Introduction to Fermi LAT Data Analysis

Jeffrey Morais

McGill University

August 2020

Contents

1 Preliminaries

1.1 LAT Telescope & Cherenkov Radiation

Before getting into the specifics and results of the data analysis performed, we must briefly introduce what is being analysed and why. One of NASA's telescopes, the Fermi Gamma-Ray Space Telescope (FGST), monitors astrophysical events in the gamma-ray energy range from low Earth orbit. [19] Its main instrument, the Large Area Telescope (LAT), operates as an all-sky survey telescope with a period of 3 hours, detecting high energy particles whose energy ranges from 20MeV to over 300GeV. [17] Gamma rays-emitted in that energy range are associated with extreme conditions, such as superluminous supernovae detected though gamma-ray bursts (GRBs), pulsars, and quasars. [16] This is of particular interest when those violent events emit high energy charged particles which can pass through a dielectric [†] medium and emit electromagnetic radiation, which is known as Cherenkov radiation. This occurs when the radiation enters the medium at a speed faster than the phase velocity, the speed at which the wave propagates through the medium, of light. [1]

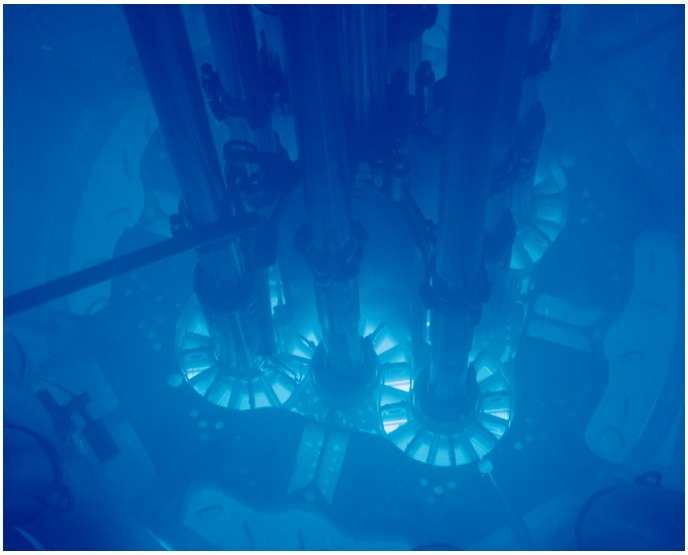

Figure 1: Cherenkov radiation emission in the core of the Advanced Test Reactor at the Idaho National Laboratory. [22]

This process is analogous to how sound behaves in the sonic boom effect. [7] The radiation that is relevant in our case is in the gamma-ray energy range, which allows us to obtain information about the specific event. The Very Energetic Radiation Imaging Telescope Array System (VERITAS) directly images short flashes of Cherenkov radiation by observing clusters of relativistic charge particles that collide with the atmosphere to produce gamma-rays. [4] The main focus of this paper, however, will consist of data analysis on publicly available data from the LAT, as is discussed in the following section.

[†]An insulator that can be polarized.

1.2 Objectives

The research project itself doesn't have a specific terminal objective, instead it focuses on developing data analysis and coding proficiency relevant to Fermi data. More specifically, it revolves around performing statistical analyses on very high energy (VHE) gamma-ray sources using the public data from the LAT. This includes performing maximum likelihood fits on potentially transient or extended objects to build models of the specified region of the sky to extract parameter values, or compute upper limits on non-detections to test the sensitivity of the LAT instrument. This is useful in drawing conclusions on the interactions of gamma-rays and particles, such as Cherenkov radiation. Moreover, it also fixates on reproducing results from published or currently drafting research papers, which will be presented at the end of the paper.

2 Fermitools: Tutorials

The following tutorials and results are using the latest version of Fermitools 1.2.23, an open source package distributed by NASA for data analysis on the LAT. The package comprises of various shell commands that are being run directly from the terminal in the Ubuntu 20.04.1 LTS Linux distribution, which proved less problematic than macOS. Although the commands will be explicitly shown at times, most of these are run on either shell or Python scripts that can be made by the user. For the Python side of things, there are no specific environments being used, so it is encouraged to abandon the use of Python 2 for scripting. Furthermore, the visualization of the data is using the software SAOImage DS9 version 8.1.

The purpose of this section is to demonstrate and guide through results that have been replicated from tutorials offered by the Fermi Science Support Center (FSSC) website. Although trivial, it will be noted that all graphics presented are results generated by the author and not the website.

2.1 Counts Map

The first order of business is producing count maps for your data being that it's the simplest way to visualize whats going on in terms of the events of interest in the selected region of the sky. It should be noted that for this paper all of the visualizations will be using the Aitoff projection. In essence, a counts map partitions the sky into rectangular bins where the amount of photons/counts in each bin will determine its magnitude that is represented by different colour mappings. To start off, the data must be extracted from the LAT Data Query, where the coordinates of the event, search radius, observation dates, and energy range are specified. It should be noted that the coordinate system we are interested in is the J2000 equatorial coordinate system that uses right ascension (RA) and declination (DEC) as the basis, essentially longitude and latitude projected onto the sky. [20] The search radius refers to the steradian of sky being observed in square degrees. The photon files retrieved from the query contain information of the intensity and distribution of the counts, whereas the spacecraft file contains the position of the spacecraft. These can be obtained more efficiently using the `wget` shell command, used in the proper working directory.

