Automated Processing of Fermi/GBM GRB Triggers

Stephen Holland, Jerry Bonnell, and Davide Donato

September 25, 2012

1 Introduction

The purpose of this document is to describe the DO_GBM.PY script. This script is intended to do a preliminary automated analysis of *Fermi*/GBM gamma-ray burst trigger data. It operates on one GRB trigger at a time. The script is run in the directory that contains the input data, for example

`~/gbm/triggers/year/bnYYMMDDNNN/current`

However, it is *strongly* recommended that the data in this directory be copied to a working directory (e.g., `~/gbm/triggers/year/bnYYMMDDNNN/work`) before running the script in order to avoid any accidental loss of the input data. The script implements the GBM Gamma-Ray Burst Analysis Thread at
`http://fermi.gsfc.nasa.gov/ssc/data/analysis/scitools/gbm_grb_analysis.html`
as it was in August 2011. Some changes have been made to enable automation and to try and improve the results.

DO_GBM.PY is written in Python 2.6, but every effort has been made to make it consistent with Python 3. The script requires that the following third party Python libraries.

numpy - This is needed to fit the background, and is required by some of the other third party libraries.

pyfits - This is the STScI Python library for operating of FITS files.

xspec - This is the Python XSPEC module distributed by HEASARC.

2 Processing Steps

The basic structure of the automated processing script is given below.

1. Read the command line arguments

2. Identify the detector files.
3. For each detector file do the following.
   (a) Create a light curve.
   (b) Determine Bayesian blocks for the light curve.
   (c) Fit and subtract the background.
   (d) Determine new Bayesian blocks to identify the source.
   (e) Compute the signal-to-noise ratio (S/N) per bin for the source.
4. Take the detector with the highest S/N per bin to be the best detector.
5. Extract a background-subtracted light curve from the best detector.
6. Compute T_{90} for the best light curve.
7. Create a PHA2 file for each detector.
8. Identify the detectors with a good source detection.
9. Find the response matrix files.
10. Use PYXSPEC to perform a joint spectral fit to all of the good data.
11. Write a summary of the results.

2.1 Read the Command Line Arguments

At present this is done by brute force in the main Python script. This should be changed to take advantage of Python's `argparse` module. The command line arguments currently allow the user to specify the trigger name, the background and source time intervals to use, the detectors to use, and to specify TRIGTIME. See § 3 for details on how to call the script.

Background and source intervals can be optionally specified. If NONE is entered then the code will automatically determine the time intervals for each detector. If the time intervals are specified by the user they are given in the form `bkg1_start-bkg1_stop:src_start-src_stop:bkg2_start-bkg2_stop` where `bkg1` indicates the background interval before the burst starts, `src` indicates the interval when the source is active, and `bkg2` indicates the background interval after the source. All three must be specified.

TRIGTIME can either be specified by the user (in MET), or read from the detector files by entering TRIGTIME.

2.2 Identify the Detector Files

There are three ways that the user can specify which detector files to use. The `detectors` command line argument can be set to ALL, TRIGGER, or a string of 0s and 1s that specify specify which detectors to use in the format `nnnnnnnnnnnn`.

If `detectors` = ALL then the working directory is scanned for any files of the form `glg_*_*_v*.fit`. These files are assumed to be GBM detector files.

If `detectors` = TRIGGER then the DET_MASK keyword is read from the `glg_tcat_all_<trigger_id>_vNN.fit` file and only the detectors that were triggered by this burst are used.

If `detectors` = `nnnnnnnnnnnn` then only the detectors that are indicated by a "1" are used. The format of this string is exactly the same as the format of the DET_MASK keyword. For example, the string '010010001000' indicates that detectors n1, n4, and n8 are to be used.

2.3 Process the Individual Detector Files

Once the detector files that we are interested in have been identified we work with each one one at a time.

2.3.1 Create a Light Curve

Run the GTBIN program to extract the light curve. Use the default light curve time bins from Valerie Connaughton's presentation at the Goddard Data Analysis Workshop in Dec 2010. These are 0.128 s for TTE data, 1.024 s for CSPEC data, and 0.064 s for CTIME data. If the data type is not one of these then the bin width is set so that there are 1024 bins in the light curve.

2.3.2 Determine Bayesian Blocks

If the user specifies the background and source time intervals then use those and skip to § 2.3.5.

