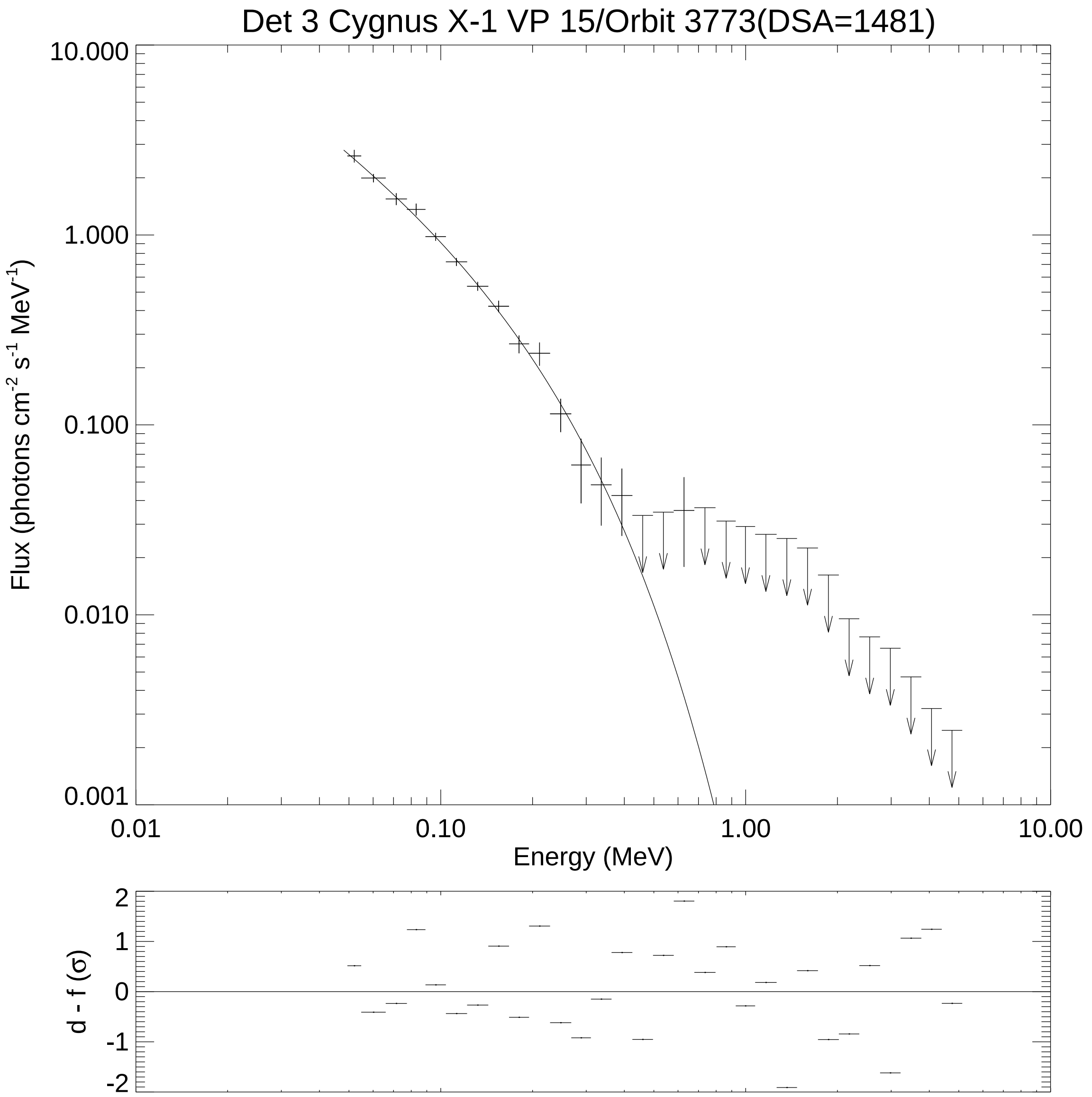
where keywords DAYSUM_CONFIG and FITPREP_CONFIG point to the respective parameters files and the keywords NOPLOTANAL and NOBLN are set to 1, indicating that we don't wish to generate this output. We start the job executing with a command like:

```
Igore> specanal,'CYGX-1_15_ORBIT.ANL',que='sys$ossep'
```

Once we have the *DAY.SDB file, we can generate multiple model fits. For this particular example, we create a relativistic thermal bremsstrahlung model file RTBREM.MOD using BUILDMODEL. Then we generate the fits for each orbit using an IGORE program like the one below. Figure 5.1 is a sample of one of the fits.

```
Igore> sdr_load,'CYGX-1_15_DAY.SDB',sdr
Igore> startmodel=0
Igore> ; get orbit numbers
Igore> orbit=getfld('hdr.orbit',sdr)
Igore> orbits=uniquevals(orbit)    ; find the number of unique orbit entries
Igore> detid=getfld('hdr.detid',sdr)
Igore> print,'Records found: ',n_elements(orbit)
Igore> ; loop through each unique orbit
Igore> norbits=n_elements(orbits)
Igore> print,'Orbits to process: ',norbits
Igore> for i=startmodel,norbits-1 do begin
Igore>    dets=where(orbit eq orbits(i))   ; collect detectors for each orbit
Igore>    print,'Processing Orbit: ',orbits(i),'/Dets:',detid(dets)
Igore>    ; generate appropriate fit
Igore>    orbitnum='00000'+strtrim(string(orbits(i)),2)
Igore>    outlen=4
Igore>    num=strmid(orbitnum,strlen(orbitnum)-4,4)
Igore>    x=fitit(file='setup_fitit.txt',$    ; minimal setup file defined to
Igore>                                        ; suppress prompt
Igore>         sdr=sdr(dets),fitmodel='cygfit_'+num+'.mdl',$
Igore>         title='Cygnus X-1 VP 15/Orbit '+num)
Igore> endfor
```

For the above example, we have created a small setup file with default parameters to supress the setup file prompt that FITIT generates. The contents of SETUP_FITIT.TXT are listed below:




```
1995 Sep  7 16:18
Config: CYGX1_15_3773_CFG.FIT      Model: CYGFIT_3773.MDL
SDR_SUM: Det 3    91/339 16:06:05 - 91/339 16:17:00 Accum:      262.1 sec
   ChiSqr/Dof:  0.912  NDof =    437     (Prob =  0.906615)
   En Range:    0.047 -  5.152 MeV
   Relativisitic Bremsstrahlung Function for CYG X-1
No       P            dP          Pmin-P        Pmax-P   Cnstr  Par_Name
 1   8.0311e-01   1.5868e-02                             SIGN+  Amplitude
 2   1.0000e-01                                          FIXED  Normalizing Energy
 3   1.2051e-01   6.0224e-03                             SIGN+  kT
```

Figure 5.1: The Spectrum of Cygnus X-1 integrated over one orbit.

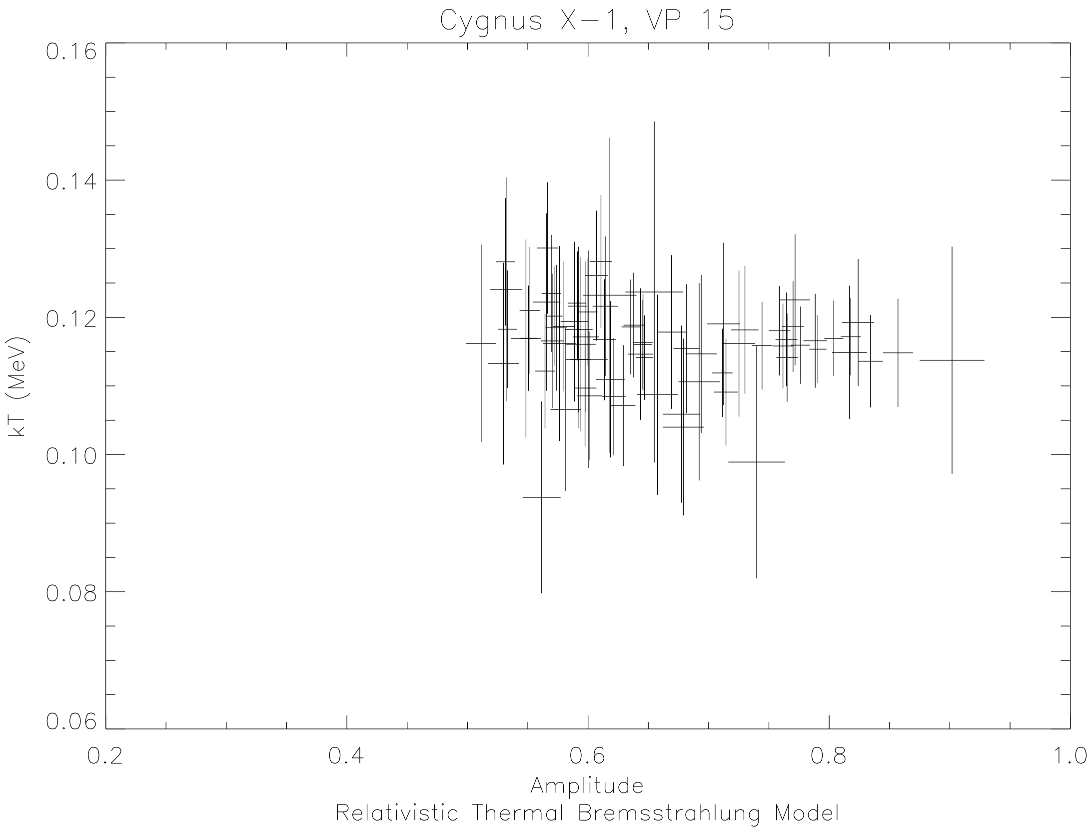


Figure 5.2: Amplitude vs. Temperature for Cygnus X-1 assuming a Relativistic Thermal Bremsstrahlung model (variation plotted over 75 orbits).

```
modelfile="rtbrem.mod"
plot_nchan=30
\end
```

Now that we have the data, what can we do with it? For starters, we might wish to examine how the model fits vary with time (do temperature and amplitude have a well-defined relationship with each other?) Below is a simple IGORE program to read the model files generated above and plot the amplitude vs. temperature (Figure 5.2).

```
Igore> modfiles=findfile('cygfit_*.mdl')
Igore> npts=n_elements(modfiles)
Igore> a=fltarr(npts) & da=fltarr(npts)
Igore> kt=fltarr(npts) & dkt=fltarr(npts)
Igore> ; now loop through the model files
Igore> for j=0,npts-1 do begin
Igore>         print,'Reading file: ',modfiles(j)
Igore>         ms=model_read(modfiles(j))
Igore>         a(j)=ms.p(0) ; get amplitude
Igore>         da(j)=ms.dp(0)
Igore>         kt(j)=ms.p(2) ; get temperature
```

```
Igore>          dkt(j)=ms.dp(2)
Igore> endfor
Igore> ; now generate the appropriate plots
Igore> binplot,a-da,2.0*da,kT-dkT,2.0*dkT,/ii,$
Igore>          xtitle='Amplitude',ytitle='kT (MeV)',title='Cygnus X-1, VP 15',$
Igore>          subtitle='Relativistic Thermal Bremsstrahlung Model'
```

5.1.2 Example 2: The Problem of x2 Gain

Back in Example 3 of Chapter 4, we performed an analysis of the Crab spectrum, with particular emphasis on handling the pecularities of it being a relatively hard source compared to many objects observed with OSSE. Through the analysis, we obtained a 'reasonable' spectrum but some might have wondered why the spectrum seemed harder than expected, and had that large break away from the power-law fit. In fact, there is indeed a problem in that spectrum and it is one I demonstrate how to detect and fix in this example.

