An Error Calculation Tool for Fits of Gamma-ray Burst (GRB) Spectral Energy Distributions

Reference Manual

Version 1.0

Saeeda Sajjad & Khadeejah Motiwala

Department of Space Science,
Institute of Space Technology, 1, Islamabad Highway,
Islamabad 44000, Pakistan

Space and Astrophysics Research Lab,
National Centre of GIS and Space Applications,
Islamabad 44000, Pakistan

14 October 2021

Contents

Foreword

In this manual, examples are in `monotype face`. Part A of the manual contains a brief explanation of the underlying principles of the tool. Part B is the user's guide to operating the tool.

The tool itself consists of two Python files. The first one, `ErrorCalcTool.py` is the program that the user interacts with. The secondary file `multiple_SEDs_loop.py` is the backend program which contains the calculator. The user is not required to edit parts of this file.

1 Introduction

This is a software tool that calculates and plots errors in Spectral Energy Distributions (SEDs) in Gamma-ray Burst analyses. The tool inputs FITS files which contain the results of spectral fits. In particular, the tool utilizes the covariance matrix which contains covariances of correlated parameters of a particular spectral model. The tool's output is a SED butterfly plot with complete error propagation.

2 Part A: Basic Concepts

2.1 Spectral Models

The application of empirical functions to the analysis of gamma-ray burst (GRB) spectra plays an important role in understanding their fundamental nature. These functions, called spectral models, are fitted to spectral energy distributions (SEDs). In order to obtain a true fit, two or more models are often used in conjunction, and are referred to as the base model and additional components. Following are some of the more commonly used models in GRB spectral analyses.

2.1.1 Power Law

$$f_1 = A(E/E_{piv})^{\lambda}$$

Parameters:

1. A = amplitude in $photons s^{-1} cm^{-2} keV^{-1}$
2. E_{piv} = pivot energy in keV
3. λ = index

2.1.2 Smoothly Broken Power Law

$$m = \frac{\lambda_h - \lambda_l}{2}$$
$$b = \frac{\lambda_h + \lambda_l}{2}$$
$$\alpha_{piv} = \frac{log_{10}(E_{piv}/E_b)}{\Delta}$$
$$\beta_{piv} = m\Delta log_e \frac{exp(\alpha_{piv}) + exp(-\alpha_{piv})}{2}$$
$$\alpha = \frac{log_{10}(E/E_b)}{\Delta}$$
$$\beta = m\Delta log_e \frac{exp(\alpha) + exp(-\alpha)}{2}$$
$$f_4 = A(E/E_{piv})^b 10^{(\beta - \beta_{piv})}$$

Parameters:

1. A = amplitude in $photonss^{-1}cm^{-2}keV^{-1}$
2. E_{piv} = pivot energy in keV
3. λ_l = lower index
4. E_b = break energy in keV
5. Δ = break scale in decades of energy
6. λ_h = upper index

2.1.3 Band's Function, E_{peak} parameterization

$$f_5 = A(E/100)^{\alpha} exp(-E(2+\alpha)/E_{peak})$$

if

$$E < (\alpha - \beta)E_{peak}/(2+\alpha)$$

and

$$f_5 = A(\alpha - \beta)E_{peak}/[100(2+\alpha)]^{(\alpha-\beta)} exp(\beta - \alpha)(E/100)^{\beta}$$

if

$$E \geq (\alpha - \beta)E_{peak}/(2+\alpha)$$

Parameters:

1. A = amplitude in $photonss^{-1}cm^{-2}keV^{-1}$
2. E_{peak} = peak energy in keV
3. α = low-energy index
4. β = high-energy index

2.1.4 Comptonized, E_{peak} parameterization

$$f_7 = Aexp[-E(2+\lambda)/E_peak](E/E_{piv})^{\lambda}$$

Parameters:

1. A = amplitude in $photonss^{-1}cm^{-2}keV^{-1}$
2. E_{peak} = peak energy in keV
3. λ = index
4. E_{piv} = pivot energy in keV

2.1.5 Black Body

$$f_{17} = A\frac{E^2}{exp(E/kT) - 1}$$

Parameters:

