The ~~Fermi~~ ~~GBM~~ Gamma-ray Data Tools

Adam Goldstein
AGoldstein@usra.edu

USRA

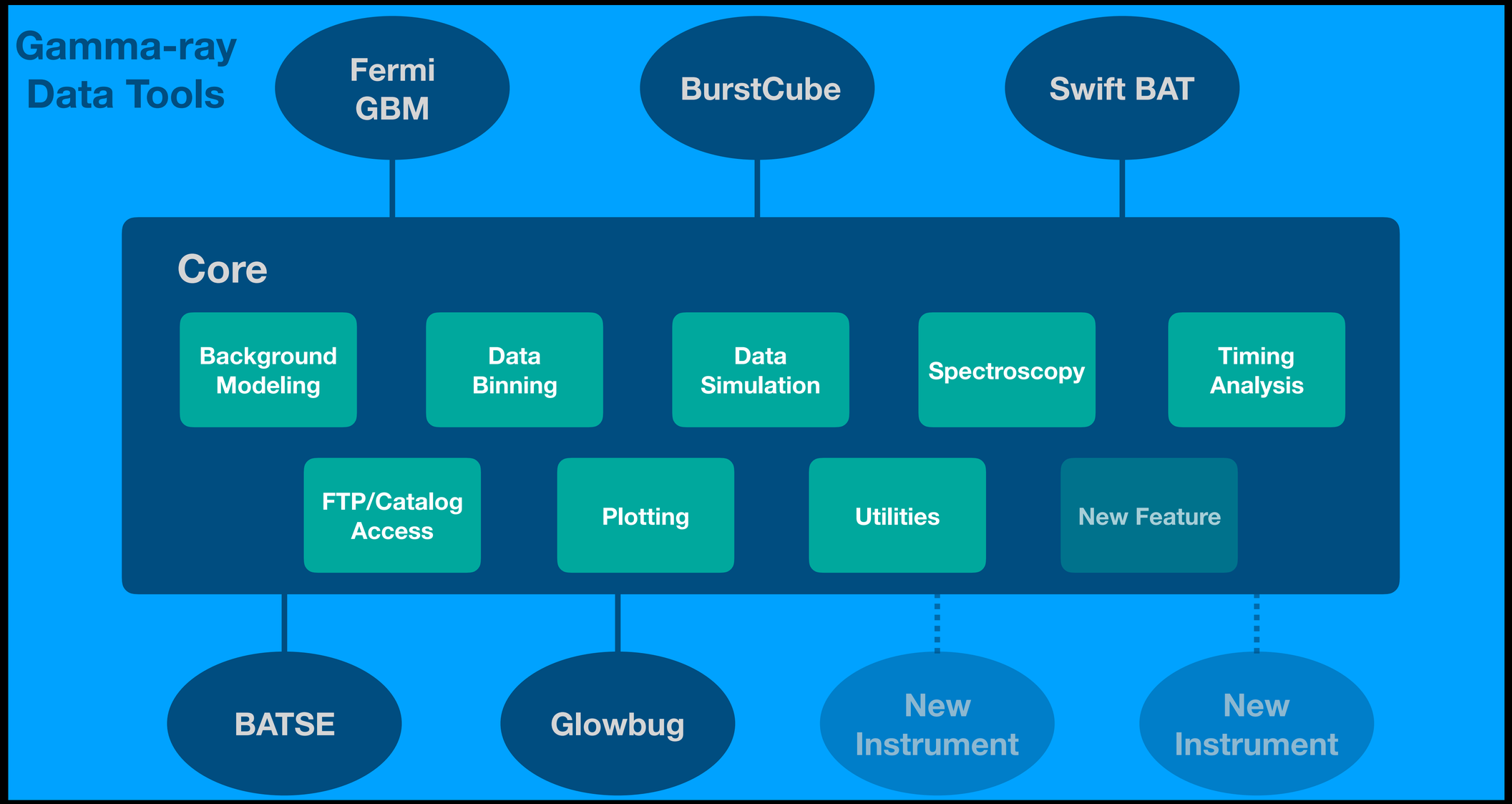
The Gamma-ray Data Tools (GDT)