The first part in preparing your data is letting the tools know the cuts you made to the data, meaning the specifications you inputted when selecting your data on the site. Furthermore, the `tmin` and `tmax` parameters are in mission elapsed time (MET), referring to the launch of the FGST. The preparation consists of everything

you specified on the site, although additionally including the event class, in which case we chose SOURCE (128) as we mostly deal with point sources, and event type which changes the reconstruction of the data, which is chosen to be the default value 3. Furthermore, it also asks of a maximum zenith angle cut, and due to Earth's limb which might clutter the data with extra background events, a cut of 90° is suggested. This done using the `gtselect` tool, which looks like the following:

```
time gtselect evclass=128 evtype=3 infile=@events.txt
outfile=example_sel.fits ra= dec= rad= tmin= tmax= emin= emax=
    zmax=90
```

The blanks are meant to be replaced by numerical values pertaining to the source of interest. The `time` parameter at the beginning is simply to record the execution time of the command. Next, what is important to take into consideration is the position of the spacecraft as it oscillates during each period around Earth. Ignoring it would cause warped counts maps that reduce its legibility. For this we use the `gtmktime` tool which orients the data properly according to the spacecraft position. This intakes the following inputs: the spacecraft file, a data filter (which we will use the default input), and whether you want to apply additional cuts to your region of interest, which isn't necessary in this case. This will look something like:

```
time gtmktime scfile=spacecraft.fits
    filter="(DATA_QUAL>0)&&(LAT_CONFIG==1)" roicut=no
evfile=example_sel.fits outfile=example_gti.fits
```

Note that the files outputted from this function are known as good time interval (gti) event files. Finally, comes the formation of the counts map. For this we use the `gtbin` tool. This intakes the coordinates of the event, projection method, coordinate system (celestial), bin scaling, binning algorithm (counts map), and the size of the square surrounding the counts map, which can be calculated using trigonometry and dividing by the bin scaling. This is executed as follows:

```
gtbin evfile=example_gti.fits scfile=spacecraft.fits
    outfile=example_cmap.fits algorithm=CMAP nxpix=
nypix= binsz= coordsys=CEL xref= yref= axisrot=0 proj=AIT
```

Here the `nypix` and `nxpix` inputs refer to the square size, and the `axisrot` is attributed to the rotation of the image, which we want none of. Once completed, the outputted file can be opened and analyzed in ds9. We present two examples in the following graphics:

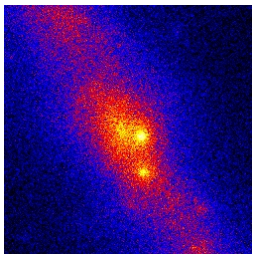

Figure 2: Cygnus X-3, x-ray binary system with energy range 100MeV to 50GeV and search radius of 10 square degrees.

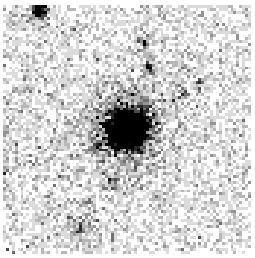

Figure 3: Markarian 421, blazar with energy range 100MeV to 5GeV and search radius of 10 square degrees.

Next we move onto producing counts map of the entire sky using LAT weekly data.

2.2 Full Sky Map

The procedure for producing all-sky counts maps is analogous to what was done in the previous section, with a few minor alterations. Instead of using the LAT query which has a max search radius of 60 square degrees, we will be employing the LAT Weekly Data Query, which has a max search radius of 180 square degrees instead. Furthermore, it is more convenient in this case to use galactic coordinates to position our data at the galactic center, the position of which will be simply (0,0).

Using the same commands in the previous section with the updated data, we produce the following result:

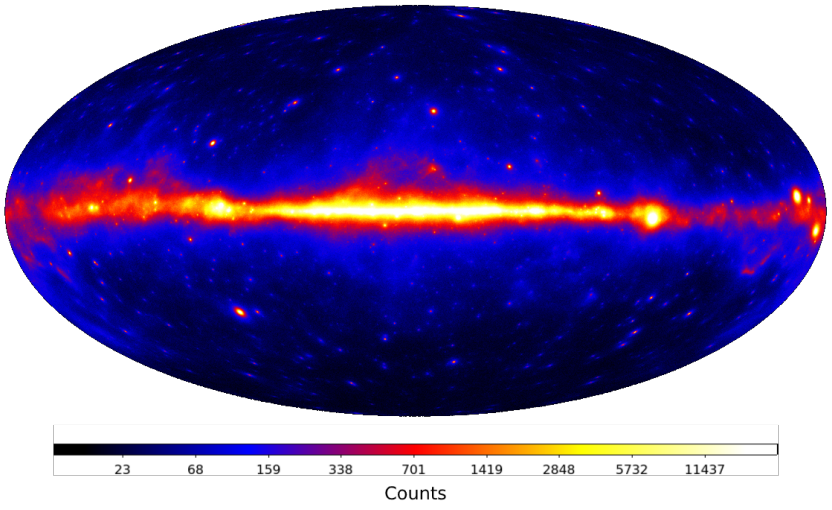

Figure 4: All-sky counts map evaluated at the galactic center, with energy range of 100MeV to 50GeV, using the 'min-max' ds9 algorithm, and logarithmic scaling.

The colour bar under the graphic represents the counts that a pixel bin has. The algorithm for assigning count values to each pixel is 'min-max' on the ds9 application, with a logarithmic scaling. It should be noted that unless specified, any data presented in the Fermitools tutorials is using the 'min-max' algorithm, with a square root scaling. [‡]The production of counts map will be useful in analysing sources for more complicated analyses, which will be discussed in the following section.

2.3 Unbinned Likelihood Analysis

Before jumping into performing an unbinned likelihood estimator on LAT data, some important concepts should be first covered.

2.3.1 The Test Statistic

One essential notion in characterizing the significance of an event is its test statistic. Consider the probability of obtaining data given an input model, denoted as the likelihood $\mathcal{L}$. [3] Maximizing $\mathcal{L}$ thus provides the best estimate of the data, given a model. One can bin a given data set, the number of events or counts observed in a given region of the sky for example, to represent the data via a counts map. The likelihood itself is defined as a product of probabilities:

$$\mathcal{L} = \prod_{\mu} \frac{m_{\mu}^{d_{\mu}} e^{-m_{\mu}}}{d_{\mu}!}, \tag{1}$$

where d_{μ} are the probabilities observing detected counts and m_{μ} are the counts predicted by the model. The product can be split by factoring out the exponential term, and due to the addition rule of exponents, the multiplied exponentials will sum over the counts [†], and results in the following expression:

$$\mathcal{L} = e^{-\Gamma_p} \prod_{\mu} \frac{m_{\mu}^{d_{\mu}}}{d_{\mu}!}, \tag{2}$$

where Γ_p is the total predicted number of counts, that have been summed over. Taking the limit as the bin sizes become infinitesimally small, the probabilities of observing counts in each bin, d_{μ}, collapses to 0 or 1. If the value is 0, the the likelihood simply becomes the exponential value $e^{-\Gamma_p}$, but if 1, the likelihood takes on its simplest form:

$$\mathcal{L} = e^{-\Gamma_p} \prod_{\nu} m_{\nu}, \tag{3}$$

[‡]It should be noted that although there is no documentation online explaining these algorithms, it is very likely that 'min-max' refers to having the color map scaling range from the minimum and maximum of the data set in terms of counts, while 'z-scale' refers to taking the whole possible range of counts.