1. A = amplitude in $photonss^{-1}cm^{-2}keV^{-1}$
2. kT = temperature in keV

2.2 Error Propagation

The Gamma-ray Spectral Fitting Package (RMfit)[1] calculates best-fit parameters for user-chosen models and outputs a FITS (Flexible Image Transport System) file which contains information about the quality of the fit. In order to calculate the propagation of errors in multiple models, the following information is required from the FITS file:

1. covariances between all the different parameters involved (i.e. covariance matrix)
2. values of the parameters
3. errors in the parameter values

As is usually the case, when parameters are correlated, then covariance must be considered. The variance-covariance matrix of x = $(x_1$,..., $x_n)$ is

$$\begin{pmatrix} \sigma_1^2 & \sigma^{12} & \sigma_{13} & \cdots \\ \sigma_{21} & \sigma_2^2 & \sigma_{23} & \cdots \\ \sigma_{31} & \sigma_{32} & \sigma_3^2 & \cdots \\ \vdots & \vdots & \vdots & \ddots \end{pmatrix}$$

Moreover, the standard deviation σ_f of any non-linear differentiable function $f(a, b)$ of two variables a and b can be given by[2]:

$$\sigma_f^2 \approx |\frac{\partial f}{\partial a}|^2\sigma_a^2 + |\frac{\partial f}{\partial b}|^2\sigma_b^2 + 2\frac{\partial f}{\partial a}\frac{\partial f}{\partial b}\sigma_{ab}$$

where σ_a is the standard deviation of a, σ_b is the standard deviation of b and $\sigma_{ab} = \sigma_a\sigma_b\rho_{ab}$ is the covariance between a and b.
This tool calculates σ_f for each model component and then uses it to obtain the total sum of errors in the best-fit model.

[1] https://fermi.gsfc.nasa.gov/ssc/data/analysis/rmfit/

[2] https://en.wikipedia.org/wiki/Propagation_of_uncertainty

3 Part B: User Guide

3.1 About the tool???

This error calculation tool is a small Python program compatible with Python 3.2 and Python 2.7. This first version has been tested on the command line.

3.2 Data Files

3.2.1 Input Directories

FITS files can be obtained from RMfit, and should be named following the pattern:

```
BASEMODEL_ADDITIONALCOMPONENT1_ADDITIONALCOMPONENT2_EAC.fit
```

for example:

```
BAND_BB_EAC.fit
```

If the Effective Area Correction (EAC) is not present in the particular fit, it should be omitted in the name.

To set up the directory, the user must supply the paths for each of the intervals in the `directories` list. In an example where there are three intervals, it would look like this:

```
directories = ['/home/fits/interval1', '/home/fits/interval2', '/home/fits/interval3']
```

3.2.2 Output Files

The tool outputs a butterfly SED, displaying individual model components, as well as the best-fit model. The user must define the output directory in `output_directory`. For example:

```
output_directory = '/home/outputs'
```

The output figures are saved in PNG (Portable Graphics Format) and/or PDF (Portable Document Format) formats. The default output file name is in the pattern `'SED_component1_component2.png'` but the user has the option to change it by simply replacing `'SED_'+str_allmodels` in lines `88` and `89` with a name of their choice.

3.3 Commands

3.3.1 Setting time intervals for all model components

After defining the input directories for each model component, the user must supply the intervals of time which each of them span. The beginning times for each component are stored in the list `Tstart_all` and the ending times are stored in Tend_all. For example, if there are three intervals for three components,

```
Tstart_all = [3.8, 4.5, 6.3]
Tend_all = [4.5, 6.3, 9.4]
```

3.3.2 Defining the models

The base model and components for each time interval should be defined by the user in the list models. For example:

```
models = ['BAND_BB', 'BAND_PL', 'CPL_PL']
```

3.3.3 Setting the Error Area Correction (EAC) term

If the fit contains an EAC term, then the variable eac should be set accordingly:

```
eac = True
```

or else,

```
eac = False
```

3.3.4 OPT_plot_limit_vE_highest

The user must specify if there is an upper limit to the energy intervals for the models. If so,

```
OPT_plot_limit_vE_highest = True
```

and the user must supply these upper limits in highest_energy_photons. For example:

```
highest_energy_photons = [408.11203e3,  277.63062e3, 152.10425e3]
```

In the case that

```
OPT_plot_limit_vE_highest = False
```

then the default Energy intervals are defined by the minimum and maximum energies for plotting the SEDs, E1 and E2 respectively, which are also supplied by the user and are in units of kEV.