If the program is to determine the background and source intervals automatically then run GTBURSTFIT to compute Bayesian blocks for the light curve. These will be used to estimate the start and end of the burst. This is a preliminary estimate because the background has not been subtracted yet. The pre-burst background interval is assumed to be the duration of the first Bayesian block. The source interval is assumed to be the interval from the end of the first Bayesian block to the start of the last Bayesian block. These criteria are the same as those used by the *Swift*/BAT tool BATTBLOCKS. If the last Bayesian block is shorter than one second then the source stop time is taken to be the start of the second-to-last Bayesian block. This was done so that there will be at least one second of data at the end of the light curve to fit the background.

There is an error in the FHELP documentation for GTBURSTFIT. The documentation says that GTBURSTFIT fits the following model, $C(t)$, to a light curve consisting of N pulses.

$$C(t) = B + \sum_{i=1}^{N} A_i \exp\left(\tau_{1,i}/(t - t_{0,i}) + (t - t_{0,i})/\tau_{2,i}\right) \tag{1}$$

The background counts is B, A_i is the amplitude of pulse i, $t_{0,i}$ is the start time of pulse i, and $\tau_{1,i}$, $\tau_{2,i}$ are the rise and decay constants respectively of pulse i. However, the equation that is actually used in GTBURSTFIT is

$$C(t) = B + \sum_{i=1}^{N} A_i e^{2\mu_i} \exp\left(-\tau_{1,i}/(t - t_{0,i}) - (t - t_{0,i})/\tau_{2,i}\right) \tag{2}$$

where $\mu_i = \sqrt{\tau_{1,i}/\tau_{2,i}}$. DO_GBM.PY does not make direct use of this equation, but this discrepancy will be important if we ever want to add the option to plot the fitted model against the GBM light curve.

2.3.3 Fit and Subtract the Background

This step is only done if the source and background intervals are being determined automatically.

Fit and subtract the background from the light curve. Add back the mean background so that Poisson statistics will still give reasonable results. The background is fit with a 2$^{\text{nd}}$ order polynomial. The background regions before and after the source are used for the fit. The last bin of the light curve may be incomplete because the duration of the data may not be an even multiple of the bin width. To avoid having this bias the results we set the counts and error for the last bin equal to the values for the second-to-last bin.

2.3.4 Determine Better Bayesian Blocks

This step is only done if the source and background intervals are being determined automatically.

Now that we have a background-subtracted light curve recompute the Bayesian blocks to try and get a better estimate of the burst duration.

2.3.5 Compute S/N Per Bin

Compute the mean S/N per bin of the source. This is intended to be an estimate of the quality of the signal in this detector. The background is fit and subtracted so that all of the counts are either from the source or noise. The S/N is computed for each bin in the light curve. The source is assumed to start at the source start time (third entry) in the input `intervals` list. The source is assumed to stop when three consecutive bins have negative S/N. If this does not happen then the source is assumed to stop at the source stop time (fourth entry) in the input `intervals` list.

2.4 Find Detector with Best Source Signal

The best detector is defined to be the detector that gives the highest S/N per bin. If none of the detections have a S/N per bin value above some threshold then we assume that there was no detection for this GRB and stop. The threshold is currently S/N per bin = 1.5.

2.5 Extract the Background-Subtracted Light Curve for the Best Detector

We want to compute T_{90} in a specified energy range using the data from the best detector. The energy range is currently 50–300 keV. These numbers are set at the start of the `main()` section of the code. Run FSELECT to select events in this energy range,

We follow the steps in § 2.3 to produce this light curve, but we omit step 2.3.5.

2.6 Compute T_{90}

Compute T_{90} for the energy-selected, background-subtracted light curve from the best detector. Use the algorithm described in the BATTBLOCKS documentation. See that help file for details. We use the TOTVAR method to compute the error in T_{90}.

2.7 Create the Spectra (PHA2) Files

Use GTBIN to create PHA2 spectra for each detector.

2.8 Identify Good Spectra

Good spectra are defined as those where the source S/N per bin in the light curve is above some threshold. This is currently the same threshold as in § 2.4. Only detectors that satisfy this criterion are used in the spectral fitting.

2.9 Find the Response Matrices

Find the response matrix that corresponds to each PHA2 file. The response matrix is assumed to in a file with a name like `gle_cspec_SSS.rsp` where `SSS` is a string that exactly matches that of the PHA2 file.