To see just how serious the problem is, let's take a closer look at the spectrum generated for that example. Let's see how it looks with smaller energy channels such as in Figure 5.3

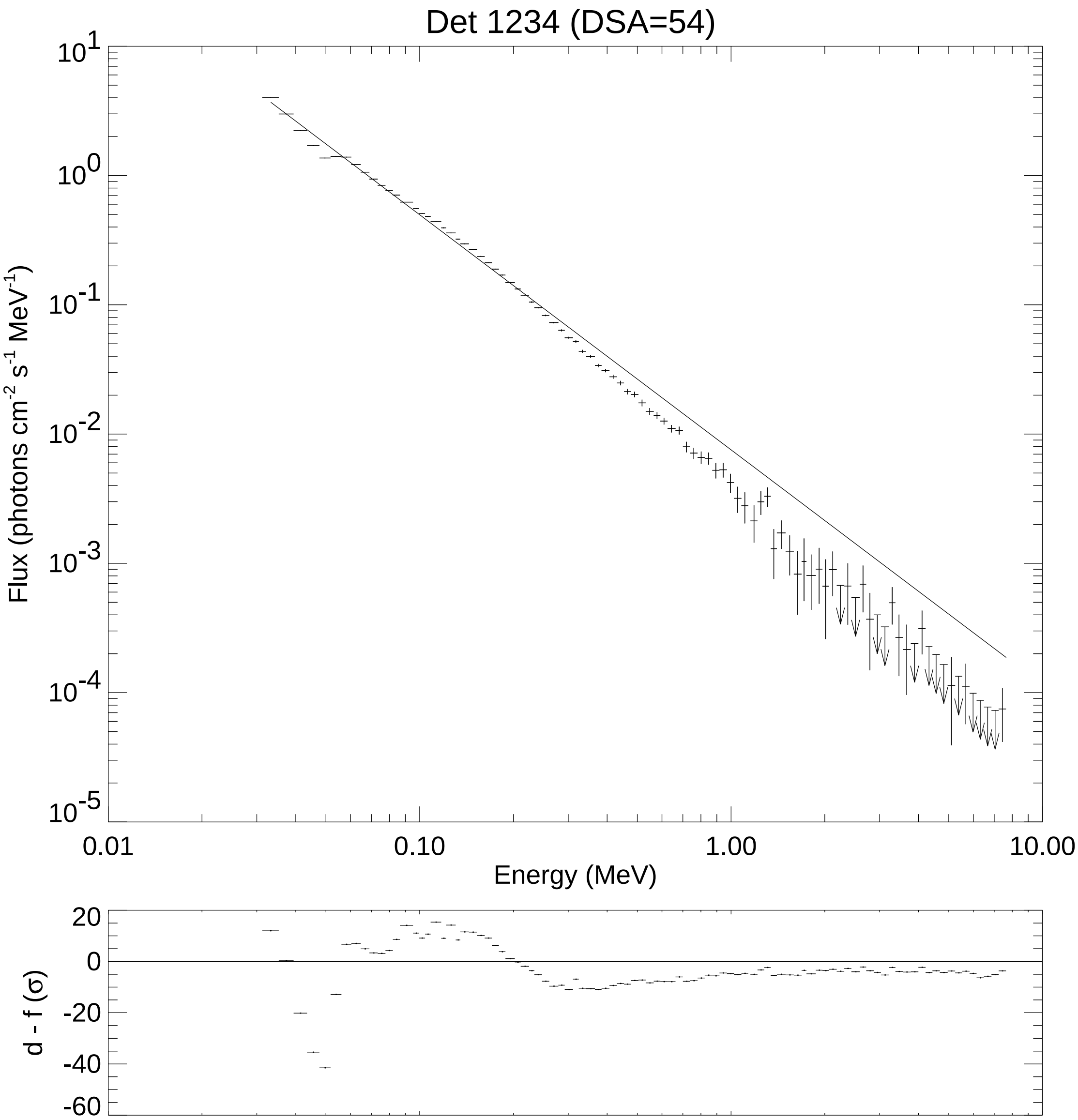
The "kink" in the spectrum near 50 keV is not real. It is an instrument artifact due to a change in the detector settings during the observation.

OSSE has a standard, normal gain (called "x1 gain") which provides energy channels 6 keV wide below 1.5 MeV and 40 keV wide from 1.5 to 10 MeV. Below 50 keV, electronic noise contributes to the spectra. In twice-normal gain ("x2 gain") the energy channels are 3 keV wide below 750 keV and 20 keV wide from 750 keV to 5 MeV and electronic noise doesn't appear until below about 40 keV. However, below this energy, the OSSE instrument itself has absorbed almost all the incident flux. Times-two gain is very useful in observing soft sources.

IGORE performs most of its spectral manipulation operations in count-space and photon energy doesn't come into the equation until FITIT is performed. Prior to this, spectra are added channel by channel. The problem arises because in normal gain, channel 20 may correspond to an energy of 78 keV with a width of 6 keV. In x2 gain, channel 20 has an energy of 37 keV and a width of 3 keV. The cause of the problem is that you are adding two spectra, one translated and stretched with respect to the other. The solution is to treat the two different gain settings separately and only add them after they have been converted into photon- space.

First we'll gather together the normal gain data and generate a spectrum. Here is an IGORE program to do it:

```
Igore> sdr_load,'crabpulsar_221_gsm.sdb',sdr
Igore> gain=getfld('hdr.rgain',sdr)
Igore> tags=getfld('hdr.tagword',sdr)

Igore> exclude=tags(where(gain gt 1.5))  ; make an array of tags of x1 gain

Igore> ; regenerate response matrices with new range & exclude x2 gain
```




```
1995 Sep 13 16:57
Config: CRAB_HIRES_CFG.FIT      Model: CRAB_HIRES.MDL
SDR_SUM: Det 1234    93/133 18:23:16 - 93/144 11:18:01 Accum:   228081.7 sec
SDR_SUM: CRAB PULSAR VP 221 Det 1234
   ChiSqr/Dof: 18.697  NDof =    446     (Prob =  0.000000)
   En Range:    0.031 -  7.666 MeV
   Powerlaw Function for CRAB PULSAR
No        P            dP          Pmin-P        Pmax-P    Cnstr  Par_Name
 1   4.9715e-01   2.3669e-03                               SIGN+  Amplitude
 2  -1.8182e+00   8.1811e-03                               SIGN-  Pwrlaw Index
 3   1.0000e-01                                            FIXED  Normalizing Energy
```

Figure 5.3: A Spectrum of the Crab Nebula & pulsar from Viewing Period 221. This was performed with the expanded response matrix and plotted with 100 logarithmically-spaced energy channels. The 'kink' in the spectrum near 50 keV is *not* real! It is an instrument artifact.

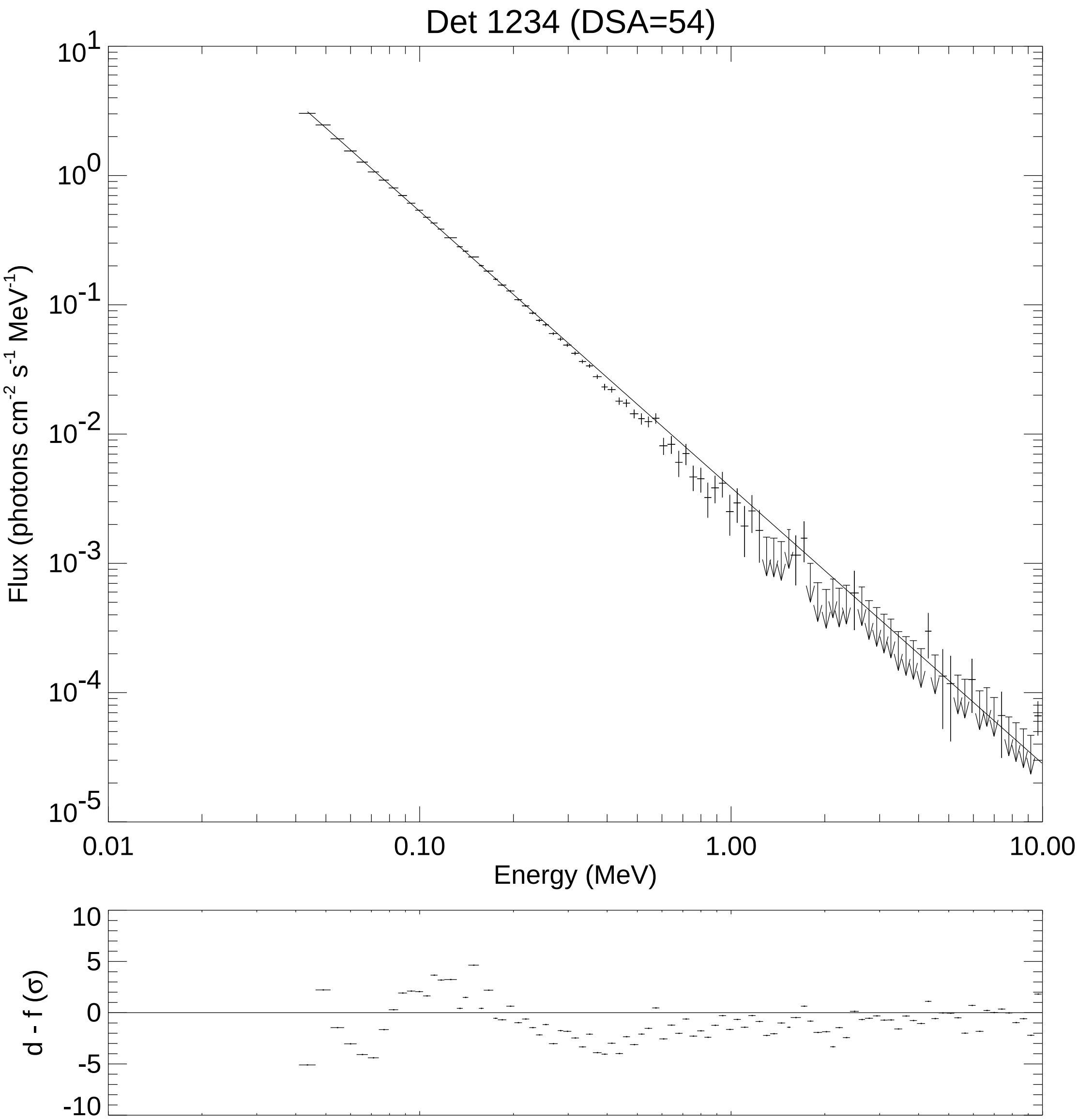
```
Igore> stat=ffit_prep(fileroot='CRABPULSAR_221x1',$
Igore>         sdbfile='CRABPULSAR_221_GSM.SDB',$
Igore>         Not_tags=exclude,$
Igore>         filesetup='NONE',$
Igore>         mtrx_nphot=600,$
Igore>         mtrx_prange=[0.02,20.0])

Igore> ; now reload it and fit it
Igore> sdr_load,'crabpulsar_221x1_tot.sdb',sdr
Igore> sdr$create,r_out,1
Igore> sdr_sum,sdr,r_out,/matrix              ; add them up
Igore> stat=fitit(fileroot='CRABPULSAR_221x1',$
Igore>         sdr=r_out,plot_nchan=100,$
Igore>         modelfile='PWRlaw.MOD')
```