[†]$\prod_{\mu} e^{-m_{\mu}} = e^{-\sum_{\mu} m_{\mu}}$.

where the index now products over the number of photons. Note that this applies only to an unbinned analysis, for a binned analysis, the index doesn't change. Moreover, it is often simpler to deal with the logarithm of the likelihood and then raising it as an exponential:

$$\log \mathcal{L} = \sum_{\nu} \log(m_\nu) - \Gamma_p. \tag{4}$$

The test statistic T_s itself is defined as the difference of logged likelihoods with and without the target source in the input model. This can be put into a more compact form:

$$T_s = -2 \log \left(\frac{\max(\mathcal{L}_0)}{\max(\mathcal{L}_1)} \right), \tag{5}$$

where $\mathcal{L}_0$ refers to a source model with the target source excluded, and $\mathcal{L}_1$ is for the model with the target source included. It is important to note that the significance of a source can be approximated to the square root of the test statistic. Now, this calculation is trivial given a direct input of the number of counts, but what is more is important is how we go about it with data that comes from telescopes, which is more complex than a string of counts values. We will consider this is the following subsection.

2.3.2 Analytical Computation of the Test Statistic

To properly compute the test statistic, we must construct the model in terms of potential event sources as a function of energy, time, and direction, which is as follows:

$$S(E, \hat{\rho}, t) = S_G(E, \hat{\rho}) + S_D(E, \hat{\rho}) + \sum_{\mu} S_\mu(E, t)\delta(\hat{\rho} - \hat{\rho}_\mu) + \sum_{\nu} S_\nu(E, \hat{\rho}, t), \tag{6}$$

where S_G and S_D are the galactic diffuse and isotropic models respectively of the sky which are time invariant. S_μ are the point sources contributing to the region of interest (ROI) of the sky Ω, and S_ν accounts for other classes of significant sources. E refers to the energy of the sources, t the time of the event, and $\hat{\rho}$ the incident direction of the γ rays in a celestial frame of reference [6].

To obtain the equivalent of the individual predicted number of counts m_μ, we must first consider the instrument response functions [†] (IRFs) R in terms of its components. Consider its representation with respect to the parameter space $(E', \hat{\rho}', t)$:

$$R(E', \hat{\rho}'; E, \hat{\rho}, t) = A(E, \hat{\rho}, \hat{O}(t))D(E'; E, \hat{\rho}, \hat{O}(t))P(\hat{\rho}'; E, \hat{\rho}, \hat{O}(t)). \tag{7}$$

Here we have the effective area $A(E, \hat{\rho}, \hat{O}(t))$, which is the product of the cross section area of the source region Λ [‡], the γ ray conversion probability, and the efficiency of a given event. Next we have the energy dispersion $D(E'; E, \hat{\rho}, \hat{O}(t))$, which is the

[†] The linear mapping between incident γ ray photon fluxes and detected events. [2]

[‡] A region encompassing the sources from the ROI, centered at the same event.

probability density of measuring event energies of γ rays. Finally we have the point-spread function $P(\hat{\rho}'; E, \hat{\rho}, \hat{O}(t))$, which is the probability density of reconstructing event energies for given γ rays. $\hat{O}(t)$ represents the normalized orientation of the spacecraft. To construct the predicted number of counts m_μ, we convolve the instrument response function with the source model over the source region, producing the following:

$$M(E', \hat{\rho}', t) = \int_\Lambda dE d\hat{\rho}\ R(E', \hat{\rho}'; E, \hat{\rho}, t) S(E, \hat{\rho}, t). \tag{8}$$

Note that $M(E', \hat{\rho}', t)$ now represents the total number of predicted counts. Furthermore, it is standard to exclude transient sources from the analysis, thus allowing us to remove time dependency from the source model, which implies $S(E, \hat{\rho}, t) \to S(E, \hat{\rho})$. Using the expression previously mentioned, we now have a new representation of equation (4):

$$\log \mathcal{L} = \sum_\nu \log(M(E'_\nu, \hat{\rho}'_\nu, t_\nu)) - \Gamma_p. \tag{9}$$

Finally, to compute the total number of predicted counts Γ_p, we must first calculate the exposure map [§] ε for the source region. To do so, the IRFs are integrated over the ROI with respect to each variable:

$$\varepsilon(E, \hat{\rho}) = \int_\Omega dE' d\hat{\rho}' dt\ R(E', \hat{\rho}'; E, \hat{\rho}, t). \tag{10}$$

This is convolved with the source model over the source region to give us our total number of predicted counts Γ_p:

$$\Gamma_p = \int_\Lambda dE d\hat{\rho}\ \varepsilon(E, \hat{\rho}) S(E, \hat{\rho}). \tag{11}$$

Thus concludes the computation of the test statistic given data from the LAT.

2.3.3 Likelihood Analysis

Now we are ready to proceed with the likelihood analysis. The process itself develops a model of the portion of the sky and performs as fit to extract parameters such as energy flux, or to plot the energy spectrum of sources within the ROI. The most prominent use of it is to compute test statistic maps to see the significance of each source, as well as looking at plots of counts/energy ratio as a function of energy, which demonstrates the behaviour of the source at different energy levels. One thing to note before beginning, we are starting with an unbinned likelihood analysis [†] which is more useful for when the number of events in each time bin is small as compared to a binned likelihood fit. [13]

[§] A map of the sky in physical coordinates that combines the effective area of the instrument with a map of the dwell time [21] (the time that the γ rays lingers in the effective area), to produce a map without artifacts caused by instrument imperfection. [8]

[†] The data handled during the likelihood fit is unbinned.