A Python toolkit for space-based Hard X-ray and Gamma-ray Mission Data

https://github.com/USRA-STI/

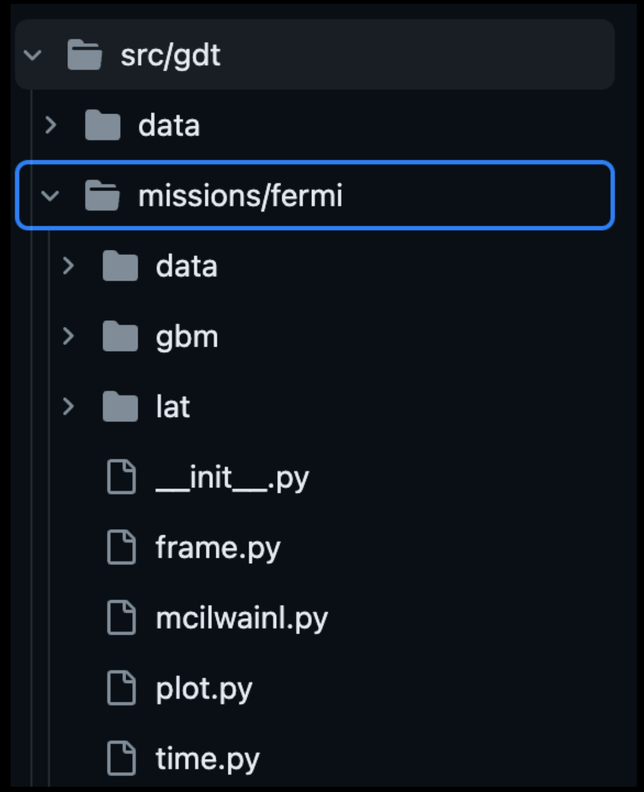
- Generalized version of the Fermi GBM Data Tools
- Interface to science and auxiliary data files
- Sufficiently **high-level** part of the API so that it is easily accessible to many, but also **lower-level** part of the API for expert users
- Reduce and Analyze data (binning, background estimation)
- Export/conversion of data
- Observing conditions — Source visibility, GTIs, detector angles, etc
- Spectral analysis
- Simulations
- Wide range of visualizations
- Interface to HEASARC FTP archive and Browse Catalogs
- Full documentation and unit tested

GDT Structure



Namespace Packages

- GDT utilizes **namespace packages**
- gdt-core is the central library: gdt.core
- The source directory structure is organized as the following (e.g.):
  - src/gdt/missions/fermi
  - src/gdt/missions/swift/bat
- Mission packages are accessed within the gdt namespace as the following:
  - gdt.missions.fermi
  - gdt.missions.swift.bat
- This enables each mission toolkit to be maintained and used separately, with only a dependency on gdt-core.
- Instructions on how to setup your own package in the gdt namespace is in the README of gdt-core.



Installing Packages

- Packages on PyPI as astro-gdt. E.g. astro-gdt-fermi

```
$ pip install astro-gdt-fermi
```

- Data for unit testing/tutorials is not included. There is a script that will download the data to a standard directory. This initializes that directory.

```
$ gdt-data init
```

- Download test/tutorial data:

```
$ gdt-data download fermi-gbm
```

- Within python, the data can be accessed in this way:

```
> from gdt.core import data_path
> gbm_path = data_path.joinpath('fermi-gbm')
```

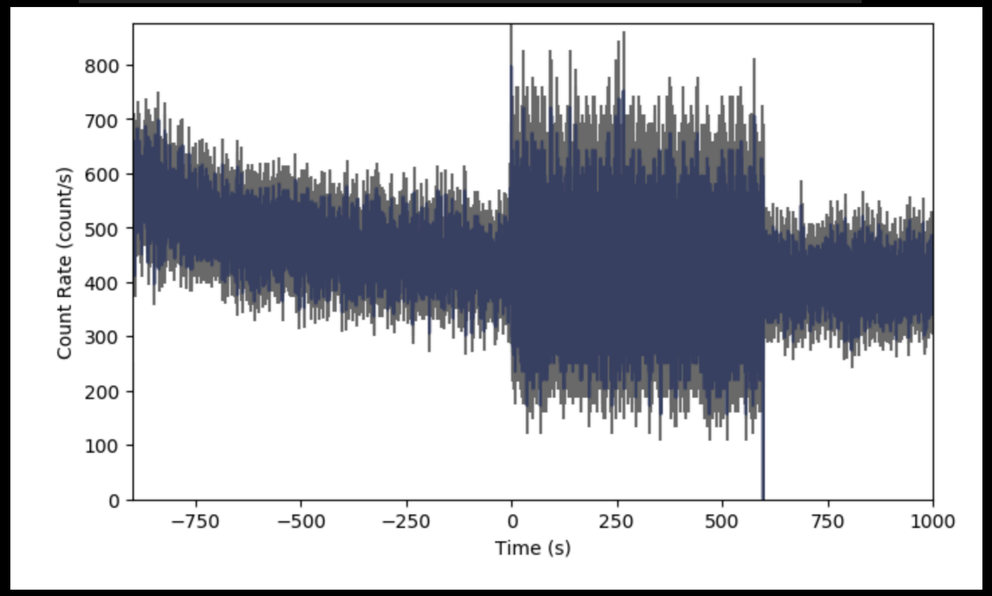
High-Level API — Lightcurves

Read a file and convert to lightcurve

```
from gdt.core import data_path
from gdt.missions.fermi.gbm.phaii import GbmPhaii
gbm_path = data_path.joinpath('fermi-gbm')
filepath = gbm_path / 'glg_ctime_nb_bn120415958_v00.pha'
ctime = GbmPhaii.open(filepath)
lightcurve = ctime.to_lightcurve(energy_range=(50.0, 500.0))
```