The results appear in Figure 5.4.

Next we select out the x2 gain data and fit it. We'll use a different method for extracting the data. It turns out that the EXCLUDE keyword in FFIT_PREP can only handle up to 30 tags for exclusion. For this next step, there are more than 30 SDRs which should be excluded, so instead we create a new TOT.SDB file which contains only x2 gain data.

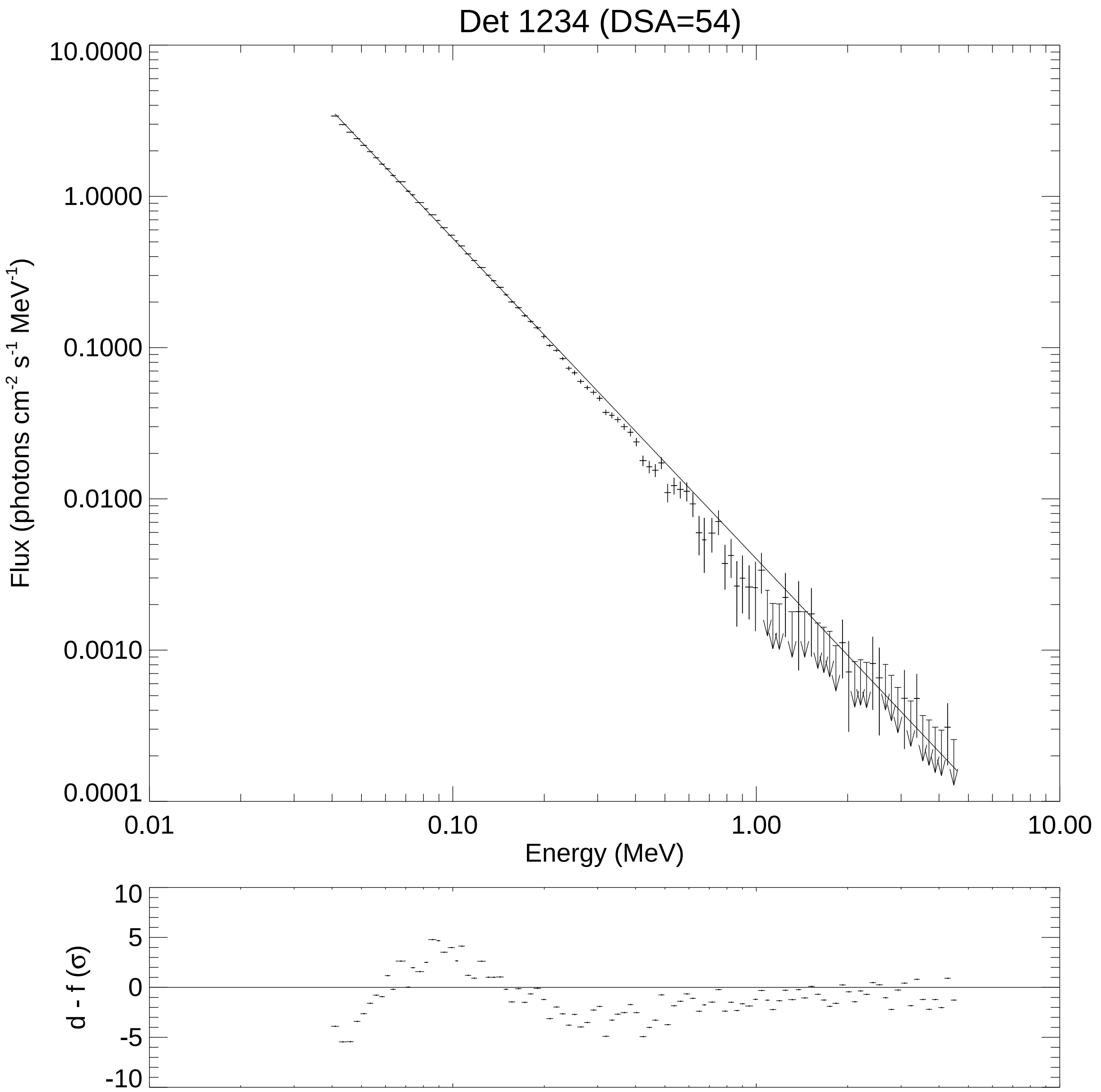
```
Igore> sdr_load,'crabpulsar_221_gsm.sdb',sdr,plus='crc,note'
Igore> gain=getfld('hdr.rgain',sdr)
Igore> index=where(gain gt 1.5) ; select x2 gain elements
Igore> write_specsdr,sdr(index),'Crab_tempx2_gsm.sdb'

Igore> ; regenerate response matrices with new range & only x2 gain
Igore> stat=ffit_prep(fileroot='CRABPULSAR_221x2',$
Igore>         sdbfile='CRAB_tempx2_GSM.SDB',$
Igore>         filesetup='NONE',$
Igore>         mtrx_nphot=600,$
Igore>         mtrx_prange=[0.02,20.0])

Igore> ; now combine the detectors
Igore> sdr_load,'crabpulsar_221x2_tot.sdb',sdr
Igore> sdr$create,r_out,1
Igore> sdr_sum,sdr,r_out,/matrix              ; add them up

Igore> ; fit the result
Igore> stat=fitit(fileroot='CRABPULSAR_221x2',$
Igore>         sdr=r_out,plot_nchan=100,$
Igore>         modelfile='PWRlaw.MOD')
```

The results appear in Figure 5.5.




```
1995 Sep 21 15:11
Config: CRABPULSAR_221X1_CFG.FIT      Model: CRABPULSAR_221X1.MDL
SDR_SUM: Det 1234    93/133 18:23:16 - 93/139 13:41:13 Accum:   129482.8 sec
SDR_SUM: CRAB PULSAR VP 221 Det 1234
   ChiSqr/Dof:  1.663  NDof =    446     (Prob =  0.000000)
   En Range:    0.041 -  9.979 MeV
   Powerlaw Function for CRAB PULSAR
No       P            dP          Pmin-P        Pmax-P    Cnstr  Par_Name
 1   5.2873e-01   9.7398e-04                              SIGN+  Amplitude
 2  -2.1360e+00   4.0352e-03                              SIGN-  Pwrlaw Index
 3   1.0000e-01                                           FIXED  Normalizing Energy
```

Figure 5.4: A Spectrum of the Crab Nebula & pulsar from Viewing Period 221 selecting normal gain data only. This was also performed with the expanded response matrix and plotted with 100 logarithmically-spaced energy channels. Note that the 'kink' is gone.




```
1995 Sep 21 14:57
Config: CRABPULSAR_221X2_CFG.FIT      Model: CRABPULSAR_221X2.MDL
SDR_SUM: Det 1234    93/139 17:45:49 - 93/144 11:18:01 Accum:    98598.9 sec
SDR_SUM: CRAB PULSAR VP 221 Det 1234
   ChiSqr/Dof:  2.040  NDof =    437     (Prob =  0.000000)
   En Range:    0.040 -  4.599 MeV
   Powerlaw Function for CRAB PULSAR
No       P           dP          Pmin-P        Pmax-P    Cnstr  Par_Name
 1   5.2754e-01   1.1587e-03                             SIGN+  Amplitude
 2  -2.1185e+00   4.4255e-03                             SIGN-  Pwrlaw Index
 3   1.0000e-01                                          FIXED  Normalizing Energy
```

Figure 5.5: A Spectrum of the Crab Nebula & pulsar from Viewing Period 221 selecting twice-normal gain data only. This was also performed with the expanded response matrix and plotted with 100 logarithmically-spaced energy channels. Note that the 'kink' is also gone here.

Chapter 6

Reference Guide

DAY_SUMMER

Process OSSE SDB difference files, summing into daily average spectra. Gain information is obtained from difference SPM files and energy scales are applied to the resultant sums. Optional user screening can be applied. An SDB sum record in the output file is created for each DetConfig, detector and DSA (scan angle) combination for each input file. Optional reprocessing of differences can be performed to produce statistically correct error estimates, correcting for multiple use of individual backgrounds.

Keyword support permits optional division of day into "orbital" periods.

NOTE: live-time weighted sums are performed here.

Calling Sequence

DAY_SUMMER, InFiles, Outfile, Status [,keywords]

INPUTS

InFiles
string array of SDB difference files to be summed. There is one SDB record per detector per DSA per infile in the output SDB file.

Outfile
String identifying output file specification for resultant sums. A new file is created or sums appended to existing file depending on parameter Status If outfile is a zero length string or all blanks, no standard sum file will be created. (SEE keyword, Good_sumfile, below. If this keyword is not set and outfile is blank, then you get nothing. What a waste of time.)