2.10 Perform a Joint Spectral Fit

This section is still somewhat rough. We use PYXSPEC to do joint fits to all of the good spectral data. We fit a Band function and a cutoff power law, then take the one with the best fit as the best-fit model. The energy range for computing the fluence is currently 10 – 1000 keV and is specified at the start of the `main()` section of the code.

The script creates an XSPEC command file called `<trigger_id>_xspec_save-.xcm` that contains the XSPEC commands needed to read in the data and fit the models. This file is intended to allow a user to restart and refine the spectral analysis for a trigger.

In addition to the XSPEC save file the code creates a postscript plot showing the fits to the Band model and the cutoff power law model. These files are called `grbm_<trigger_id>_xspec.ps` and `cutoffpl_<trigger_id>_xspec.ps` respectively. PYXSPEC creates a log file called `<trigger_id>_xspec.log` that contains the standard XSPEC logging output.

Here are some notes on how the spectral fitting is done.

Each PHA2 file contains two spectra. The background spectrum is stored first, because it extracts the spectrum before the burst went off. The source plus background spectrum is stored second. Hence, the XSPEC `data` command reads the {2} part of the file and the `background` command reads the {1} part or the spectrum.

The good channels are set based on the rules given in Valerie Connaughton's presentation at the Goddard Data Analysis Workshop in Dec 2010. These rules are to ignore energies below 8 keV and above 900 keV for the NAI detector and to ignore energies below 200 keV and above 40 MeV for the BGO detectors.

The fit to each model is done as follows. The default model parameter initial values are used. The `pgstat` statistic is used rather than the Cash statistic specified by Valerie. *The now recommended* `pgstat` *statistic assumes Gaussian error for the background.* The flux is computed using the convolution model `cflux` in XSPEC and is then multiplied by the duration of the spectrum to get the fluence. The model that gives the smalled value of the `pgstat` statistic divided by the number of degrees of freedom is taken to be the best-fit model.

2.11 Write a Summary

Write a neat table of results for this trigger.

3 Running the Script

The script was written in Python 2.6. The following non-system packages are required: `numpy`, `pyfits`, and `xspec`. The script operates on the data for an individual GBM trigger. To run the script first `cd` to the directory containing the trigger data.

```
> cd ~/gbm/triggers/2008/bn080825593/current
```

Next, run the script DO_GBM.PY in this directory. We recommend making a back-up of the `current` directory so that you can reprocess the original data if you are not happy with the automated processing. The DO_GBM.PY script currently takes four command line argument, which is the GBM trigger id.

```
> do_gbm.py trigger_id time_string detectors timezero
```

trigger_id - This is the GBM trigger ID of this trigger. For example, `bn080825593`.

time_strong - This is one of the following values:

- `NONE`—Let the script compute the source and background time intervals.
- `b1-b2:s1-s2:b3-b4`—These are the start and stop times of the first background interval, the source interval, and the second background intervals respectively. All six times are in MET.

detectors - This is one of the following values:

- `ALL`—Use all of the detector files in the working directory.
- `TRIGGER`—Use only the detectors that triggered for this burst.
- `nnnnnnnnnnnn`—Use the detectors specified in the `nnnnnnnnnnn` string.

timezero - This is one of the following values:

- `TRIGTIME`—Use TRIGTIME values in the detector files.
- `<user time>`—Set TRIGTIME to the specified time, in MET.

The script currently prints several status messages, most of which are intended for debugging. The last thing the script does is print out a summary of processing that looks something like this.

```
*****
Summary Information for bn080825593

                 Source: GRB080825593
           Trigger Time: 241366429.105 (MET)
         Best Data File: glg_tte_n9_bn080825593_v01.fit
Signal-to-Noise Per Bin: 6.07
                   T_90: 20.027 +/- 0.011 s (50-300 keV)
               Spectrum: Band Function
                  alpha: -0.71 +/- -1.00
                   beta: -2.99 +/- -1.00
                 E_peak: 203.2 +/- 157.2 keV
  fluence (10-1000 keV): -4.27e-02 +/- 0.00e+00 erg/cm^-2
```

Source - This is the value of the OBJECT keyword from the detector file with the best source detection.

Trigger Time - This is the value of the TRIGTIME keyword from the detector file with the best source detection.

Best Data File - This is the data file with the best detection.

Signal-to-Noise Per Bin This is the mean S/N per bin for the source light curve.

T_90 - This is the T_{90} value and 1-σ error in the given energy band. This is computed from the best detector.