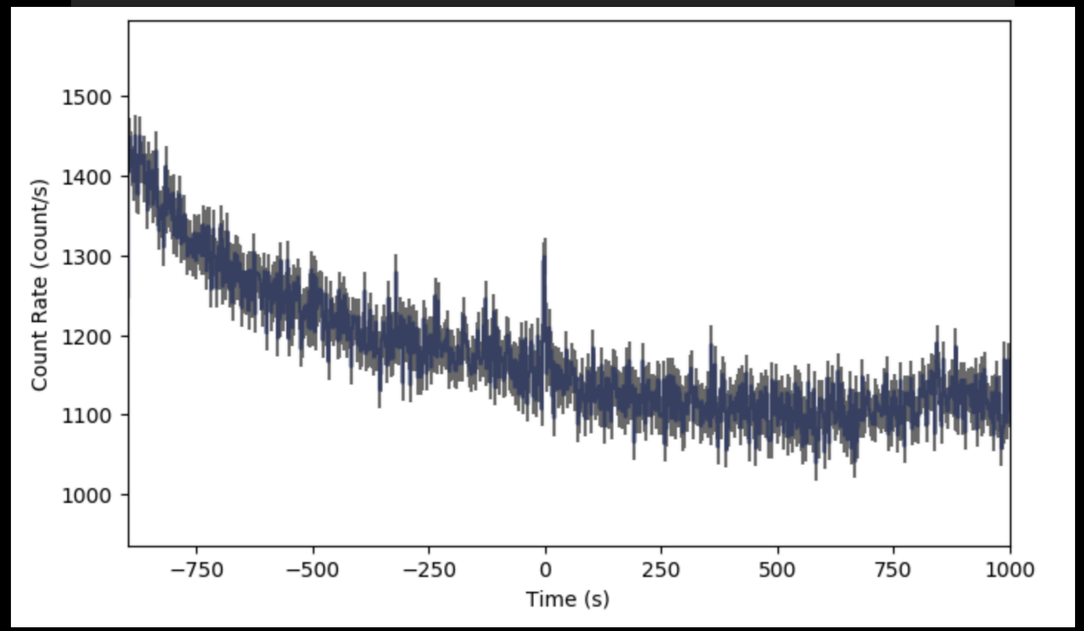
Plot it

```
from gdt.core.plot.lightcurve import Lightcurve
lcplot = Lightcurve(data=lightcurve)
plt.show()
```



Rebin it

```
from gdt.core.binning.binned import rebin_by_time
rebinned_ctime = ctime.rebin_time(rebin_by_time, 2.048)
lcplot = Lightcurve(data=rebinned_ctime.to_lightcurve())
```



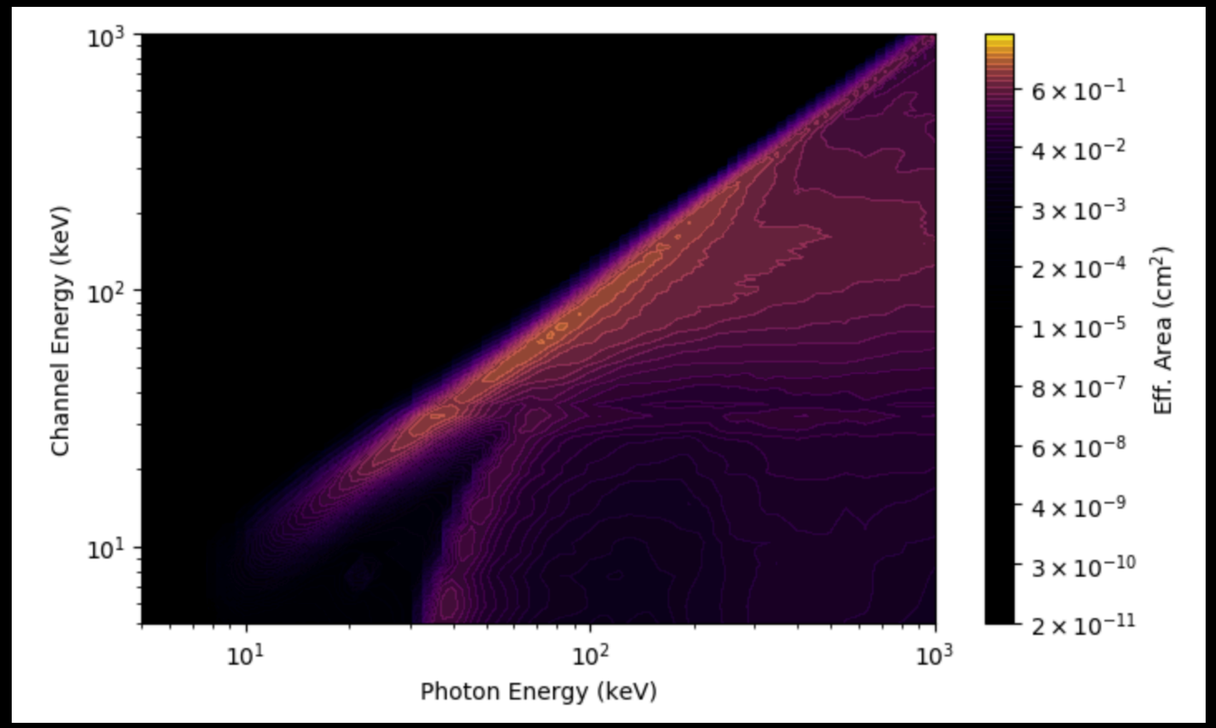
Detector Responses

Read a Response file

```
from gdt.missions.fermi.gbm.response import GbmRsp2
filepath = gbm_path / 'glg_cspec_n4_bn120415958_v00.rsp2'
rsp2 = GbmRsp2.open(filepath)
# extract a single DRM (at T0=0.0)
rsp =  rsp2.nearest_drm(0.0)
```