Status
Character string identifying output file status. This should specify "NEW" for creation of new output file or "OLD" to append sums to existing file. If status parameter not specified, "NEW" is assumed.

KEYWORDS

E_Range
Two element array specifying the energy range to use in doing standard χ^2 screening of individual 2-minute differences. Default range is [0.045,0.25]. This spec should be within the low range. If specified, will override keyword Chan_Range. Routines convert this to channel range based on the gain of the detectors.

Chan_Range
Two element array specifying the range of channels to use in doing standard χ^2 screening of individual 2-min differences. Default range is [14,40]. Obsolete, use keyword E_Range instead.

Good_sumfile
ascii string containing an SDB file specification for the output of difference sums with correct statistics propagation. There is a one to one correspondence between tags in this file and tags in Outfile. NOTE: If this keyword is not specified, correct statistics are not computed. NOTE:NOTE: Computation of the correct statistics requires reopening of original SPM files and is a time consumming process.

Thresh
input spec of χ^2 probability threshold ($0 < \text{thresh} < 1$) for rejecting bad 2-min differences as determined by deviations from average or template. Default threshold is 0.9

NoDSA
logical switch, if set, the default separating of of sums by DSA is disabled. All SDRs from a given detector summed together for a day.

NoStandard
logical switch, if set, no standard screening is performed. If Userscreen is specified, user screening will be performed, else no screening is performed. Default is perform standard screening.

Orbit
logical switch, if set, Data are summed into orbital time sums. Approximately 16 sums per day per detector are created. Sums are synchronized to continuous periods of target visibility (boundaries of sums are the centers of occultations). This overrides default daily sums. NOTE: orbital sums can bridge day boundaries. For correct operation, the file names require the nominal "date root" format so that a filename sort will put them in chronological order.

Prod_Dir
string specifying the directory for the input files. This input overrides the default

specification of "$PROD" for access to the lower level SPM files which are used in computing the correct statistical ouput file.

Start_Time
string specifying the starting time for the sum(s). Format is yy/ddd hh:mm:ss. This overrides the default action of starting with the first SDR found in the input file list.

Stop_Time
string specifying the stopping time for the sum(s). Format is yy/ddd hh:mm:ss. This overrides the default action of stopping with the last SDR found in the input file list.

Userscreen
string specifying the name of a user-supplied screening routine to perform additional data selection criteria prior to summing. Default is no userscreening performed. If specified, the routine is called as a procedure with two parameters: input SDR array and returned array of indices for SDR array which passed user screening. This is approx. equivalent to :

```
Igore>  Userscreen, SDR\_Array, NDX
Igore>  if NDX(0) ge 0 then SDR$SUM, SDR\_Array(NDX), Sum\_SDR
```

summing is performed by detector and DSA, the userscreen routine is called once per detector (per file), passing SDRs for each detector in separate calls.

Userscreening is performed after standard screening or instead of standard screening, depending on keyword, NoStandard.

DSA_SCREEN
Logical switch specifying that data from each unique DSA should be screened separately.

BYCMDSTATE
Logical switch specifying that data from each unique CmdState should be screened AND summed separately. Default breaks on DetCfg_id change.

dqHaz_Mask
Input long integer specifying dqHaz bits to ignore. Default is reject data if any dqHaz bit is set (dqhaz ne 0). eg. To ignore bit 0, set dqhaz_mask = 1.

dqInfo_Mask
Input long integer specifying dqInfo bits to use in data quality screening. Default is to ignore contents of dqInfo. A bit set in the mask will cause data rejection if the corresponding bit is set in a spectrum's dqInfo status. eg. To reject data with dqinfo bit 0 set, set dqinfo_mask = 1.

dsfStat_Mask
Input long integer specifying dsfstat bits to use in data quality screening. Default

is to ignore contents of dsfstat. A bit set in the mask will cause data rejection if the corresponding bit is set in a spectrum's dqInfo status. eg. To reject data from ODS (bit 20), set dsfStat_Mask = 2^{20}.

EarthSep
Input float specifying the minimum earth separation (in degrees) for consideration as good data. Default is 72 degs - approximately the earth limb + 50 km.

Covar
Input logical switch specifying that covariance of the resulting sums should be computed and stored in the output Good_SumFile. Default is no covariance computations. This switch is functional only with Good_SumFile keyword specified.

Verbose
Input logical switch to turn on diagnostic/informational messages.

catsave
catsave

prelimsave
prelimsave

OUTPUTS

Output consists of Statistics and Errors written to the screen and summed SDB difference spectra written to specified output file. There is one SDB record in the output file for every unique cmdstate, detector and DSA (scan angle) combination in each input file. Energy scale is computed and stored with each record in sumfile. The optional Good_sumfile is identical to the Outfile but with correct errors (yes, it's an oxymoron).

FFIT_PREP

FFIT_PREP prepares spectra for forward folding analysis. It sums spectra (either by detector or not, depending on the state of BYDET, DETID), runs routines to correct for DTW efficiencies and to eliminate low-medium range overlap, then, optionally, builds a response matrix with user specified count and photon channel ranges (or it uses reasonable defaults).

Calling Sequence

status = FFIT_PREP([FILE=PSETUP] [,NOMTRX=NOMTRX])

INPUTS

No argument inputs. The setup hierarchy is (in decreasing order of precedence):

- Individual setup parameter keyword. Any setup parameter may be passed in via a keyword. Keyword names are same as setup parameter names, as described below. Setup parameters passed in in this manner take precedence over passed structures or files. Note that these keywords are NOT documented in a namelist file

  If parameter keywords are set but SETUP and FILE are not set, the function will prompt for a file spec, read in the namelist file, then override those parameters with the keyword values. Note that this mode does NOT result in a namelist file that documents the run.
- keyword FILE is set to a string containing a namelist file specification. This file will be read and a setup structure produced internally.
- FILE is not set, in which case the routine prompts the user for a file specification.

The normal modes of operation are anticipated to be 2) and 3).

Setup Variables

Title
String containing title used for count spectrum plot.

SDBfile
String containing name of the SDB file containing DAY_SUMMER results.

Not_tags
Array of tagwords to exclude from summation (limit of 30). Defaults to all tags processed.

Bydet
Flag to indicate whether detectors are summed together (0) or not (1). Defaults to 0.

Detid
Bit pick to indicate which detectors to process. Defaults to all dets.

CRC_rounder
Flag to indicate whether redundant source aspect configurations are combined using CRC_ROUNDER (1) or not (0). If used, this speeds up matrix creation. If observations with different transscan angles are combined, do not use this option. Defaults to 1.

Outfile
String containing SDB file name for output spectra. Defaults to <fileroot>_tot.sdb. If not set and no FILEROOT, or if set to 'NONE', no output file is produced.

Mtrxroot
String containing root of matrix file name. Defaults to null, in which case no matrix file will be written, but matrix will be placed in a common block.

Fileroot
String containing root name for output, log and plot files. Overriden by explicit file name entries (outfile, plotfile, logfile). If not present and explicit entry not present, file will not be produced.

Logfile
String containing name of file containing log information on run. Defaults to <fileroot>.log. If not set and no FILEROOT, or if set to 'NONE', no log file is produced.

Plotfile
String containing name of plot file. Defaults to <fileroot>.ps. If not set and no FILEROOT, or if set to 'NONE', no plot file is produced.

Mtrx_nphot
Number of photon bins in matrix to be created. Bins should be chosen so as to be small compared to resolution width. Defaults to 460 logrithmic photon channels starting at 0.04 MeV.

Mtrx_prange
Two element array containing the minimum and maximum matrix photon energies in MeV. Defaults to standard overscan of full energy range in spectrum.

Mtrx_crange
Two element array containing the minimum and maximum count spectrum energies in MeV. Spectrum and matrix are truncated to this range. Defaults to a range from 0.06 MeV to the maximum energy in the spectrum.

Keywords

Setup

Optional keyword containing a setup structure (see INPUTS for description)

File
Optional keyword containing a namelist file specification (see INPUTS for description)

Nomtrx
Flag which if set, results in no response matrix production. Default is to produce the matrix.

SDR
returned array of summed SDR - contents of *_TOT.SDB

DAYSDR
returned array of SDRs read from input SDB file.

Outputs

Output consists of an SDB file containing spectra ready to deconvolve, a plot file containing count spectrum plots, a log file detailing the run and a response matrix. See FFIT_PREP_SETUP for file naming conventions.

FITIT

FITIT performs forward folding spectrum deconvolution and analysis via DO_FOLD_ANAL. It takes as input SDB files or SDRs produced by FFIT_PREP or similar procedures and produces as output best fit models and pseudo-deconvolved spectra.

FITIT reads in prepared spectra and performs a forward folding deconvolution analysis, the result of which is a best fit spectrum model and a pseudo-deconvolved photon spectrum. Log files and plot files are produced.

Unlike previous (v.7.3 and earlier) versions, FITIT does not expect the input file to contain a detector sum spectrum in addition to the individual detector spectra.

Calling Sequence

status = FITIT([FILE = SETUPFILE] [,keywords]

INPUTS

KEYWORDS

*** Initialization and Documentation

FILE
string containing name of namelist setup file. If omitted, user will be prompted for file name.

Title
Title used for spectrum plot.

FILEROOT
String containing root name for best fit model, deconvolved spectrum, both SDB and ascii, fit routine diagnostic output, log and plot files. Overriden by explicit file name entries (FITMODEL, PHOTFILE, binfile, fitfile, logfile, and plotfile). If not present and explicit entry not present, file will not be produced.

Logfile
String containing name of file containing log information on run. Defaults to <fileroot>.log. If not set and no FILEROOT, or if set to 'NONE', no log file is produced.

Plotfile
String containing name of plot file. Defaults to <fileroot>.ps. If not set and no FILEROOT, or if set to 'NONE', no plot file is produced.

*** Input Specifications, Selection and Processing

SDBfile
String array(10) containing names of SDB files with spectra to model. These spectra must either be in Counts/second, in which case they are interpreted as OSSE spectra, or $Photons/sec/cm^2$, in which case they are interpreted as additional data (ie data from other instruments).

SDR
Optional list of SDRs to be used as input in lieu of SDB file specified in SETUP.