Like in the previous section, you begin with preparing the data with the proper cuts and time corrections. Next, to compute an exposure map we must first create a livetime cube, which calculates the dwell time of the detected counts for each source in the ROI over the selected time range [14]. This is accomplished using the `gtltcube` function which has a new parameter that intakes the cosine of the inclination angle between the source and detector. This will look something like:

```
time gtltcube zmax=90 evfile=example_gti.fits scfile=spacecraft.fits
    outfile=example_ltcube.fits dcostheta=0.025 binsz=
```

Next, we can move on into the computation of the exposure map. This is done utilizing the `gtexpmap` tool which in addition to the event and spacecraft files, intakes the source radius, the number of energy bands in consideration, and the number of longitude and latitude points that encompasses the source region. This can be computed via the following command string:

```
time gtexpmap evfile=example_gti.fits scfile=spacecraft.fits
    expcube=example_ltcube.fits outfile=example_unbin_expmap.fits irfs=
    srcrad= nlong= nlat= nenergies=
```

Following this we must now construct a source model in an XML format which will be used in future calculations. For unbinned analyses, it is standard to build it manually with so few sources of interest, but in the following section when considering a binned analysis, we will be utilizing a python scripts that automatically identifies all sources from the Fourth Gamma-ray LAT (4FGL) catalog and creates the source model.

The way in which information is assigned to each source in the model is the rate at which the amount of photons radiated change with varying energy levels, or simply dN/dE. This can take on many forms depending on what type of source is being fitted, the most common of which is the power law function [11], defined as follows:

$$\frac{dN}{dE} = N_0 \left(\frac{E}{E_0} \right)^{\gamma}, \tag{12}$$

where N_0 is known as the prefactor, γ is the spectral index, and E_0 is the scale parameter. This is inputted before performing the fit for each source, including the galactic diffuse and isotropic models of the sky. An example of such would be as follows:

```
<source name="4FGL J1036.5+1231" type="PointSource">
    <spectrum type="PowerLaw">
      <parameter free="1" max="1000" min="1e-05" name="Prefactor"
          scale="1e-14" value="0.9175000745" />
      <parameter free="1" max="5" min="0" name="Index" scale="-1"
          value="1.9729141" />
      <parameter free="0" max="3461.901611" min="3461.901611"
          name="Scale" scale="1" value="3461.901611" />
    </spectrum>
    <spatialModel type="SkyDirFunction">
      <parameter free="0" max="360" min="-360" name="RA" scale="1"
          value="159.148" />
      <parameter free="0" max="90" min="-90" name="DEC" scale="1"
```

```
        value="12.5227" />
    </spatialModel>
  </source>
```

Notice the name of the source 4FGL J1036.5+1231 is just notation where the J refers to the J2000 epoch the celestial coordinates are based on, and the 1036.5 and +1231 numbers are simply the right ascension and declination respectively, that can be expressed in an alternate form: 10:36:50, +12:31:00.

Now that we have a source model we can move onto the last step before computing the fit on the source of interest, which is computing diffuse responses. This corresponds to convolving the isotropic diffuse source within the model with the IRFs, which is done to save on computation time, instead of making the optimization function do it as well. The function performing the integration is the `gtdiffrsp` command, that takes the source model and response functions as inputs, which will look like the following:

```
time gtdiffrsp evfile=example_gti.fits scfile=spacecraft.fits
    srcmdl=example_input_model.xml irfs=CALDB
```

We now have all of the necessary steps to complete the likelihood optimization which is running the `gtlike` command. This will output the fit values of the sources such as the proper values of the parameters in the power law. It also outputs the test statistic, flux, and predicted number of photons (to name a few) for each source with free parameters. The command for beginning the optimization looks like the following:

```
time gtlike refit=yes plot=yes sfile=example_output_model.xml >
    fit_data.txt irfs=CALDB expcube=example_ltcube.fits
    srcmdl=example_input_model.xml statistic=UNBINNED
    optimizer=NEWMINUIT evfile=example_gti.fits scfile=spacecraft.fits
    expmap=example_unbin_expmap.fits
```

Notice no new inputs in the parameters of the command. Moreover, because the parameter `plot` has been enabled, another output of the `gtlike` function is a plot of the counts/MeV ratio as a function of MeV, which is plotted to demonstrate how well or poor the model has converged to the actual data. Note that the plot and the upcoming test statistic maps are presented only in the binned analysis section.

We may now also compute the test statistic and corresponding residual maps using the post-fitted source model using the `gttsmap`. Aside from file inputs, the function also asks for the projection method, and the size of the square that encompasses the source region, which will look like:

```
time gttsmap statistic=UNBINNED evfile=example_gti.fits
    scfile=spacecraft.fits expmap=example_unbin_expmap.fits
    expcube=example_ltcube.fits srcmdl=example_output_model_res.xml
    outfile=example_tsmap_resid.fits irfs=CALDB optimizer=NEWMINUIT
    nxpix= nypix= binsz= xref= yref= coordsys=CEL proj=AIT
```

The input source model will determine the type of map (residual or test statistic) depending on whether or not only the galactic models are taken into account. If not, then it will output a residual map, showing the difference between the data and the fitted model.

We now move onto a binned likelihood analysis with some alterations and additions

to the commands, but ultimately the same results with graphics included.

2.4 Binned Likelihood Analysis

We now move onto a binned likelihood analysis. What is primarily different for this approach which is utilized for longer time bins, is the addition and removal of a few shell commands from Fermitools, the automatic source model creation script, and of course, the binned data inputted into `gtlike`.