Spectrum - This is the spectral model that gives the best fit.

alpha - This is the α value for Band model. For a cutoff power law alpha is the power-law index. Note that the sign convention may depend on the model.

beta - This is the β value for the Band model. For a cutoff power law beta is not defined.

E_peak - This is the E_{peak} value for the best-fitting model. The error is currently meaningless.

fluence - This is the fluence in the given energy range. This value is currently meaningless.

Plots showing the spectral fits for each spectral model are also produced. They are called `cutoffpl_bn080825593_xspec.ps` for the cutoff power law model and `grbm_bn080825593_xspec.ps` for the Band model. Two log files are also produced: `TT_gtburstfit.LOG` contains output from the GTBURSTFIT command while `bn080825593_xspec.LOG` is the XSPEC log file.

3.1 Output Files

For each detector the following files are created. If no source is found then some of these files will not be created.

glg_tte_XX_trigger_vNN.lc - This is the preliminary lightcurve file for detector XX.

glg_tte_XX_trigger_vNN_flatbkg.lc - This is the lightcurve file for detector XX with the background curvature subtracted.

glg_tte_XX_trigger_vNN.pha - This is the PHA2 spectrum file for detector XX.

For the detector with the "best" source detection the following files will be created.

glg_tte_XX_trigger_vNN_emin_emax.fit - This is the screened event file for detector XX. It contains events in the energy range [emin,emax].

glg_tte_XX_trigger_vNN_emin_emax.lc - This is the preliminary lightcurve file for detector XX in the [emin,emax] energy range.

glg_tte_XX_trigger_vNN_emin_emax_flatbkg.lc - This is the lightcurve file for detector XX in the [emin,emax] energy range with the background curvature subtracted.

glg_tte_XX_trigger_vNN_emin_emax_tbins.fits - This file contains the time intervals for the background and source spectra.

The following files are also created.

cutoffpl_trigger_xspec.ps - A postscript plot of the cutoff power law model fit to the spectral data.

grbm_trigger_xspec.ps - A postscript plot of the Band model fit to the spectral data.

trigger_xspec.log - The XSPEC log file.

trigger_xspec_save.xcm - The XSPEC save file. This can be used to rerun XSPEC using the command `xspec - trigger_xspec_save.xcm`.

TT_gtburstfit.LOG - This is an intermediate file that can be discarded.

3.2 Magic Numbers

There are a few "magic numbers" embedded in the code. These are described here.

If the user specifies to only use the triggered detectors then the DET_MASK keyword is read from the file `glg_tcat_all_TRIGGER_vNN.fit` where TRIGGER is the trigger id and NN is the version number. This is in GET_DETECTORS.

The script assumes that the detector file filenames have the form `glg_STRING-_DETECTOR_TRIGGER_vNN.fit` where STRING is the data type (TTE, CSPEC, &c.), DETECTOR is the detector id (n0,. . . ,nb,b0,b1), TRIGGER is the trigger id, and NN is the version number. This is in GET_LIST_OF_DETECTOR_FILES.

The background is fit with a 2nd order polynomial. This can be changed by changing the value of `polynomial_order` in FIT_BACKGROUND.

The second background interval must have a minumum length of one second. This is set with `min_block_duration` in GET_TIME_INTERVALS.

The maximum number of source bins with a negative S/N before the source is assumed to have ended is currently set to three. This is set with `max_consecutive-_negative_sn` in COMPUTE_SIGNAL_TO_NOISE.

The response matrix files are assumed to have the form `glg_cspec_DETECTOR-_TRIGGER_vNN.rsp` where DETECTOR is the detector id, TRIGGER is the trigger id, and NN is the version number. This is in FIND_RESPONSE_FILE.

XSPEC is set to ignore energies <8 keV and >900 kev for the NAI detectors and <200 keV and >40 MeV for the BGO dectors. These are set in FIT_SPECTRA.

The energy range for the T_{90} calculation is $[50, 300]$ keV. The energy range for the fluence calculation is $[10, 1000]$ keV. These are set at the start of MAIN with `e_t90_min`, `e_t90_max`, `e_flu_min`, and `e_flu_max`.

The minimum S/N per bin required for a source signal to be accepted as real is currently 1.5. This is set with `threshold_sn` in MAIN.