Add_data
String array(10) containing file names of files containing additional data to be fit along with OSSE data. Data are assumed to be in $photons/cm^2 s$ unless units are encoded otherwise (see below). A unit response matrix is created to match. Additional data may overlap OSSE data in energy range, and additional data from multiple instruments that overlap each other may be concatenated together into the same file or supplied by the same function or be supplied in separate files or functions. All data are concatenated, including additional data supplied in SDB files specified via the SDBFILE variable (see above). Data may be presented as a formatted file readable by the READ_DATAPTS routine. The order of data is channel edge (MeV), channel width (MeV), flux integrated over channels ($photons/cm^2/s$), flux uncertainty with one line per spectrum channel.

Although the desired flux units or $photons/cm^2/s$, other units may be used. In particular, units that are per unit energy (e.g. /MeV or /keV) may be used. In this case, the comments of the READ_DATAPTS file must include the phrase UNITS="*/*MeV*" or UNITS="*/*keV*" or UNITS="@nnnn" (case insensitive) where the * represents a wild card and nnnn represents a numeric value of the SPECUNITS variable from the SDB header in which the /MeV or /keV bits are set. If this is found, the data are multiplied by the channel widths. Additional data may also be supplied via a function call. In this case, ADD_DATA is a string, starting with '@', that specifies the name of the function to call (e.g. ADD_DATA='@DATFUNC'). The calling sequence of the function is:

DATFUNC(ADDEDG,ADDWID,ADDDAT,ADDSIG,UNITS)

where the first four variables are arrays containing the channel edges, widths, data and uncertainties and the UNITS variable is a string specifying the units as described above. If UNITS is not supplied or does not contain MeV or keV in the denominator, the units are assumed to be $photons/cm^2/s$ and no modification is undertaken.

Mtrxfile
String array (30) containing names of response matrix files. The matrices match the data in that, if more than one spectrum is present, the matrix must have more than one response (each in its own file). These are assembled into a concatenated matrix in FITIT. The default is to look for matrices in SDB file aux. data types.

TARGNAMES
List of target names to use in fit. Default is to use all targets provided in SDR/RMTRX configuration. Each target name corresponds to one source spectrum folded through a response appropriate to that target's position in the collimator for

each count spectrum. Hence, if you desire a fit to only a subset of all the targets included in the compound response, TARGNAMES identifies the desired subset.

Userscreen

String containing name of user screening routine to select data from SDB files. Calling sequence is:

USERSCREEN,SDRS,GOOD_INDEX

where SDRS is an input array of SDRs with dat,err,edg,wid,crc populated and GOOD_INDEX is an output index array pointing to the selected SDRs.

Detid

Integer that specifies which detectors from input SDB files to use. If input is sum of detectors, if any detectors in sum match any of criterion detectors, select spectrum Standard detid bitpick that defaults to 15 (all detectors).

DSA

Integer array(20) that specifies which DSA's to use from input spectra. Functions if input spectra are SDB files or SDR keyword SDRs. If all zeroes, use all DSA's (default).

Scan

Float array(20) that specifies which which scan angles to use from input spectra. Alternative to DSA selection. If both present, Scan selection is used. Functions if input spectra are SDB files or SDR keyword SDRs. If all zeroes, use all scan angles (default). Works by comparing values to targscan from header, with 0.2 degree tolerance.

TJD_period

Double Precision Array(2,10) containing start and stop time pairs which specify intervals to include. Format is day.fraction. Spectra to use must lie entirely within one of the specified intervals. Default (all zeroes) is use all intervals.

*** Fitting Specifications

Modelfile

String containing model file name. This file is used for definition and initialization of the model. If a different model file is used for fit result output, this initialization file can be used repeatedly. Defaults to MODELFILE.DAT.

Fitmodel

String containing file name of model file to receive best fit model. Can be the same as MODELFILE, but this will overwrite the original file, requiring a new initialization file for each fit. Alternatively, the output model file from one fit may be used as the input to another fit. Defaults to <fileroot>.mdl.

Photfile

String containing file name for SDB file containing deconvolved photon spectra. Defaults to <fileroot>_PHOT.SDB. If not set and no **FILEROOT**, or if set to 'NONE', or if model has multiple sources (which implies that the deconvolved photon spectrum is not meaningful) no SDB file is produced.

Erange
Two element array containing minimum and maximum energies to fit (in MeV) i.e. [.06,2.] will cause channels from .06 to 2.0 MeV to be fit. Defaults to entire spectrum.

Nprint
Integer that specifies SUPERFIT diagnostic output mode. See SUPERFIT for details.

Duper
Flag that, when set, activates χ^2 mapping for parameters with the duper flag set in the model file. Defaults to cleared (no χ^2 mapping).

Chi2_prob
Floating value of the significance level desired for parameter uncertainties derived via χ^2 mapping. Defaults to 0.68.

EPS1
SUPERFIT iteration termination criterion (see SUPERFIT). Defaults to 0.0001.

EPS2
SUPERFIT iteration termination criterion (see SUPERFIT). Defaults to 0.0001.

Fitfile
String containing file name of file to receive SUPERFIT diagnostic output. Contents depend on value of NPRINT. Defaults to <fileroot>.dat.

*** Binning Specifications for Plotting

Binfile
String containing file name for ASCII file containing deconvolved spectrum. Defaults to <fileroot>_PHOT.DAT. If not set and no FILEROOT, or if set to 'NONE', no file is produced.

Plot_bindef
String containing name of file or procedure that defines rebinning for plots.

Plot_binfrac
Flag that, if set allows rebinning with fractional spectrum channels. Otherwise, integral numbers of energy channels are added together. Defaults to clear (integral channels).

Plot_chanmod
String containing code that specifies alternative means of defining channel summing for plots. 'LOG': PLOT_NCHAN logrithmically spaced channels 'SQRT': PLOT_NCHAN square root channels 'LIN': Add channels in PLOT_NCHAN groups 'NOSUM':No summation Defaults to 'LOG'. This is overridden by PLOT_BINDEF.

Plot_nchan
Number of summed channels (see above).

*** Display Specifications

Show_cnts
A flag which, if set, results in plots containing count spectra rather than photon spectra. In either case, the best fit model is superimposed on the data. Defaults to photon spectra (i.e. cleared).

Plotmodels
In the case of a model composed of the sum of several model elements, the number of model elements to plot. Defaults to plot all elements. Elements are plotted from the first element to the total requested. For example, if a model consists of a power law and two gaussian lines, entered in the model file in that order, setting PLOTMODELS=1 would result in only the power law being plotted, =2 would result in only the power law and first line, etc. The only way to plot the lines without the power law is to enter them first in the model file (using, for example, BUILDMODEL and MODEL_WRITE), then set PLOTMODELS=2.

Screenplot
Integer flag, set to 1 to produce plots on screen, 0 to produce no plots on screen. Default is produce plots on screen (1).

Hardcopy
String containing the hardcopy device as used in an IDL SET_PLOT command (e.g. SET_PLOT,'PS'). Blank => no hardcopy produced. Defaults to postscript ('PS').

Plot_erange
Two-element array containing minimum and maximum energies to plot (in MeV) e.g. [.06,2.0] would plot from .06 to 2.0 MeV. Note that IDL plotting will round the axes in some cases. Defaults to entire energy range.

Plot_frange
Two-element array containing minimum and maximum fluxes to plot (in $photons/cm^2/s/MeV$ or $counts/cm^2/s/MeV$) e.g. [1.e-4,2.] would plot from 0.0001 to 2.0 $photons/cm^2/s/MeV$. Note that IDL plotting will round the axes in some cases. Defaults to entire flux range.

Plot_xtype
Plot x-axis linear (0) or log (1). Defaults to log.

Plot_ytype
Plot y-axis linear (0) or log (1). Defaults to log.

Residuals
If set, plot residuals in addition to spectra and fit results. Defaults to set (residuals plotted).

*** χ^2 Grid Mapping Specifications

Mapfile
String containing file name for XDR file containing χ^2 map and related variables. Defaults to <fileroot>_MAP.XDR. If not set and no FILEROOT, or if set to 'NONE', no file is produced.

GENCHIMAP
If set, turn on grid-type 1- or 2-Dimensional χ^2 mapping. Default is not set.

NOTE: The following parameters define which parameter(s) to map. They identify the domain, function within the domain (note that many models may be the sum of several functions) and parameter within the function.

The output array MAPARRAY, which (for a 2-D map) contains χ^2 vs parameter-1 and parameter-2 grid point, and the vectors MAP1 and MAP2, which contain the parameter values for each grid column and row (respectively), are defined such that performing a CONTOUR,MAPARRAY,MAP1,MAP2 will align parameter-1 with the x-axis of the plot and parameter-2 with the y-axis of the plot.

MAPPARDOMAIN1
"domain" or source spectrum containing parameter-1 for two dimensional χ^2 mapping (or the only parameter in the case of 1-D mapping). This is a string containing a target name. The target name is converted into a target id number, which is compared to the domain indices in the matrix/model structure. Remember that each domain is one source spectrum and hence is identified by a single target id. If this variable is null, then the first domain will be used regardless of target id. String array of 30 elements, each of which defines the domain for a 1D map of one parameter. For 2-D map, only first element is used.

MAPPARDOMAIN2
"domain" or source spectrum containing parameter-2 for two dimensional χ^2 mapping. This is a target name string (see above).

MAPPARFUNC1
Within the domain defined by MAPPARDOMAIN1, this is a string containing the call name of the function (e.g. Pwrlaw for a power law) containing the desired parameter. String array of 30 elements, each of which defines the function for a 1-D map of one parameter. For 2-D map, only first element is used.

MAPPARFUNC2
Within the domain defined by MAPPARDOMAIN2, this is a string containing the call name of the function containing the desired parameter.

MAPPARFUNC1OCC
Since some models may have multiple occurances of a particular function within a single domain (e.g. multiple Gaussian line functions), this is an integer that identifies which occurance of the function specified in MAPPARFUNC1 contains the desired parameter. 1 is the first occurance, etc. Defaults to 1. Integer array of 30 elements, each of which defines the occurance number for a 1-D map of one parameter. For 2-D map, only first element is used.

MAPPARFUNC2OCC
Since some models may have multiple occurances of a particular function within a single domain (e.g. multiple Gaussian line functions), this is an integer that identifies which occurance of the function specified in MAPPARFUNC2 contains the desired parameter. 1 is the first occurance, etc. Defaults to 1.

MAPPAR1
Parameter sequence number within the specified function of the desired first parameter. Integer array of 30 elements, each of which defines the parameter number for a 1-D map of one parameter. For 2-D map, only first element is used.

MAPPAR2
Parameter sequence number within the specified function of the desired parameter.