Plot the DRM

```
from gdt.core.plot.drm import ResponsePlot
rsp_plot = ResponsePlot(rsp.drm)
plt.xlim = (5.0, 1000.0)
plt.ylim = (5.0, 1000.0)
```



Fold a photon model through the response

```
from gdt.core.spectra.functions import PowerLaw
pl = PowerLaw()
# amplitude=0.01, index=-2.0
count_spectrum = rsp.fold_spectrum(pl.fit_eval, (0.01, -2.0))
print(count_spectrum.rates)
        array([0.20455526, 0.24133158, 0.20801155, 0.15628108,
                0.15712484, 0.19561199, 0.21861904, 0.26870772,,...])
```

Observing Conditions

Read a position history file

```
from gdt.missions.fermi.gbm.poshist import GbmPosHist
filepath = gbm_path / 'glg_poshist_all_170101_v01.fit'
poshist = GbmPosHist.open(filepath)
sc_frames = poshist.get_spacecraft_frame()
```

Is a position visible at some time?

```
from gdt.missions.fermi.time import Time
time = Time(504975500, format='fermi')
one_frame = frame.at(time)

from astropy.coordinates import SkyCoord
coord = SkyCoord(324.3, -20.8, unit='deg')
one_frame.location_visible(coord)
    True
```

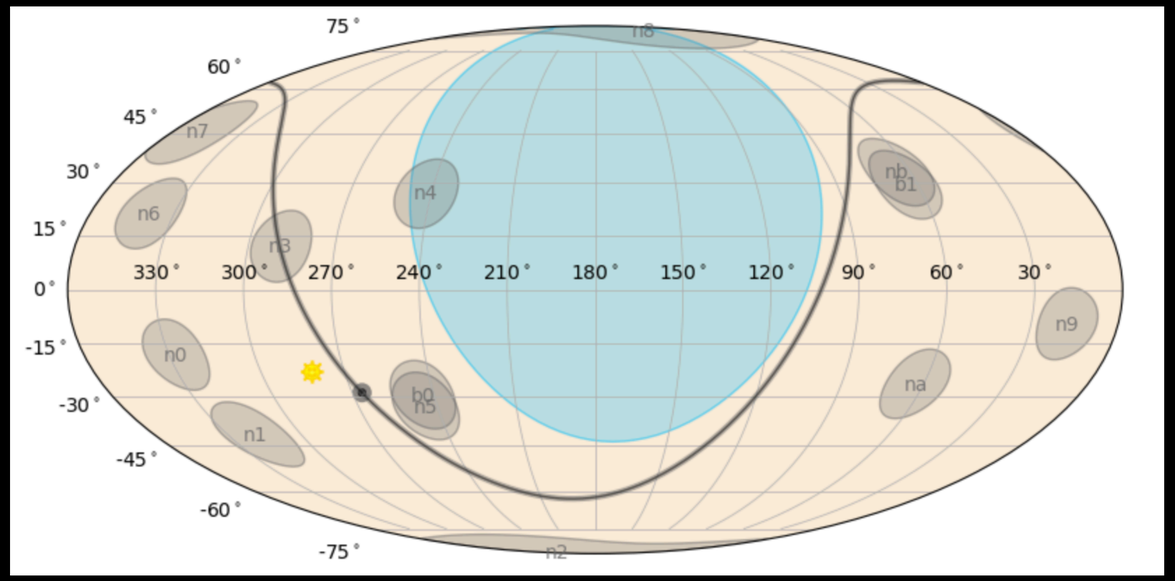
Angle of the position to detector n0:

```
one_frame.detector_angle('n0', coord)
    <Angle [4.27219806] deg>
```