3.4 Customizing the plots

3.4.1 Model component colors

The user should define distinct colors for each of the model components in color_list. For example:

```
   color_list = [’navy’, ’red’, ’turquoise’]
```

3.4.2 Linestyles

The plotting line style for the models and components can be defined in `model_linestyle` and `component_linestyle`, respectively. Some types of linestyles in `matplotlib` are `’--’`, `’:’`, and `’-’`. For example:

```
model_linestyle = ’-’

component_linestyle = ’:’
```

3.4.3 Linewidths

Similarly, the user can choose the linewidths for the models, components, and total model, in `model_linewidth`, `component_linewidth`, and `totalmodel_linewidth` respectively.

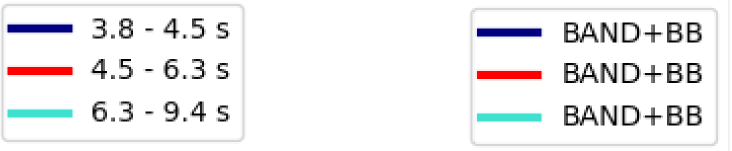
3.4.4 OPT_LEGEND

The user has the option to choose whether the figure legend should contain time intervals (`’interval’`) or model names (`’model’`). For example:

```
OPT_LEGEND = ’interval’
```

or,

```
OPT_LEGEND=’model’
```



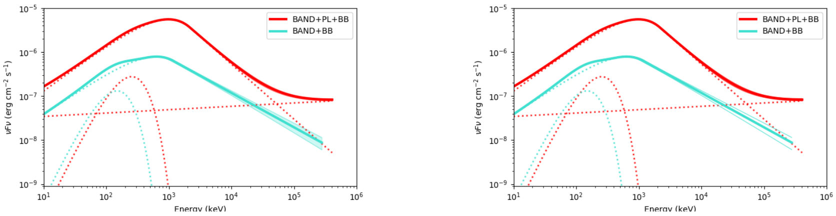
(a) OPT_LEGEND=’interval’

(b) OPT_LEGEND=’model’

Figure 1: Two different options for legend text.

3.4.5 OPT_SHADE

The user can also choose to shade the confidence interval for the models by defining `True` or `False` to the variable `OPT_SHADE`.



(a) OPT_SHADE = True

(b) OPT_SHADE = False

Figure 2: Unshaded and shaded confidence interval options using OPT_SHADE command.

3.4.6 x-axis and y-axis limits

The x-axis and y-axis minimum and maximum limits can be set by the user in lines 54 to 57. For example:

```
xlim_min = 10
xlim_max = 1.e6
ylim_min = 9.e-10
ylim_max = 1.0e-5
```

3.5 A complete example

What follows is a complete example of the user-defined parts of the tool.

```
directories=['/home/user/directory1/', '/home/user/directory2/']
output_directory='/home/user/outputdirectory/'

Tstart_all = [5.6, 7.3]
Tend_all = [7.3,9.5]
OPT_plot_limit_vE_highest = True
highest_energy_photons = [ 408.11203e3,  277.63062e3]

models = ['BAND_PL_BB','BAND_BB']

eac = True

E1=1.
E2=7.

color_list=['red', 'turquoise']
component_linestyle=':'
component_linewidth=2.

model_linestyle= '-'
model_linewidth = 1.

totalmodel_linewidth = 3.

OPT_LEGEND = 'model'
OPT_SHADE = False

xlim_min=10.
xlim_max=1.e6
ylim_min=9.e-10
ylim_max=1.0e-5

fig = plt.figure(1,figsize=(7.,4.))

plt.legend(loc='upper right')

plt.xlabel('Energy (keV)')
plt.ylabel(r'$\nu F\nu$ (erg cm$^{-2}$ s$^{-1}$)')

plt.savefig(output_directory+'fig1'.png',bbox_inches="tight")
```