MAPPAR1INC
Increment in parameter-1 between grid values. Floating array of 30 elements, each of which defines the parameter increment for a 1-D map of one parameter. For 2-D map, only first element is used.

MAPPAR2INC
Increment in parameter-2 between grid values.

NMAPPAR1
Number of grid points to each side of the best fit point for parameter-1 (i.e. 2*NMAPPAR1 +1 points). Integer array of 30 elements, each of which defines the number of parameters for a 1-D map of one parameter. For 2-D map, only first element is used.

NMAPPAR2
Number of grid points to each side of the best fit point for parameter-2 (i.e. 2*NMAPPAR2 +1 points).

Outputs

Outputs include the following files:

LOGFILE: file containing history of run

PLOTFILE: file containing PostScript plots of deconvolved spectra

FITFILE: file containing SUPERFIT output as specified by NPRINT

FITMODEL: modelfile containing output best fit model

PHOTFILE: SDB file containing deconvolved spectra

BINFILE: ASCII file containing deconvolved spectra binned as specified

PLT_ENE_BANDS

Generates a plot of temperal count rate variablity over a given energy range.

Calling Sequence

PLT_ENE_BANDS, GOOD_SUMFILE, BANDEDG, BANDWID [, keywords]

INPUTS

GOOD_SUMFILE
Name of DAY_SUMMER file.

BANDEDG
Array of lower edge energy values. If not specified, will default to [0.05, 0.15, 0.30, 0.80, 1.50, 4.00].

BANDWID
Array of energy widths. If not specified, will default to [0.10, 0.15, 0.50, 0.70, 2.50, 6.00].

OUTPUTS

One plot of detector count rate variablity over time, and another count rate plot of the sum of the detectors.

KEYWORDS

BYDSA
If set and non-zero, will distinguish SDRs by DSA.

NOT_TAGS
Array of SDR tag numbers not to include in plot.

INSDR
Input SDR used in lieu of file name.

OUTDATA
Output IDL structure array in following format:
{ostruc, strttim: dblarr(300),timwid: dblarr(300), rates: dblarr(300), rerr: dblarr(300), detid:0L, dsa:0L, band:dblarr(2)}.
Structure array is dimensioned (NDETS, NBANDS, NDSA).

AVG
Output IDL structure array of detector-averaged count rates, etc. Has following format:
{astruc, strttim: dblarr(300), timwid: dblarr(300), avgrate: dblarr(300), avgrerr: dblarr(300), var: dblarr(300), dsa:0L, band:dblarr(2)}.
Structure array is dimensioned (NBANDS,NDSA).

TARGNAME
Input target name.

PLOTFILE
Plots to specified file name instead of screen.

SDR$CREATE

This procedure creates an IGORE SDR array having the specified name and number of elements. The format of this command is:

Calling Sequence

SDR$CREATE, r_sdr, ndims

Arguments

R_SDR

The IDL variable into which the IGORE SDR pointers will be placed. The IGORE record created by this routine will be of type IGR$SDR. The name will be truncated if longer than 15 characters.

NDIMS

An integer specifying the number of elements in the record array to be created.

Keywords

NAME

A character string which, if present, specifies the name of the IGORE record to be created. The name will be truncated if longer than 15 characters. If not provided, the IGORE record will be created using the name of the IDL variable passed as R_SDR.

/UNIQUE

If present and non-zero, /UNIQUE forces the generation of a unique name for the IGORE record. Ignored if NAME is specified.

Discussion

This routine creates an IGORE record array having a structure type IGR$SDR. Note that SDR$CREATE performs the same operation as the routine CREREC with the structure type IGR$SDR. An example of the usage of SDR$CREATE is given below:

```
Igore> SDR$CREATE, r_sdr1, 10, name='sdr1'
Igore> SDR$CREATE, r_sdr2, 10
Igore> CREREC, 'IGR$SDR', r_sdr3, 10, name='sdr3'
Igore> CREREC, 'IGR$SDR', r_sdr4, 10
```

NOTE: IGORE does not currently support record arrays of more than one dimension.

SDR_LOAD

This procedure creates and loads an array of SDRs from an SDB file. Optionally, selections can be made on the contents of the file to be loaded. The default is header, data, errors, edges, and widths from all records in the file. In addition, data can be optionally summed into summary SDRs.

Calling Sequence

SDR_LOAD, filename, sdr [,outsdr]

[This procedure is written in IDL]

Arguments

FILENAME

A character string specifying the name of the input SDB file. The default directory is the current directory, the default file extension is .SDB. The file must exist.

SDR

An array of SDR records created from the information in the input SDB file and selection criteria. If the SDR array exists on entry, it is deleted and re-created with the information in the file.

OUTSDR

An optional parameter for returning the individual SDRs from the file when the SUM keyword is set to return sums in the SDR argument. If SUM is not specified, this argument is not used.

Keywords

AUXA

This keyword specifies the aux types to be extracted from the SDB records in the SDB file. By default, hdr, dat, err, edg, wid, are read. AUXA can be specified as a string of aux types separated by commas - eg. 'dat,err,edg,wid,crc,note' or as a two element long array specifying "bit-picks". AUXA overrides the default and requires default aux types to be specifies in AUXA if desired.

AUX1FLD

A longword which specifies the AUX1 bits. The default is equivalent to hdr, dat, err, edg, and wid.

AUX2FLD

A longword which specifies the AUX2 bits. The default is zero.

/BYCFG
Input switch which if present and nonzero, specifies that summing should be performed for each unique detector configuration. Default is single sum of available data.

/BYDAY
Input switch which if present and nonzero, specifies that summing should be performed for each (approximate) daily interval. Default is single sum of available data.

/BYDET
Input switch which if present and nonzero specifies that summing should be performed by detector. By default, all SDRs are summed into a single sum. The keyword SUM must be set along with BYDET to create individual SDRs for each detector.

/BYDSA
Input switch which if present and nonzero, specifies that summing should be perfomed by detector step angle or source aspect. By default, all SDRs are summed into a single sum. The keyword SUM must be set along with BYDSA to create individual SDRs for each step angle. BYSCAN overrides BYDSA and a warning is issued. The combination of BYDSA and BYDET will create a #dsa by #det element SDR array.

/BYSCAN
Input switch which if present and nonzero, specifies that summing should be perfomed by target scan angle (HDR.TARGSCAN) or source aspect. By default, all SDRs are summed into a single sum. The keyword SUM must be set along with BYSCAN to create individual SDRs for each scan angle. BYSCAN overrides BYDSA and a warning is issued. The combination of BYSCAN and BYDET will create a #scan by #det element SDR array. NOTE: TARGSCANS are rounded to 0.2 degrees to find unique ranges.

DETID
This keyword specifies which detectors to extract from the the SDB file. Its format is the standard "bit-pick" specification for 1 or more detectors. The default is to select all detector data from the SDB file.

ELEMENTS
An input array of indices of SDB records to be read from the requested SDB file. By default, all elements are read. Simultaneous usage of the keywords, TAGS and ELEMENTS is not supported.

ERSTAT
A output array containing the SDB$READ status. Array elements are non-zero if the data requested for an associated SDR record was not available.

/MATRIX
Input switch, which if present and non-zero, causes summing of response matrices if found in input SDRs. Live time weighted rmtrx is stored in output SDRs.

NOT_ELE
An input array of indices of SDB records to exclude. By default, all elements are read.

NOT_TAGS
An input array of one or more tagwords of SDB records to be excluded. By default, all tags are read.

PLUS_AUX
This keyword specifies which additional aux types should be loaded. The standard dat, err, edg, wid will be loaded, plus those specified with this keyword.

/RELSPEC
Input switch which if present and nonzero, specifies that RELSPEC records should be added to input SDR. The RELSPEC records document the file name and tagword of the file and SDR containing the summed data.

START_TIME
Input specifying start time limit for selection. Records with start times greater or equal to this time are selected. Time is specified as tjd and fraction in one of many forms. eg. as string - 'yy/ddd hh:mm:ss', as real - yyddd.fff. See routine TJD_FROM_ASCII. Default TJD_START is 0, ie accept all records.

STOP_TIME
Input specifying stop time limit for selection. Records with stop times less than or equal to this time are selected. Time is specified as tjd and fraction in one of many forms. eg. as string - 'yy/ddd hh:mm:ss', as real - yyddd.fff. See routine TJD_FROM_ASCII. Default TJD_STOP is 1e10, ie accept all records.

/SUM
An input logical which if present and nonzero indicates that the sum of the individual records should be performed and returned. By default, a single sum is generated. Keyword BYDET changes this default action. If SDRs exists upon entry, they are delete and output SDRs for the sums are created.

TAGS
An input array of one or more tagwords of SDB records to be read from the requested SDB file. By default, all tags are read.

/UNIQUE
Input switch, which if present and non-zero, indicates that unique logical names should be created for each input SDB filename. The default is to abort if duplicate filenames are found, thus /UNIQUE permits using the same filename in different directories.

WEIGHT
An integer value which specifies the weighting method to be used when performing the spectral sum. The supported values are given in Table **??**. Default is 2 (LTNORM).

Discussion

```
EXAMPLE:

  #1 get SDRs from file for dets 1 and 2 and sum into individual sums

     igr>  sdr_load, '$prod:p931230000_dif.sdb', sdr_sums, $
     igr>            /sum,  /bydet,  detid = 3

  #2 get SDRs from file with footnotes, crc, and psdeff data

     igr> sdr_load, filename, sdr, auxa = 'dat,err,edg,wid,note,crc,psdeff'
```

Related Routines

Also see the related function SDR_SUM.

SDR_SUM

This procedure sums an array of IGORE SDRs into an output SDR. The format of this command is:

Calling Sequence

SDR_SUM, in_sdr, out_sdr

Arguments

IN_SDR

Array of one or more Igore SDR records which have HDR, DAT, and ERR fields. The SDB header must be loaded, this routine will load others as needed.

OUT_SDR

A single SDR in which to store the sum of the input SDRs. OUT_SDR must exist on call. It is returned with HDR, DATA, ERR, EDG, and WID fields populated.

Keywords

AUX

Input string specifying list AUX data types required to be in the input SDRs. Warning messages are printed if these data types are not in input SDRs.

NOTE: Currently supports only CRCs.

/BYCFG

Input switch which if present and nonzero, specifies that summing should be performed for each unique detector configuration. Default is single sum of available data.

/BYDAY

Input switch which if present and nonzero, specifies that summing should be performed for each (approximate) daily interval. Default is single sum of available data.