For this section any code or results presented will be from a separate analysis done on Markarian 421, a bright blazar gamma-ray source. For reference, here is a wide counts map of the region encompassing the source:

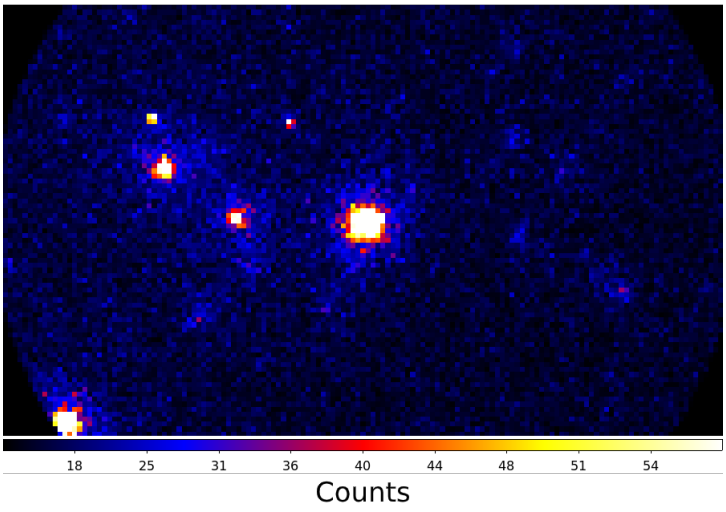

Figure 5: Counts map of Markarian 421, with energy range from 100MeV to 500GeV, using the 'z-scale' ds9 algorithm, and a square scaling.

After selecting the proper data cuts and creating the counts map, we must now create a three dimensional binned counts map, with the third dimension being energy, known as a counts cube. This is due to the fact the the data inputted is binned in three dimensions and thus a regular counts map won't suffice. [12]

What must be specified in the counts cube is the size of the square projected that must be inscribed the ROI, which is done by multiplying the search radius by $\sqrt{2}$. Next we have a newly specified parameter in the calculation of the counts cube, which is the number of logarithmically uniform energy bins, and of which we will be using the default value of 37. This is accomplished using the previously utilised function `gtbin`, although now specifying the algorithm to a counts cube and not counts map, which is as follows:

```
time gtbin evfile=Mark_gti.fits scfile=NONE outfile=Mark_ccube.fits
    algorithm=CCUBE nxpix=100 nypix=100 binsz=0.2 coordsys=CEL
```

```
    xref=166.114 yref=38.2088 axisrot=0 proj=AIT ebinalg=LOG emin=100
    emax=500000 enumbins=37
```

Next, we will develop our source model using the make4FGLxml.py script provided on the FSSC website. As previously mentioned, it detects all 4FGL sources in the ROI including the diffuse and isotropic galactic models, meaning any extra sources not contained in the catalog will have to be manually added in the XML file. This python script is run in the terminal through the following command:

```
time python make4FGLxml.py gll_psc_v21.xml Mark_gti.fits -v TRUE -s
    15000.0 -o Mark_input_model.xml -G gll_iem_v07.fits -g gll_iem_v07
    -I iso_P8R3_SOURCE_V2_v1.txt -i iso_P8R3_SOURCE_V2_v1
```

Any files within the script line is just to clarify to the script which background model files and event files are being used. Also notice the $-s$ parameter, which dictates how significant a source must be to have its parameters freed, and 15000 was found to be the best value. Note it is not mentioned what the units are for that parameter in the script's documentation, but its certainly not in $\sqrt{TS}$ units as that would surpass the sensitivity of the instrument.

Once the model is constructed and the live time cube is computed, what's next is to calculate an exposure map in three dimensions, known as an exposure cube, with the third dimension being once again energy. This is done using the `gtexpcube2` command, which is executed analogously to `gtexpmap`, as follows:

```
time gtexpcube2 infile=Mark_ltcube.fits cmap=none
    outfile=Mark_expcube.fits irfs=P8R3_SOURCE_V2 nxpix=300 nypix=300
    binsz=0.2 coordsys=CEL xref=166.114 yref=38.2088 axisrot=0 proj=AIT
    emin=100 emax=500000 enumbins=37
```

The final calculation before running the optimization is a source map. The map itself is a convolution of the source model parameters with the IRFs which is used to stitch together the count map data in space and energy, for the optimization to run more efficiently. [15] This is done with the `gtsrcmaps` function, which inputs all of what has been calculated so far, including the live time cube, counts cube, input source model, and exposure cube. This results in a compact file readily accessible to `gtlike`. The function itself is run as such:

```
time gtsrcmaps expcube=Mark_ltcube.fits cmap=Mark_ccube.fits
    srcmdl=Mark_input_model.xml bexpmap=Mark_expcube.fits
    outfile=Mark_srcmaps.fits irfs=CALDB
```

Running the optimization, if properly converging, will output a counts/MeV ratio as a function of MeV plot to demonstrate how well each source, including background models, converged to the model of the source of interest, in this case Markarian 421. The following figure demonstrates this:

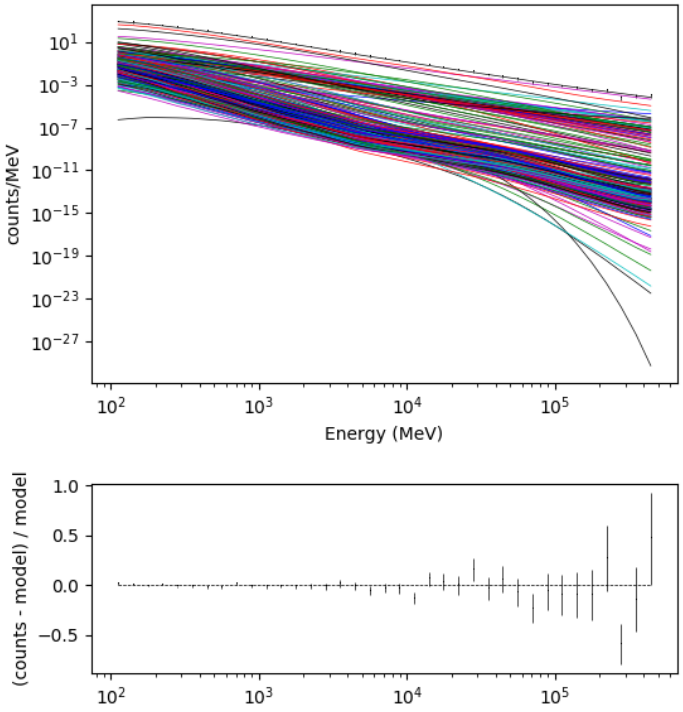

Figure 6: counts/MeV as a function of MeV for sources in ROI, targeted on Markarian 421.