Observing Conditions

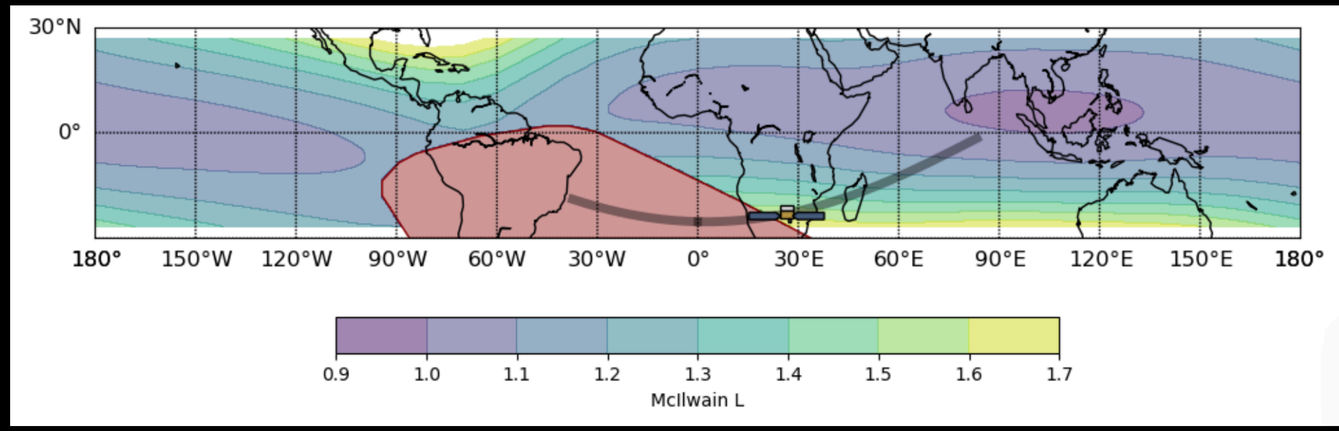
Plot the detector pointings

```
from gdt.core.plot.sky import EquatorialPlot
eqplot = EquatorialPlot()
eqplot.add_frame(one_frame)
plt.show()
```



Plot the orbital position

```
from gdt.missions.fermi.plot import FermiEarthPlot
from gdt.missions.fermi.gbm.saa import GbmSaa
tstart = Time(time.fermi - 1000, format='fermi')
tstop = Time(time.fermi + 1000, format='fermi')
earthplot = FermiEarthPlot(saa=GbmSaa())
earthplot.add_spacecraft_frame(sc_frames, tstart, tstop, trigtime=time)
plt.show()
```



Localizations

Read a HEALPix localization file

```
from gdt.missions.fermi.gbm.localization import GbmHealPix
filepath = gbm_path / 'glg_healpix_all_bn190915240_v00.fit'
loc = GbmHealPix.open(filepath)
```

Plot the localization

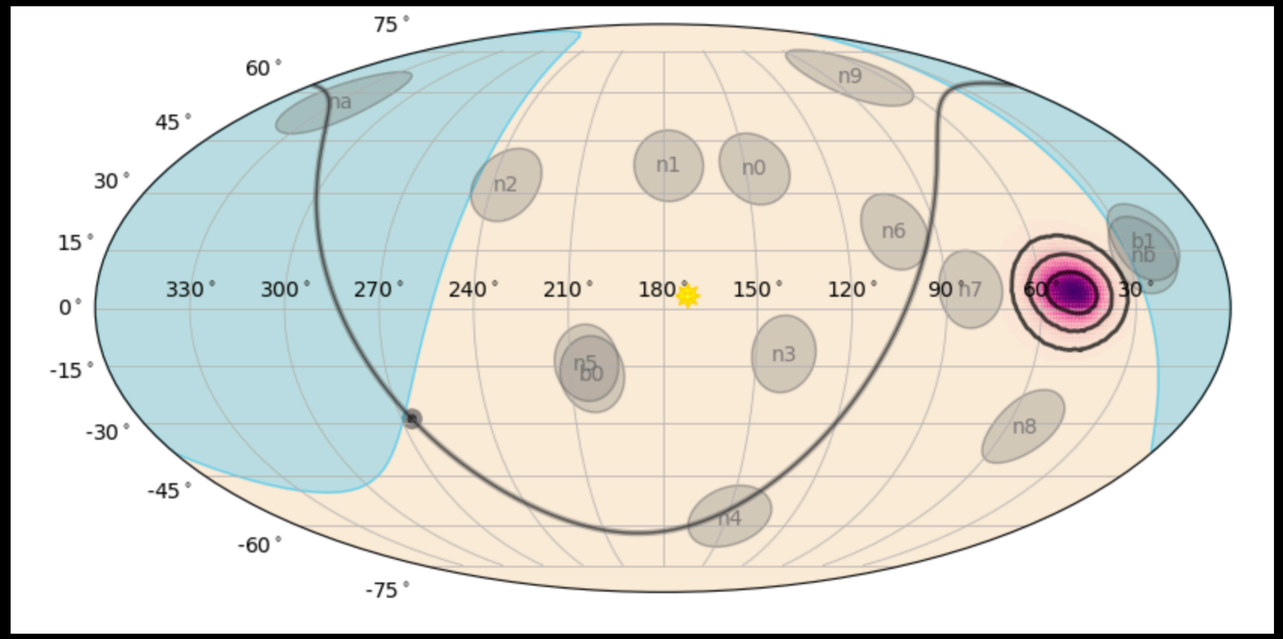
```
from gdt.core.plot.sky import EquatorialPlot
eqplot = EquatorialPlot()
eqplot.add_localization(loc)
plt.show()
```