/BYDET

Input switch which if present and nonzero, specifies that summing should be performed by detector. Default is a single sum of available data.

/BYDSA

Input switch which if present and nonzero, specifies that summing should be perfomed by detector step angle or source aspect. By default, all SDRs are summed into a single sum. The keyword SUM must be set along with BYDSA to create individual SDRs for each step angle. BYSCAN overrides BYDSA and a warning is issued. The combination of BYDSA and BYDET will create a #dsa by #det element SDR array.

/BYSCAN
Input logical which if present and nonzero, specifies that summing should be perfomed by target scan angle (HDR.TARGSCAN) or source aspect. By default, all SDRs are summed into a single sum. The keyword SUM must be set along with BYSCAN to create individual SDRs for each scan angle. BYSCAN overrides BYDSA and a warning is issued. The combination of BYSCAN and BYDET will create a #scan by #det element SDR array. NOTE: TARGSCANS are rounded to 0.2 degrees to find unique ranges.

COR_MATRX
Output IDL structure containing the correlation matrix.

CRC_SUM
Output IDL structure containing the resultant CRCs from the sum.

EDGES
Input float array which specifies the lower edges (MeV) of the output channels. Overrides default gain computation. Must have same dimensions as DAT field in SDR.

/MATRIX
Input switch, which if present and nonzero, causes summing of response matrices in input SDRs to be performed and stored in output SDRs.

/RELSPEC
Input switch which if present and nonzero, specifies that RELSPEC records should be added to input SDR. The RELSPEC records document the file name and tagword of the file and SDR containing the summed data.

/VERBOSE
Logical switch specifying that information messages should be displayed. Default is not to display information messages.

WEIGHT
An integer value which specifies the weighting method to be used when performing the spectral sum. The supported values are given in Table ??. Default is 2 (LTNORM).

WIDTHS
Input float array which specifies the widths (MeV) of the output channels. Both edges and widths should be specified to override default gain computation.

Discussion

SDR_SUM performs a live-time weighted sum of an array of SDRs into an output SDR.

The summing assumes that the input array of SDRs have HDR, DAT, and ERR fields populated. SDR$SUM is called to perform the summing of data and correct update of header information.

OUT_SDR should be created before calling SDR_SUM.

NOTE: Edges and widths returned in OUT_SDR are obtained via one of several alternatives. If edges and widths are supplied via the keywords EDGES and WIDTHS and have the correct dimensions, the supplied edges and widths are used. Otherwise, if edges and widths are found in the IN_SDR or in the corresponding SDB file, the average edges and widths are computed. If not, and the input data has 544 channels (no rebinning), edges and widths are computed by SDR_ESCALE using the SDB header RGAIN values in IN_SDR, otherwise no edges and widths are returned.

NOTE: If input IN_SDR array contains data from multiple detectors, they all get summed into a single SDR_OUT with channel edges and widths averaged — garbage in — garbage out, unless the /BYDET keyword is used to sum by detector.

SPECANAL

Calling Sequence

SPECANAL [, SETUPFILE] [, keywords]

Arguments

SETUPFILE

Input string specifying an anal_struc namelist file which defines the analysis configuration. This is an optional input parameter. Keyword specifications can set all anal_struc configuration variables and will override specifications in the setup_file. File extension for these files default to "ANL".

Indirectly Referenced Inputs:

The anal_struc namelist permits the specification of configuration files for each of the major processes: BLD_QUAD_DIF, DAY_SUMMER, and FIT_PREP. If these specifications in the setup file are blank and keyword alternate specification is not used, these configurations default to nominal configuration files which are:

IGR$IGORE_DATA:NOM_CFG.BQD	BLD_QUAD_DIF configuration
IGR$IGORE_DATA:NOM_CFG.DSM	DAY_SUMMER configuration
IGR$IGORE_DATA:NOM_CFG.FP	FIT_PREP configuration
IGR$IGORE_DATA:NOM_CFG.PA	PLOTANAL configuration
IGR$IGORE_DATA:NOM_CFG.BLN	bottomline configuration

Keywords

/NOBATCH

Input logical, if set, executes the task as part of parent process rather than submitting it to a batch queue. Default is execute processing as a batch job. Depending on exact configuration, /nobatch mode can take a long time, locking the current IGORE session.

QUEUE

Input string specifying the queue name for batch submission. Default queue is SYS$BATCH.

/BUILD

Input logical, if set, causes creation of the batch command file but does not submit it for execution. All configuration information is available for inspection prior to submission.

COMFILE
Input string specifying the name of the batch command file to be created. By default this filename is created from the FILEROOT which in turn is derived from target name and viewing period.

/CREATE
Input logical, if set, will create directories, if necessary, for products from information derived from configuration files. Default is to check for existance of these directories and abort submission of batch job if they do not exist.

/SAVE
Input logical, if set, records the specanal configuration in an anal_struc namelist file. This file is in the format of the SETUPFILE input parameter, Default name for the configuration file is derived from the FILEROOT as <FILEROOT>.ANL. Keyword CFGFILE overrides the default naming of this file.

CFGFILE
Input string specifying name of saved config file. Default is derived from FILEROOT. Only meaningful with /SAVE keyword.

Additional keywords are processed by the ANAL_SETUP routine. You may note that many correspond to entries in the SETUPFILE namelist. These are described below.

TARGET
Specify the OSSE target to process. This input may be either a target name in a string, a target id as an integer, or a dummy structure.

VP
Specify the viewing period to process (floating point value).

START_TIME

STOP_TIME

PROD_DIR

OUT_DIR

FILEROOT

LOGFILE

/NOBQD
If set, do *not* run BLD_QUAD_DIF Default is to generate them, NOBQD=0.

BQD_CONFIG

BQD_DIR

/NODAYSUM
If set, do *not* run DAY_SUMMER. Default is to generate them, NODAYSUM=0.

DAYSUM_DIR

DAYSUM_OUT

DAYSUM_GOOD

DAYSUM_CONFIG

/NOFITPREP
If set, do *not* generate FFIT_PREP plots and files. Default is to generate them, NOFITPREP=0.

FITPREP_OUT

FITPREP_CONFIG

FITPREP_DIR

/NOPLOTANAL
If set, do *not* generate 'Plot Analysis' plots and files. Default is to generate them, NOPLOTANAL=0.

PLOTANAL_CONFIG

PLOTANAL_DIR

/NOBLN

If set, do *not* generate 'Bottom-Line' plots and files. Default is to generate them, NOBLN=0.

BLN_OUT

BLN_CONFIG

BLN_DIR

PRODUCTION

NODQSUMMARY

Outputs

<dflt_dir><fileroot>.ANL SPECANAL Setup file if /SAVE is used

<dflt_dir><fileroot>.COM Batch file that is submitted and executed

<out_dir><fileroot>.CFG Complete configuration of the run including anal, bqd, dsm, fitprep namelists

<out_dir><fileroot>.LOG Result of batch execution

<bqd_dir><...>_DIFF.SDBs BLD_QUAD_DIF difference files, 1 per day

<dsm_dir><fileroot>_DSM.SDB DAY_SUMMER standard sum file

<dsm_dir><fileroot>_GSM.SDB DAY_SUMMER good sum file

<fp_dir><fileroot>_PREP.LOG FIT_PREP run log file

<fp_dir><fileroot>_TOT.SDB FIT_PREP observation total file w/response

<fp_dir><fileroot>_DAY.SDB FIT_PREP observation daily file with pointers to total file response

<fp_dir><fileroot>_PREP.PS FIT_PREP plots. These are difference count spectra plotted for each detector on the source. The are plots linear in energy from zero to 4 MeV and logarithmically for 0.01 to 10 MeV. These plots are also generated with rebinning.

<pa_dir><fileroot>_PLOT.LOG PLOTANAL run log file

<pa_dir><fileroot>_COUNT.PS PLOTANAL count rate plot. The count plots are generated by individual detectors and summed detectors. Each of these plots is run

over two energy ranges (linear & logarithmic) and two different energy-binnings. The DSA in the graph title represents the Detector Step Angle corresponding to an on-source pointing.

<pa_dir><fileroot>_EBAND.PS PLOTANAL temporal plot (i.e. a light curve). Plots are generated for 4 energy bands: 0.05–0.15 MeV, 0.15–0.30 MeV, 0.30–0.80 MeV, 0.80–1.50 MeV, showing the daily-summed source intensity for each day of the observation. Curves are plotted for each detector and their sum. For the summed plots, the mean is computed for the observation and used as the model for the fit. P is the probability that the (constant) intensity model is consistent with the data points. The final table compares the data from one detector with the model produced by the average of the other detectors.

<pa_dir><fileroot>_DAILY.PS PLOTANAL daily count rate plots. This set of plots consists of two major components: daily plots showing each detector (and two energy ranges for each detector) and each detector summed for the duration of the observation. Each data set is fit to a constant model of zero, the χ^2 representing the probability that the plot is consistent with the zero model (probability of a non-detection).

<pa_dir><fileroot>_STATE.PS PLOTANAL version of the SHOW_WHEN plot. Shows the pointing strategy used by each detector for each state of the instrument. Good for recognizing possible confusing sources in both source & background pointings.

<bln_dir><fileroot>_BLN.LOG Bottom-line run log file

<bln_dir><fileroot>_RBN.SDB Bottom-line count rates resampled to "AGN-like" bins

<bln_dir><fileroot>_BLN.SDB Bottom-line crunched count rates

<bln_dir><fileroot>_BLN.PS Bottom-line plots (PostScript file). Generates resampled plots in both count and photon space.

<bln_dir><fileroot>_PHOT.DAT ASCII file listing 'approximate' photon flux points. This is useful for plotting the output with your own plotting software.

When converting the output from counts to flux, a model must be defined to assist the instrument response deconvolution process. SPECANAL defaults to utilizing a simple power-law model, specifically $10^{-3}(E/0.1MeV)^{-2.0}$. It is NOT a best-fit model! This should be kept in mind when interpreting this output.

Note: keywords and referenced configuration files can significantly alter the naming and directories of these products.