The very top black line represents the model while the five lines below it represent the targeted source, the background model sources, and the closest sources to Markarian 421. The clutter under it represent hundreds of sources detected in the ROI that are further away. The residuals also show a successful fit as it doesn't surpass three sigma.

What is also useful in trusting parameters extracted from the fit is directly comparing the data via a counts map to a model map. This is done via the `gtmodel` function, and essentially produces a statistic map based on parameters from the fit. The execution of the command looks something like:

```
time gtmodel srcmaps=Mark_srcmaps.fits srcmdl=Mark_output_model.xml
    outfile=Mark_model_map.fits irfs=CALDB expcube=Mark_ltcube.fits
    bexpmap=Mark_expcube.fits
```

This calculation produces the following comparative result:

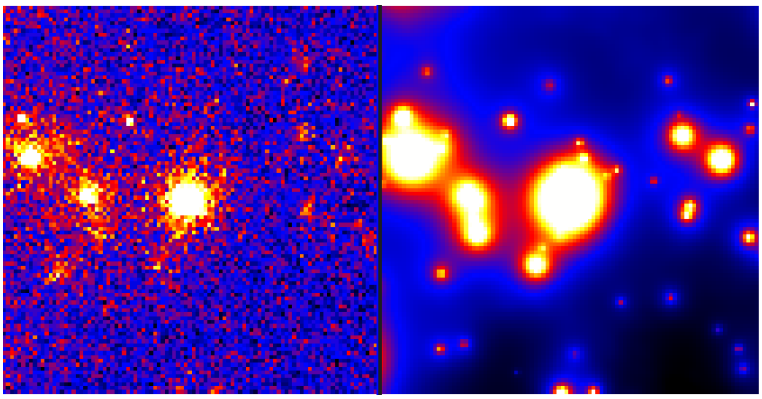

Figure 7: Comparison of counts map on the left and model map on the right, targeted at Markarian 421, using the 'z-scale' ds9 algorithm, and a linear scaling.

As you can see, in the model some of the sources are blown up, especially the doublet sources on the top right of Markarian 421, which are artifacts created by minuscule optimization miscalculations. To look more explicitly at these inconsistencies we can develop a residual map to see how off they are. This is done using HEASOFT's `farith` function, which is run as follows:

```
time farith infil1=Mark_cmap_small.fits infil2=Mark_model_map.fits
    outfil=Mark_residual.fits ops=SUB
```

This essentially takes the difference of the two creates a plot able file via ds9, which looks like the following:

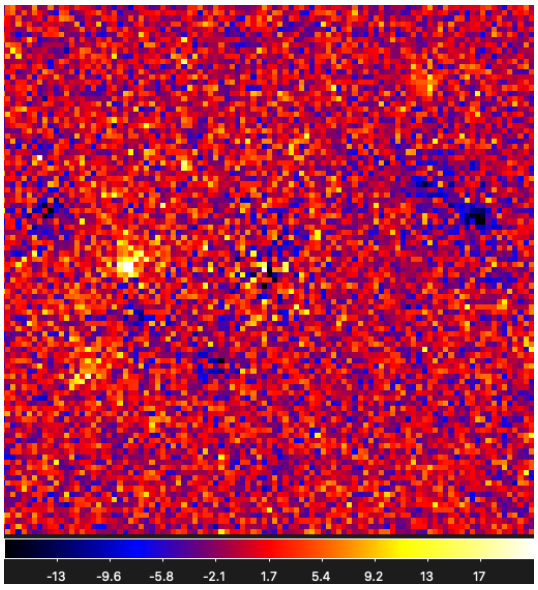

Figure 8: Residual map of Markarian 421, using the 'z-scale' ds9 algorithm, and a linear scaling.

Being that the scale being used is a z-scale, the larger the difference, the darker the output. As you can see, the main areas of discrepancy as discussed before only produce a small amount of dark patches, which confirms the accuracy of the fit.

The main point of a binned likelihood analysis is to extract parameters such as flux and test statistic, but being that they are currently unmotivated numbers they won't be presented. In the final section when the data is directly comparing to a published or currently drafted paper, the numbers will serve to directly compare results. In the next section, we will look at analyses that include extended sources (sources that are very close to the detector).

2.5 Extended Source Analysis

The main difference in handling an extended source as compared to a regular or transient source in a binned likelihood analysis is a two dimensional template that describes the event being analyzed. [10] Furthermore, when selecting data from the query, extended files are selected rather than regular photon and spacecraft files.

After selecting the proper data cuts and creating a preliminary counts map, we must now access the SkyView Query Form to select our template, which following the tutorial will be for the galaxy Centaurus A, that has two radio lobes on the vertical axis. After selecting the type of infrared cosmic microwave background (CMB) template, and the image size in degrees and pixels, we are left with the following template file, and a smoothed counts map to compare:

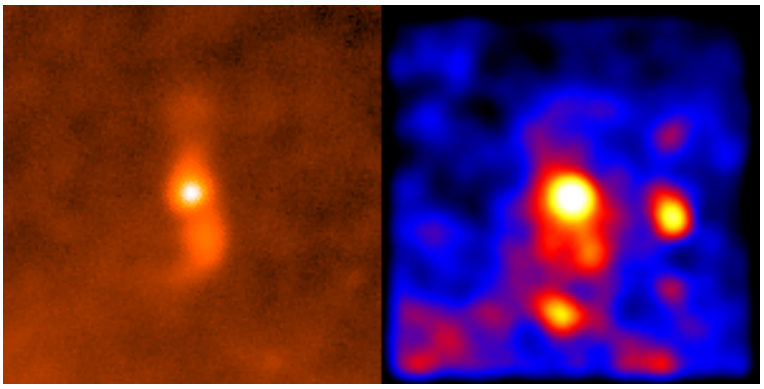

Figure 9: Template map of Centaurus A on left and smoothed counts map on right, using the 'z-scale' ds9 algorithm, and a linear scaling.