4 Comparison with the Fermi GBM Burst Catalog

The script has been run for all the 489 GRBs contained in the Fermi GBM Burst Catalog (`http://heasarc.gsfc.nasa.gov/W3Browse/fermi/fermigbrst.html`). A comparison of some of the temporal and spectral quantities (namely, the T_90, the spectral slopes α and β, the integrated flux and fluence) is presented in an accompanying document that can be found in the User Contributed section of the Fermi Science Support Center website (`http://fermi.gsfc.nasa.gov/ssc-/data/analysis/user/`). The comparison shows that there are systematic differences for some quantities (in particular the T_90) when estimated using the script DO_GBM.PY or the method adopted to generate the Fermi GBM Burst Catalog. This implies that the user should run different types of analysis and carefully select the results that best model the data.

5 Functions

5.1 main

```
"""Run basic GRB processing on Fermi/GBM data.

This script takes a Fermi/GBM burst trigger directory as input.
It constructs light curves, identifies the burst, computes burst
durations, and fits a spectrum to the burst.  The analysis is
fully automated and is intended to provide a quick look analysis
of a Fermi/GBM trigger.

Usage:
do_gbm.py trigger_id timestring detectors timezero

trigger_id -- The Fermi/GBM trigger id of this burst.
timestring -- NONE tells the script to identify the source
              and background intervals automatically.
              b1-b2:s1-s2:b3-b4 where b1, b2, s1, s2, b3, b4
              are the start and stop times of the first
              background interval (b1,b2), the source interval
              (s1,s2), and the second background interval
```

```
                (b3,b4) respectively, in MET seconds.
detectors  -- ALL tells the script to use all the detectors,
              TRIGGER tells the script to use only the
              detectors that were triggered by the burst,
              nnnnnnnnnnnn is a string of 0s and 1s where
              1 indicates that this detector is to be used.
              The string starts with the n0 detector and ends
              with the nb detector.
timezero   -- TRIGTIME tells the script to set the trigger time
              to the value of the TRIGTIME keyword in the input
              detector file.
              x.y tells the script to use the time x.y (MET
              seconds) as the trigger time.

"""
```

5.2 get_times

```
"""Split a string of times into background and source intervals.

   Usage: times = get_times(time_string)

   Input:
      time_string -- (string) b1-b2:s1-s2:b3-b4 where b1,b2 are
                     the stop and start times of the first
                     background interval; s1,s2 are the start and
                     stop times of the source interval; and b3,b4
                     are the start and stop times of the second
                     background interval.

   Output:
      times -- (list) A list containing the six times that define
               the three intervals.  Returns None if time_string
               cannot be parsed into six floats.
"""
```

5.3 is_number

```
"""Check if a string is a number."""
```

5.4 get_detectors

```
"""Parse the detectors string.

   Usage: detector_files = get_detectors(detector_string, trigger_name)
```

```
    Input:
       detector_string -- (string) ALL - Use all the detectors that
                          had a detection.
                          TRIGGER - Use the detectors that
                          triggered.
                          nnnnnnnnnnnn - Use detectors specified by
                          the user.

       trigger_name    -- (string) Fermi/GBM trigger id

    Output:
       files -- (list) List of detector files, or None if none were
                found

    If detector_string = TRIGGER this function reads the DET_MASK
    keyword from the glg_tcat_all file to identify the triggered
    detectors.

"""
```

5.5 get_specified_detectors

```
"""Create a list of detector files to use.

    Usage: files = get_specified_detectors(trigger_name, detector_files,
                                           detmask)

    Input:
       trigger_name   -- (string) Fermi/GBM trigger id
       detector_files -- (list) List of detector files to check
       detmask        -- (string) Which detectors to use

    Output:
       files -- (list) List of detector files to use

"""
```

5.6 get_list_of_detector_files

```
"""Return Fermi/GBM detector files in the current directory.

    Usage: files = get_list_of_detector_files(trigger_name)

    Input:
       trigger_name -- (string) Fermi/GBM trigger id
```

```
   Output:
      files -- (list) List of detector files

"""
```

5.7 run_gtbin_lightcurve

```
"""Call gtbin to generate a Fermi/GBM light curve.

   Usage: run_gtbin_lightcurve(detector_file, lightcurve_file,
                               tstart, tstop, dtime)

   Input:
      detector_file   -- (string) Fermi/GBM detector file name
      lightcurve_file -- (string) Output lightcurve file name
      tstart          -- (float) Start time for lightcurve in seconds (MET)
      tstop           -- (float) Stop time for lightcurve in seconds (MET)
      dtime           -- (float) Bin width for lightcurve in seconds

   Output:
      Returns None if gtbin ran without generating an error.
      Returns an error string if there was a problem.

"""
```