The confidence level at a point

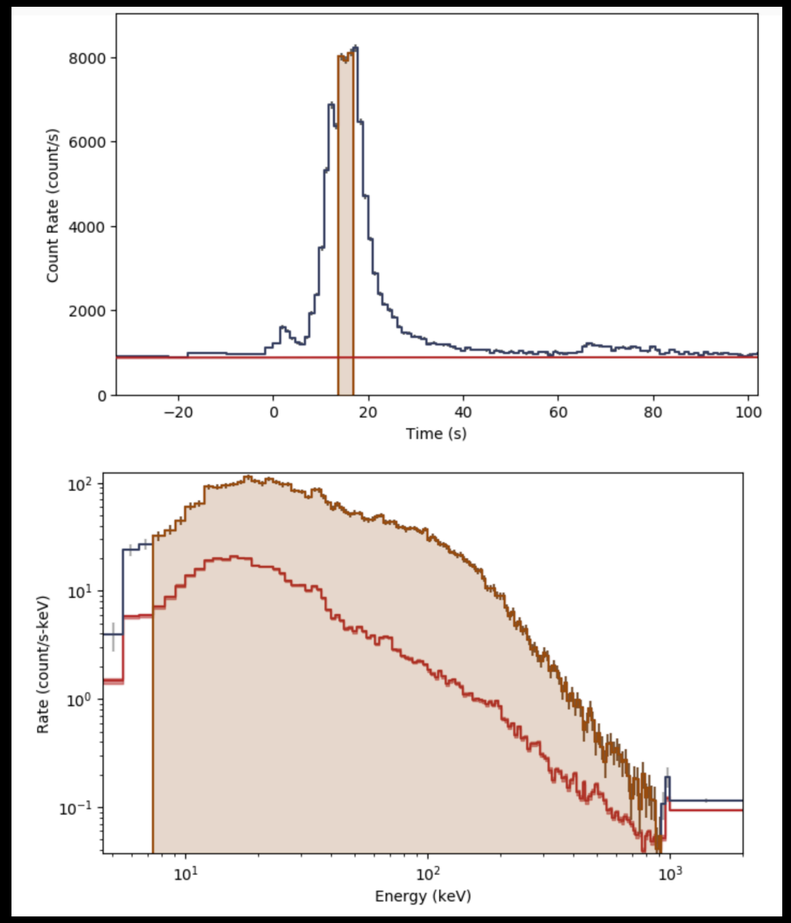
```
loc.confidence(40.0, 4.0)
  0.865783539232832
```

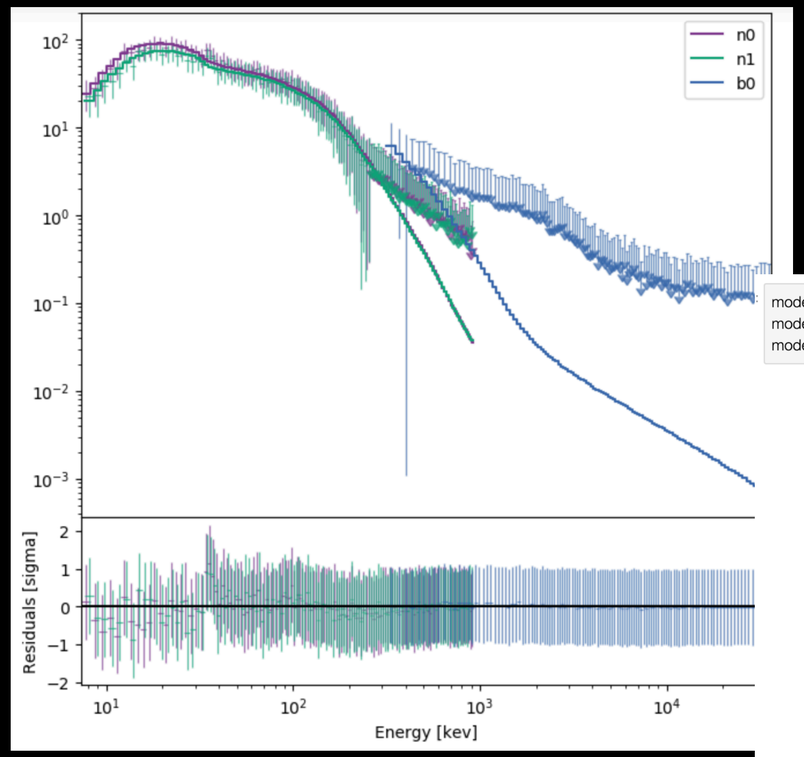
Area of the 90% conf. region

```
loc.area(0.9)
  281.1633711457409
```