Before moving on, the source and lobes of the template file must be isolated, meaning first getting rid of background noise, and then cutting out the lobes. This is done via a list of Python commands, which will be put in the annex at the end of the paper, where the dissection is also visualized.

Now we can go through the same procedure, creating a counts cube, live time cube, source model, exposure cube and source maps. We then have a standard binned likelihood analysis, which is done with Python this time instead of shell commands. Although this willl also be put in the annex, the following snippet is just to give the reader an idea of how a fit is done:

```
#!/usr/bin/env python

#importing necessities
import pyLikelihood
from BinnedAnalysis import *
#defining some variable sets
obs =
    BinnedObs(srcMaps='extTUT_srcmaps.fits',expCube='extTUT_ltcube.fits',
binnedExpMap='extTUT_expcube.fits',irfs='P8R3_SOURCE_V2')
#looser tolerance pass w/ MINUIT
like1 = BinnedAnalysis(obs,'CenA_model.xml',optimizer='MINUIT')
#setting tolerance and performing and saving fit results
like1.tol = 0.1
like1obj = pyLike.Minuit(like1.logLike)
print("logLike of first pass")
print(like1.fit(verbosity=0,covar=True,optObject=like1obj))
like1.logLike.writeXml('CenA_fit1.xml')
```

The full code as mentioned before is at the end of the paper. This results in model maps of the background, the main source, residual maps and diffuse sources, which have been presented in the previous section already. For visualization of a comparison between a full and isolated model map, we have the following figure:

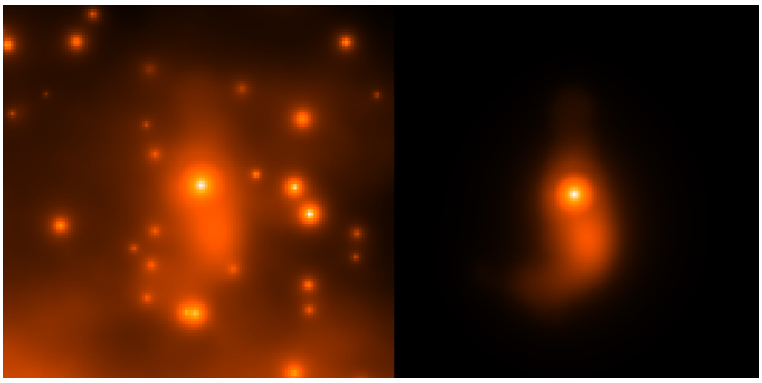

Figure 10: Full and isolated model map of Centaurus A.

Notice the more defined features of Centaurus A's lobes, this accuracy allows the extracted parameters to be more accurate when performing the analysis. With this finished, in the next section we will look at computing upper limits given a source of interest.

2.6 Upper Limit Analysis

One of the more important analyses that can be performed via Fermitools is the calculations of upper limits. This comprises of testing the sensitivity of the instrument, or rather finding the maximum significance an event can have and still be denoted as a source and not just noise by the LAT. This is done using a non-detection, either using a very dim part of the sky or a faint source. Before moving on it should be noted to the reader that computing upper limits via Fermitools is very limited and unclear what functions are, being defined with little to no documentation. This is the reason for the FermiPy section of the paper that deals with computing these upper limits more efficiently and with much more liberty.

The tutorial follows using Python to perform the preliminary binned likelihood analysis, but it won't change much using shell commands instead, so that's what we are going to use. After computing a binned likelihood analysis, that is using two passes (each with an increasing tolerance of detection) we are ready to remove some insignificant sources. This is done by deleting any sources within the ROI that has a test statistic less than 10, this will look something like:

```
sourceDetails = {}
for source in like2.sourceNames():
    sourceDetails[source] = like.Ts(source)
for source,TS in sourceDetails.iteritems():
    print(source, TS)
    if (TS < 10):
        print("Deleting...")
        like2.deleteSource(source)
print(like2.fit(verbosity=0,covar=True,optObject=like2obj))
print(like2obj.getRetCode())
like2.logLike.writeXml('example_ts10.xml')
```

We now run a third pass, meaning a third fit using the new source model and are ready to compute upper limits. We then call the `ul` function for a desired source to extract the value. This is done similar to previous fits and looks like the following:

```
like3 = UnbinnedAnalysis(obs,'example_ts10.xml',optimizer='NewMinuit')
from UpperLimits import UpperLimits
ul = UpperLimits(like3)
ul['targeted_source'].compute(emin=100,emax=10000)
print(ul['targeted_source'].results)
```

This outputs a photon flux upper limit in $ph/cm^2/s$ units, but doesn't tell us what energy bin it is evaluated at, so it is unclear how to convert it to an upper limit on luminosity or integrated flux. We will now move onto FermiPy which solves these issues.

3 FermiPy: Tutorials

The following analysis is done using the latest version of FermiPy 0.19.0. All of the analyses are done using a YAML configuration file to encode the very detailed data cuts, and the rest is ran through the fermipy Conda (Anaconda3) environment running on Python 2.7.14. This is because it hasn't been updated for Python 3, but it works smoothly after overcoming some headaches. The only instance of shell commands will be using Fermitools to compute the livetime cubes as it saves time on computation.

3.1 Configuration Setup

Lets begin by covering the setup of the configuration file. Although there are many more parameters that can be included in the file, adding too many can cause issues in computation, so it is suggested to try to limit it to the most essential components. We will break down the format file in this section and include the full configuration in the annex.