5.8 time_intervals

```
"""Determine background and source intervals.

   Usage: intervals = time_intervals(lightcurve_file, trigtime)

   Input:
      lightcurve_file -- (string) lightcurve file name
      trigtime        -- (float) trigger time of the burst

   Output:
      intervals -- (list) The start and stop times, in MET, of the
                   three intervals (first background, source,
                   second background).  Returns None if no
                   intervals were found.

   This function finds preliminary intervals using the input light
   curve then uses the two background intervals to fit and
   subtract the background.  It then finds refined intervals using
   the background-subtracted light curve.
```

```
"""
```

5.9 fit_background

```
"""Fit the background for a light curve.

   Usage: background = fit_background(time, counts, trigtime, intervals)

   Input:
      time      -- (list) time in MET seconds
      counts    -- (list) counts corresponding to time
      trigtime  -- (float) trigger time of burst
      intervals -- (list) the background and source time intervals

   Output:
      background -- (list) the fitted background corresponding to
                    time

   This function fits a polynomial to the two background
   intervals.  The source interval is not used in the fit.

"""
```

5.10 subtract_background_curvature

```
"""Fit and subtract curvature from a Fermi/GBM lightcurve.

   Usage: new_lightcurve =
          subtract_background_curvature(lightcurve, trigtime,
                                        intervals)

   Input:
      lightcurve -- (string) lightcurve filename
      trigtime   -- (float) burst trigger time
      intervals  -- (list) source and background intervals

   Output
      new_lightcurve -- (string) background-subtracted lightcurve
                        file name

   Read the light curve, then fit and subtract the background.
   The mean background is added back to the light curve to
   preserve Poisson statistics.  The final light curve is written
   to a new light curve file.
```

```
"""
```

5.11 get_time_intervals

```
"""Return background and source start and stop times intervals.

   Usage: (bkg1_start,bkg2_stop,src_start,src_stop
           bkg2_start,bkg2_stop) =
          get_time_intervals(event_file)

   Input:
      event_file -- (string) Fermi/GBM detector event file name

   Output:
      bkg1_start -- (float) background interval 1 start time (MET)
      bkg1_stop  -- (float) background interval 1 stop time (MET)
      src_start  -- (float) source interval start time (MET)
      src_stop  -- (float) source interval stop time (MET)
      bkg2_start -- (float) background interval 2 start time (MET)
      bkg2_stop  -- (float) background interval 2 stop time (MET)

  Call gtburstfit to determine Bayesian blocks for the lightcurve.
  The first background interval is the first Bayesian block.  The
  last time interval is the last Bayesian block unless it is
  shorter than min_block_duration, then it is the last two
  Bayesian blocks.  The source interval is the interval between
  the two background intervals.

"""
```

5.12 compute_signal_to_noise

```
"""Compute the signal-to-noise ratio of a lightcurve.

   Usage: sn = compute_signal_to_noise(lightcurve_file, intervals
                                        user_times, trigtime)

   Input:
      lightcurve_file -- (string) name of the lightcurve file
      intervals       -- (list) start and stop times of the source
                         and background intervals
      user_times      -- (boolean) True if the intervals were
                         set by the user and False if the
                         intervals were determined by the script
      trigtime        -- (float) trigger time (MET s)
```

```
    Output:
       mean_sn         -- (float) the mean signal-to-noise ratio per
                          bin in the source interval
       intervals       -- (list) improved source and background
                          intervals

    This function reads the lightcurve file then fits and subtracts
    the background.  The signal-to-noise ratio is computed in each
    bin in the source interval then a mean signal-to-noise ratio is
    computed.  If user_times is False (the user did not specify the
    intervals) then the source interval is refined so that it ends
    when max_consecutive_negative_sn bins with negative signal-to-
    noise ratios are encountered.  The second backgroud region is
    also adjusted.

"""
```