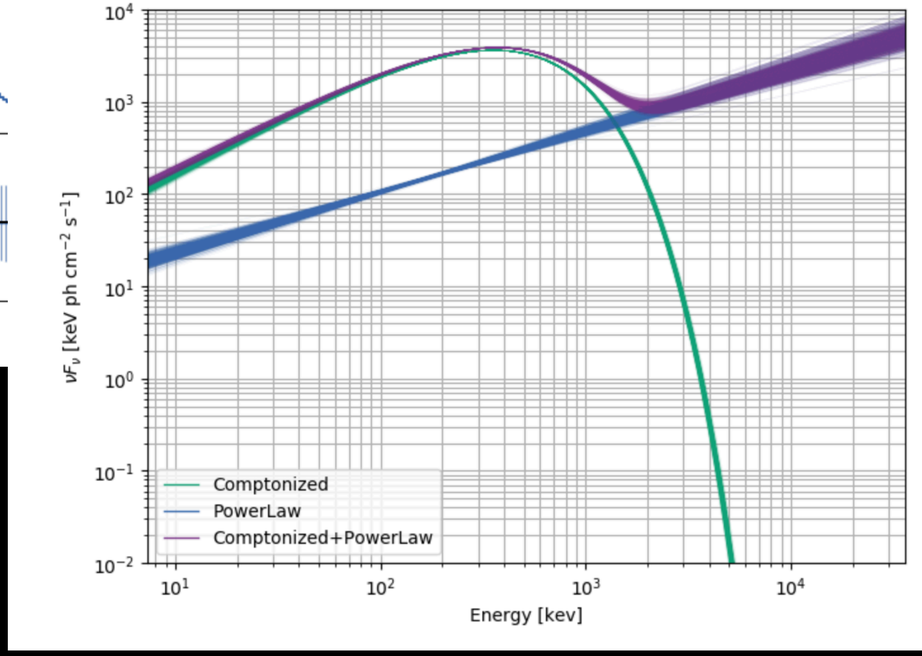
Backgrounds, Selections, & Spectral Fitting





Can fit multiple components, plot the fit, and the spectrum for each component

```
modelplot = ModelFit(fitter=specfitter, view='nufnu')
modelplot.ylim = (0.01, 10000.0)
modelplot.ax.grid(which='both')
```

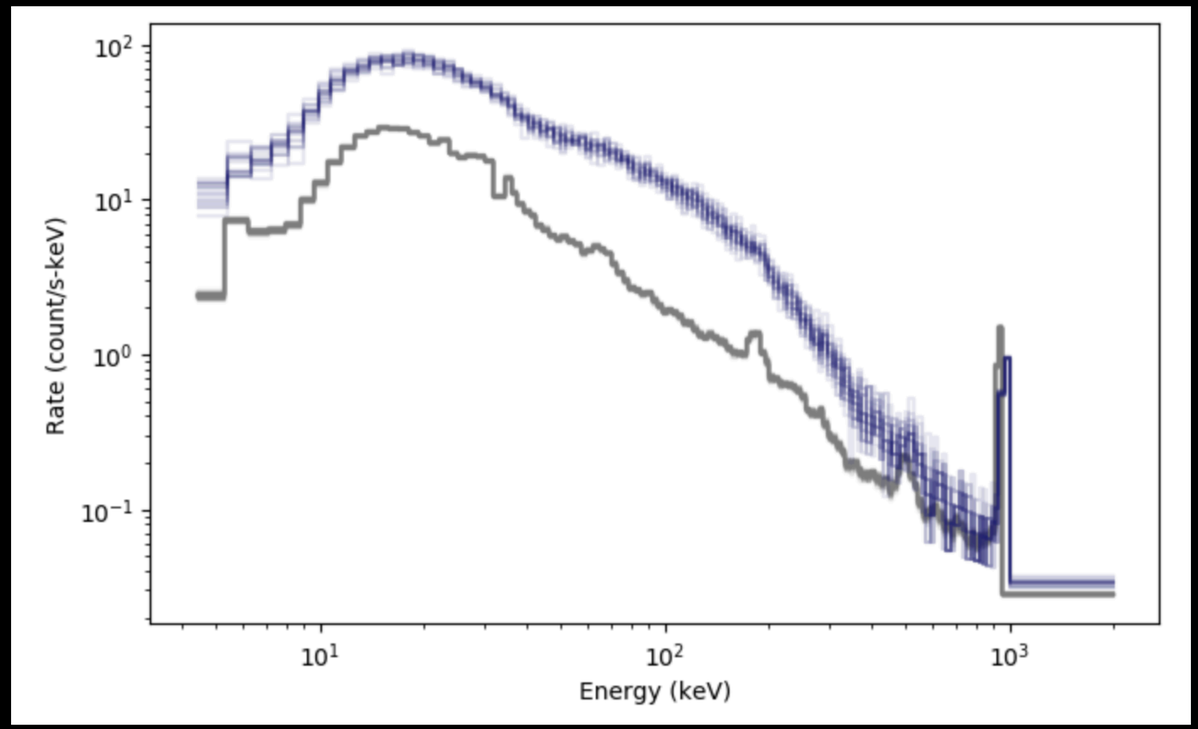


MLE with Chisq, Cstat, Pstat, or PGstat
Uses SciPy minimizers

Spectral Simulations

Simulate a spectrum (20 sims shown)

```
from gdt.core.simulate import PhaSimulator
from gdt.core.spectra.functions import Band
band = Band()
band_params = (0.01, 300.0, -1.0, -2.8)
exposure = 0.256
pha_sims = PhaSimulator(rsp, band, band_params, exposure,
                        spec_bkgd, 'Gaussian')
```