Appendix A

Data Structures

Auxiliary Data Types

Keyword	Bit	No.	S/D	Class	Description
AUX1					
DAT	0	0	D	n/a	Spectrum data
ERR	1	1	D	1	Spectrum statistical uncertainties (sym. or positive)
NERR	2	2	D	1	Spectrum negative statistical uncertainties
SYERR	3	3	D	1	Spectrum systematic uncertainties (sym. or positive)
NSYERR	4	4	D	1	Spectrum negative systematic uncertainties
EDG	5	5	S	1	Channel lower edge energies
DEDG	6	6	S	1	Channel lower edge energy uncertainties
CEN	7	7	S	1	Channel representative energies
DCEN	8	8	S	1	Channel representative energy uncertainties
WID	9	9	S	1	Channel widths
DWID	10	10	S	1	Channel width uncertainties
PSDEFF	11	11	S	1	PSD efficiency
PSDDEFF	12	12	S	1	PSD efficiency uncertainties
RES	13	13	S	1	Detector resolution
RERR	14	14	S	1	Detector resolution uncertainties
MSDSF	19	19	D	3	Multi-spectrum data from DSF (IGR$MULTI_HDR,IGR$MULTI)
MSDAT	20	20	S	1	Multi-spectrum data
MSERR	21	21	S	1	Multi-spectrum statistical uncertainties (sym. or positive)
NMSERR	22	22	S	1	Multi-spectrum negative statistical uncertainties
MSSYERR	23	23	S	1	Multi-spectrum systematic uncertainties (sym. or positive)
NMSYERR	24	24	S	1	Multi-spectrum negative systematic uncertainties
CALIB	25	25	S	2	Calibration line information (IGR$CAL)
RINFO	26	26	S	1	Detector resolution information
PSDMOD	27	27	S	1	PSD model information
RELSPEC	28	28	S	2	Related spectra (IGR$RELSPEC)
ORBHIST	29	29	S	2	Orbit history (IGR$ORB)
NOTE	30	30	D	2	Footnote (textual comment) (IGR$NOTE)
AUX2					
BKG	0	32	D	1	Estimated background
DBKG	1	33	D	1	Estimated background uncertainties
AUDEF	2	34	S	2	Analyzable unit definition (IGR$AU_DEF)
AUID	3	35	D	2	Analyzable unit identification (IGR$AUID)
SPECLST	4	36	S	1	Selected spectra list
CRC	5	37	D	2	Compound response coefficients (IGR$CRC)
RMTRX	6	38	S	1	Instrument response matrix
PRV	7	39	S	1	Photopeak response vector
MCDB1	8	40	D	1	Reserved for Monte Carlo DB
MCDB2	9	41	D	1	Reserved for Monte Carlo DB
DIR	10	42	S	2	Directory (IGR$DIR)
RMDEF	11	43	S	2	Instrument response matrix definition. (IGR$RMDEF)
RMVEC	12	44	S	1	Instrument response matrix vector.
COV	13	45	D	2	Covariance Information. (IGR$COVAR)

Table A.1: Auxiliary Data Types: S refers to “Sparse” data types while D refers to “Dense” data types. Class 1 is simple array, class 2 is structured data, class 3 is structured header plus array or structured data. The description includes the IGORE structure name, where applicable.

SDB Header Operations

SDB V7 Header		Process				
SDB Variable	Description	PDA	Best	Write	Sum	Sub
nbytes_per_record	size of data records	r	r	r	r	r
numrec	Number of records	r	r	r	r	r
nument	Number of entities	r	r	r	r	r
sizeent	Number of bytes per entity	r	r	r	r	r
dataversion	Data type version number	r	r	r	r	r
hdrvers	Header version number	u	u		1	1
hdrrev	Header revision number	u	u		1	1
softindex	Creation software index	u	u		1	1
softmajor	Creation s/w major version	u	u		1	1
softminor	Creation s/w minor version	u	u		1	1
cmdstate_id	Cmd state table ID	b	S		b	1
detcfg_id	Det cfg table ID	b	S		b	1
mdpostbl_id	MD Pos. table ID	b	S		b	1
credate	Creation date/time, VAX format	u	u		u	u
detid	Detector ID	u	S		or	1
targid	Target name code	u	S		1	1
targoff	Source/background offset	u	S		1	1
tlmid	Telemetry format ID	u	S		1	1
reactor	Reactor event code	u	or		or	or
dsfstat	DSF stop/event status code	u	or		or	or
dqhaz	Data quality flags hazardous	u	or		or	or
dqinfo	Data quality flags info	u	or		or	1
usrscrnflg	User spectrum screening flags		or		or	or
specsrc	Spectrum source code	u	S		1	1
speccat	Spectral category code	u	S		1	1,u
specunit	Units code	r	r		1,r	1,r
specflgs	Spectrum flags	r	r	r	1,r	1,r
calrpha	CALPHA/RPHA Mux	u			1	1
ltl	SPM low live time	u	S		$\sum$	1
ltm	SPM med live time	u	S		$\sum$	1
lth	SPM high live time	u	S		$\sum$	1
accumtim	Accumulation time	u	S		$\sum$	1
strttim	Start time (in TJS)	u	S		min	1
strt_frac	Start time fraction	u	S		min	1
stoptim	Stop time (in TJS)	u	S		max	1
stop_frac	Stop time fraction	u	S		max	1
orbit	Orbit number	u	S		min	1

SDB V7 Header (continued)		Process				
SDB Variable	Description	PDA	Best	Write	Sum	Sub
srcct(2)	TLM Source Cnt	u	S		1	1
nchan	Total # of spectrum channels	r	r	r	1,r	1,r
nchanl	# of low range channels	r	r	r	1,r	1,r
nchanm	# of medium range channels	r	r	r	1,r	1,r
nchanh	# of high range channels	r	r	r	1,r	1,r
nchan1	# of channels, parameter 1	r	r	r	1,r	1,r
nchan2	# of channels, parameter 2	r	r	r	1,r	1,r
auxflg1	AUX field flags (1st 32)	r	r	r	0,r	0,r
auxflg2	AUX field flags (2nd 32)	r	r	r	0,r	0,r
dptbl	Active positioning table	u	S		1	1
fdet	Detector positioning status	u	S		1	1
dsa	Detector position, end	u	S		1	1
detra	Detector RA	u	S		1	1
detdc	Detector declination	u	S		1	1
targscan	Target latitude (scan angle)	u	S		1	1
targxscan	Target longitude (transscan)	u	S		1	1
earthaz(2)	Earth azmiuth in det. coord.	ss	S		mm	1
earthsep(2)	Earth/collimator seperation	ss	S		mm	1
targra	Target RA 2000 epoch	u	S		1	1
targdc	Target declination 2000 epoch	u	S		1	1
collposang	Collimator position angle	u	S		1	1
sig_scan	Mean scan ptg error (sigma)	0				
sig_xscan	Mean xscan ptg error (sigma)	0				
rgain	Actual Relative Gain Ratio	u	S		1	1
rfwhm	Actual Relative Full Width Half Max	u	S		1	1
stdrgain	Standard Relative Gain Ratio	u	S		1	1
stdrfwhm	Standard Relative Full Width Half Max	u	S		1	1
sum_tim	Rate collection time	u	S		$\sum$	1
"det rate sums"	26 det rate sums (counts)	u	S		$\sum$	1

SDB V7 Header (continued)		Process				
SDB Variable	Description	PDA	Best	Write	Sum	Sub
mcl(2)	McIlwain L	ss	S		mm	1
mcb(2)	McIlwain B	ss	S		mm	1
rigid(2)	Rigidity	ss	S		mm	1
brad(2)	Radial component of mag fld vec	ss	S		mm	
bpol(2)	Polar component of mag fld vec	ss	S		mm	
baz(2)	Az component of mag fld vec	ss	S		mm	
day_night	Day/Night code	u	S		or	1
cpml_rate	CPM low rate	u	S		A	1
cpmh_rate	CPM high rate	u	S		A	1
orbit_tim	Accum time from strt of orb (sec)	u	S		1	1
cpml_orbit_sum	CPML dose since strt of orb (cnt)	u	S		1	1
cpmh_orbit_sum	CPMH dose since strt of orb (cnt)	u	S		1	1
last_saa_exit	Time since last SAA exit (sec)	u	S		1	1
last_saa_tim	CPM accum time for last SAA (sec)	u	S		1	1
cpml_last_saa	CPM low dose during last SAA (cnts)	u	S		1	1
cpmh_last_saa	CPM high dose during last SAA (cnts)	u	S		1	1
grora(2)	RA of s/c	ss	S		mm	1
grodc(2)	Dec of s/c	ss	S		mm	1
grolg(2)	Longitude of sub-s/c point	ss	S		mm	1
attzra(2)	Axis RA 2000 epoch	ss	S		mm	
attzdc(2)	Z Axis Dec 2000 epoch	ss	S		mm	
attxra(2)	X Axis RA 2000 epoch	ss	S		mm	
attxdc(2)	X Axis Dec 2000 epoch	ss	S		mm	
grorad(2)	Alt of s/c above earth	ss	S		mm	
rsc(3)	Barycenter vectors (sec)	u	S		1	1
etut	Barycenter time	u	S		1	1
strtbilvl(5)	Bilevels 0-4, start of acq.	u	S		1	1
endbilvl(5)	Bilevels 0-4, end of acq.	u	S		1	1
ptmp1	PMT mount plate temperature	u	S		1	1
c6tmp	Coll./Co-60 temperature	u	S		1	1
s1tmp	Xtal housing cover temperature	u	S		1	1
cpdtm	CPD temperature	u	S		1	1
"det analogs"	Various analog housekeeping (30)	u	S		1	1
caltw	Calibration tagword	u	C	r	1	1
stdcaltw	Standard calibration tagword	u	C	r	1	1
orbittw	Orbit history tagword	u	C	r	1	1
relspectw	Associated spectra tagword			1	1	1
notetw	Footnote tagword			1	1	1
resptw	Response tagword			1	1	1

SDB V7 Header (continued)		Process				
SDB Variable	Description	PDA	Best	Write	Sum	Sub
achd(25)	Acquisition header	u			1	1
hskptim	Time of slot header packet	u			1	1
tsac	Seconds/TSAC	u	S		1	1
am	Average pot main	u	S		1	1
ar	Average pot redn	u	S		1	1
effecarea	Effective area	u	S		1	1
divisor	Divisor					
phbinnum	Phase bin number					
numphbin	Total number of phase bins					
reserve(37)	Reserve					

Notes:

$\sum$: value is sum of associated headers
0: values initialized to zero
1: values copied from 1st header of array
b: value at the start time of the collection interval
C: values from "source" header, data copied from input to output
min: values copied from header with minimum time
max: values copied from header with maximum time
mm: 1st is min as defined above, 2nd is max as defined above
or: value is logical "or" of associated headers
r: required for correct structure intrepretation or analysis (implies "u")
S: values copied from "source" header
ss: elements are start and stop of interval
u: updated or assigned by process