5.13 compute_duration

```
"""Compute the T_fraction duration of the source.

    Usage: [t_fractiont_fraction, t_fraction_err,
           t_fraction_start, t_fraction_stop] =
           compute_duration(lightcurve_file, src_start, src_stop,
                            bkg_start, bkg_stop, fraction)

    Input:
       lightcurve_file -- (string) Name of lightcurve file
       src_start       -- (float) Start time of source (MET)
       src_stop        -- (float) Stop time of source (MET)
       bkg_start       -- (float) Start time of background (MET)
       bkg_stop        -- (float) Stop time of background (MET)
       fraction        -- (float) T_fraction value (0-100)

    Output:
       t_fraction       -- (float) T_fraction duration (s)
       t_fraction_err   -- (float) T_fraction 1-sigma error (s)
       t_fraction_start -- (float) Start time of T_fraction
                           interval (MET s)
       t_fraction_stop  -- (float) Stop time of T_fraction
                           interval (MET s)

    Computes the T_fraction (e.g. T_90, T_50) duration of a source
    as well as its 1-sigma error and the start and stop times of
    the T_fraction duration.  This function uses the method
    described in the battblocks documentation and the TOTVAR method
```

```
   to compute the error.

"""
```

5.14 select_on_energy

```
"""Extract a light curve in a user-specified energy band.

   Usage: select_on_energy(infile, emin, emax, outfile)

   Input:
      infile  -- (string) name of input events file
      emin    -- (float) minimum energy (keV)
      emax    -- (float) maximum energy (keV)
      outfile -- (string) name of output events file

   Call fselect to select events with energies in the
   range [emin,emax].

"""
```

5.15 create_time_bins_file

```
"""Create a time bin file for gtbin.

   Usage: create_time_bin_file(background_start_time,
                               background_stop_time,
                               source_start_time,
                               source_stop_time,
                               output_file_name)

   Input:
      background_start_time -- (float) in MET s
      background_stop_time  -- (float) in MET s
      source_start_time     -- (float) in MET s
      source_stop_time      -- (float) in MET s
      fits_file             -- (string) name of output time bins
                               file

   Call gtbindef to create a FITS file with the background and
   source intervals that will be used to extract spectra from
   the Fermi/GBM detector files.

"""
```

5.16 find_response_file

```
"""Find a .rsp file corresponding a .pha file.

   Usage: rsp_file = find_response_file(pha_file)

   Input:
      pha_file -- (string) Name of .pha file

   Output:
      rsp_file -- (string) Name of corresponding .rsp file

   This function finds all of the .rsp files in the current
   directory.  Then it looks for a .rsp file with exactly the
   same name as the .pha except that tte is replaced with cspec.
   Example: glg_tte_n6_bn121205000_v01.rsp and
            glg_cspec_n6_bn121205000_v01.pha

"""
```

5.17 fit_model_spectrum

```
"""Fit an XSpec model to spectral data.

   Usage: fit_model_spectrum(statmethod, xspec_save_file)

   Input:
      statmethod      -- (string) XSpec statmethod (e.g., chi,
                         cstat, pgstat)
      xspec_save_file -- (string) File listing the XSpec commands

   This function uses pyxspec to fit an XSpec model.  It assumes
   that the data and the model have already been set up.  This
   functions just does the actual fitting.

"""
```

5.18 fit_spectra

```
"""Do an XSpec spectral fit to Fermi/GBM burst data.

   Usage: [model_data, [goodness_of_fit, dof],
          [fluence, fluence_err]] = fit\_spectra(trigger_name,
          spectrum_list, response_list, fit_statistic
          e_flu_min, e_flu_max, duration, plot_file)
```

```
    Input:
       trigger_name  -- (string) Fermi/GBM trigger id
       spectrum_list -- (list) list of .pha files to use
       response_list -- (list) of .rsp files to use
       fit_statistic -- (string) XSpec fit statistic to use
       e_flu_min     -- (float) minimum energy for fluence
                        calculation (keV)
       e_flu_max     -- (float) maximum energy for fluence
                        calculation (keV)
       duration      -- (float) duration of the spectrum (s)
       plot_file     -- (string) name of output plot file

    Output:
       model_data      -- (list) XSpec model information
                          [name, par1, par1_err, par2, par2_err, ...]
       goodnes_of_fit -- (float) goodness of fit value
       dof             -- (float) degrees of freedom in fit
       fluence         -- (float) fluence (erg cm^-2)
       fluence_err     -- (float) error in fluence (erg cm^-2)

    This function does the spectral fitting using pyxspec.  It
    currently tries to fit a Band model and a cutoff power law
    model and chooses the one with the best fit.

"""
```