Temporal-Spectral Simulations

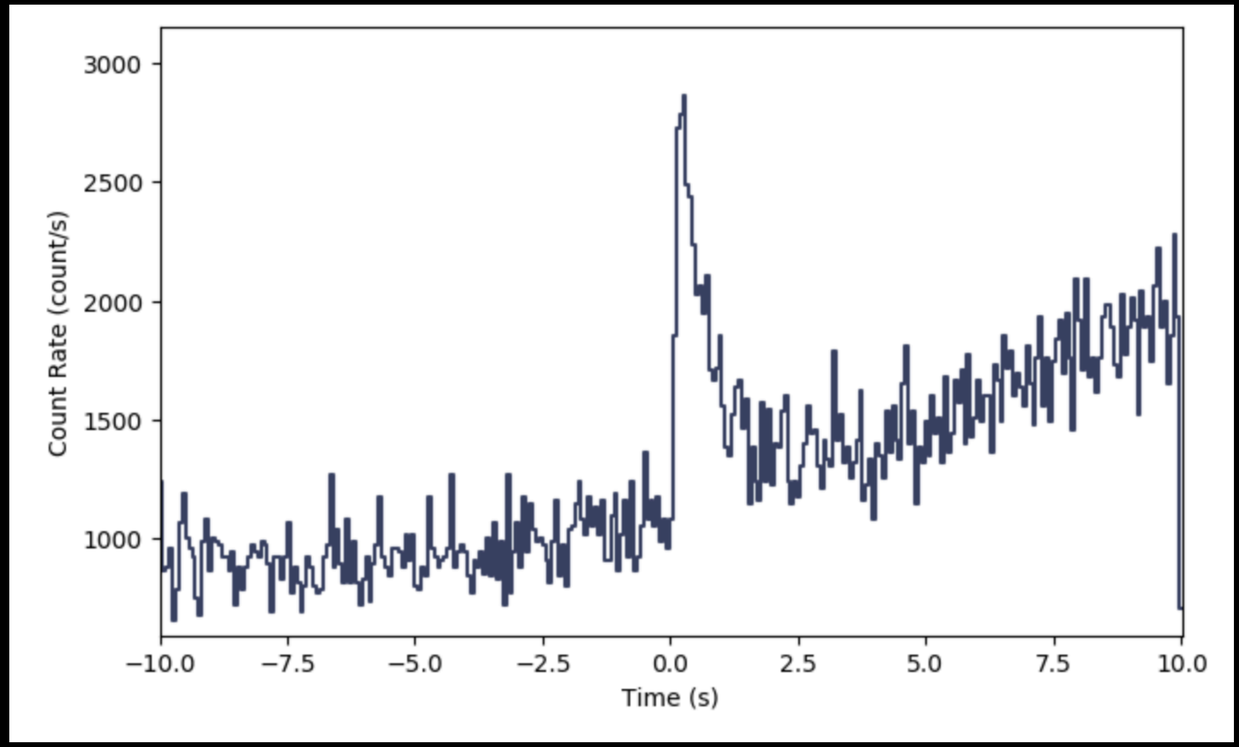
```
from gdt.core.simulate import *
from gdt.core.simulate.profiles import norris, quadratic
norris_params = (0.05, 0.0, 0.1, 0.5)
quadratic_params = (1.0, 0.05, 0.003)

# source simulation
tte_sim = TteSourceSimulator(rsp, band, band_params, norris, norris_params)
tte_src = tte_sim.to_tte(-5.0, 10.0)

# background simulation
tte_sim = TteBackgroundSimulator(spec_bkgd, 'Gaussian', quadratic,
                                 quadratic_params)
tte_bkgd = tte.sim_to_tte(-10.0, 10.0)

# merge the background and source
from gdt.missions.fermi.gbm.tte import GbmTte
from gdt.core.binning.unbinned import bin_by_time
tte_total = GbmTte.merge(tte_bkgd, tte_src)

# bin to 64 ms resolution
phaii = tte_total.to_phaii(bin_by_time, 0.064)
lcplot = Lightcurve(data=phaii.to_lightcurve())
plt.show()
```



Mission Packages

https://github.com/USRA-STI/

Funded by:

- NASA Astrophysics Data Analysis Program (ADAP)
- NASA SMD Open Source Tools, Libraries, and Frameworks
- NASA MSFC ISFM Directed Work Package

Legacy

CGRO/BATSE ✅

HETE-2/FREGATE ✅

RXTE/ASM ✅

SUZAKU/WAM⚡

Current

AstroSat/CZTI

Fermi/GBM ✅

INTEGRAL/SPI

INTEGRAL/SPI-ACS⚡

MAXI/GSC ✅

Swift/BAT ✅

Upcoming

BurstCube

Glowbug

StarBurst

✅ Publicly released

⚡ Imminent

Contributors

USRA: A. Goldstein, W. Cleveland, C. Fletcher, O. Roberts, S. Bala

ADNET Systems: J. Asercion

LSU: E. Burns

NASA/GSFC: I. Martinez-Castellanos (UMD), T. Parsotan

NASA/MSFC: C. M. Hui, D. Kocevski, C. Wilson-Hodge, J. Wood

Penn State: J. Delaunay

UAH: M. S. Briggs

Univ. of Toronto: A. Tohuvavohu