First off, we want as much output information as possible when executing commands and thus will set the verbosity to three in the logging section. We will also tell it which files to input when computing the binned analysis in the data section. This will look like the following:

```
logging :

  verbosity : 3 # how extensive the text output of the commands are

data:

   evfile : example_gti . fits  # event file
   scfile : spacecraft . fits  # spacecraft file
  ltcube : example_ltcube . fits  # livetime cube file
```

Next, we will specify specific cuts in the binning section such as the width of the ROI square, and the coordinate system. We will also specify data cuts in the selection section that is analogous to how is done in Fermitools, such as restricting the time and energy range. This will look something like:

```
binning:

  roiwidth :    10.0 # square length of square embedded in ROI
  npix :        null # pixel length of square
  binsz :       0.1  # spatial bin size in deg
  binsperdec : 8    # number of energy bins per decade
  coordsys :    'GAL' # galactic coordinate system

selection:

  emin :     500 # lower energy bin
  emax :     500000 # upper energy bin
  zmax :     100 # maximum zenith angle cut
  evclass :  128 # event class
  evtype :   3 # event type
  tmin :     239557414 # lower time bin
  tmax :     428903014 # upper time bin
  filter :   null # gtmktime filter
  # radius  : 15 # ROI radius
  ra :       260.05167 # right ascension of target event
  dec :      57.91528 # declination of target event
```

Next we will specify what IRFs to use and which sources to not allow corrections energy dispersion corrections to be applied to. This is demonstrated in the following:

```
gtlike:

  edisp : True # energy dispersion correction
  irfs  : 'P8R2_SOURCE_V6' # IRF
  edisp_disable : ['isodiff','galdiff'] # disabling corrections to
      model sources
```

Finally we have our model parameters. In here we tell the optimizer which galactic diffuse and isotropic models to use, which catalog, and the object of interest. This looks something like:

```
model:

  src_roiwidth : 15.0 # square length of source region

  galdiff  :  'gll_iem_v07.fits' # galactic model
  isodiff  :  'iso_P8R3_SOURCE_V2_v1.txt' # isotropic model

  catalogs :
    - '3FGL' # 3FGL catalog

  sources :
    - { name :'target', ra : 230, dec : 72,
        SpectrumType :'PowerLaw', Index : 2.0,  Prefactor  : { 'value' :
            0.0, 'scale' : 1e-13 } } # target source
```

```
components: null # only for independent subsections of data
```

Thus concludes setting up the configuration file for the upper limits analysis.

3.2 Upper Limit Analysis

Another benefit of using FermiPy is the automation of a binned likelihood analysis with a considerable increase and clarity in verbosity when it comes to outputs and errors. The following analysis will be based on the Draco dSph galaxy tutorial on the fermipy github. Since it is a Jupyter notebook and since there is many lines of code, it won't be annexed or presented in this paper. The reader is therefore encouraged to take a look at it to see how it works being that it is almost analogous to how the upper limit analyses in the comparative results section are computed.

The first part of the analysis includes calling the configuration file and running the setup command, which will compute all the necessary calculations before a binned analysis, such as a live time cube, source map, etc. Once completed, you can view all the objects in the ROI via a list, as well as visualize whats going on with a data counts and model map. This will look like the following figures:

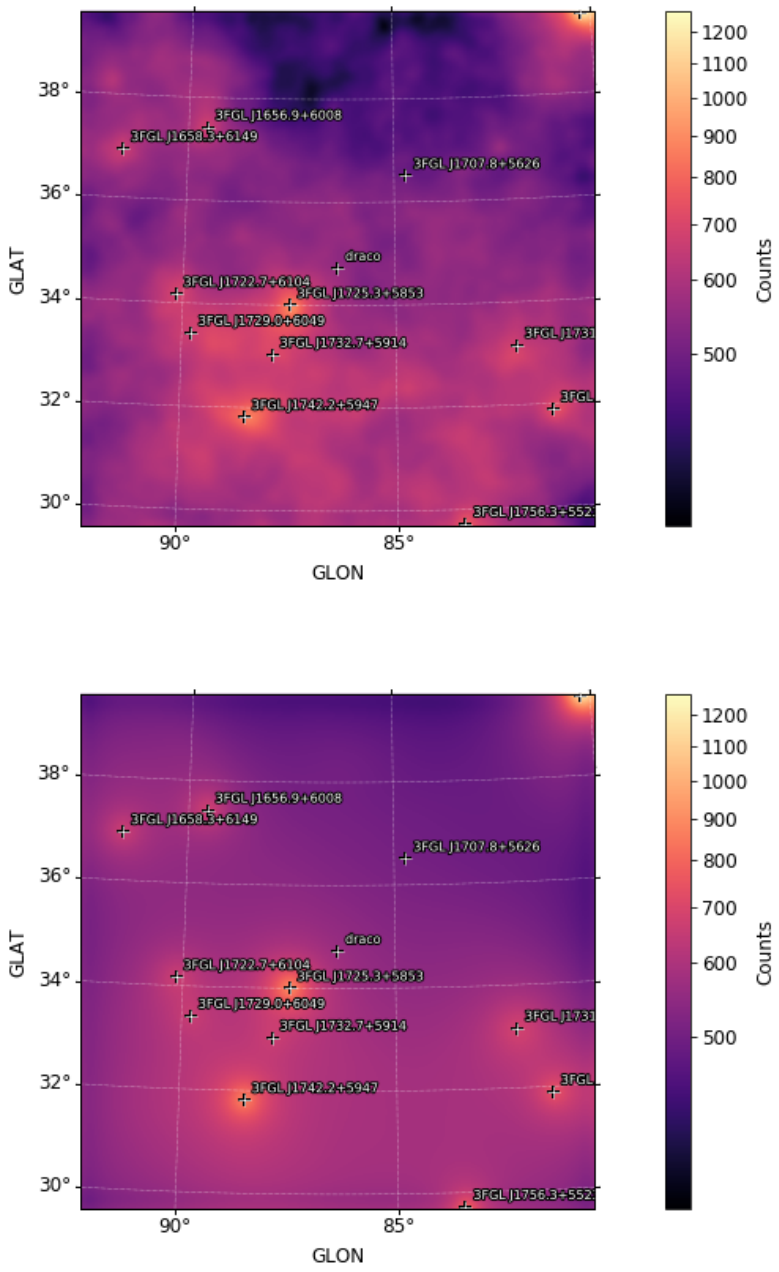

Figure 11: Data counts map on top and model map on bottom of the Draco ROI.

Next, being that we have the model already computed, we can plot the significance and excess (residual) maps, which look like the following:

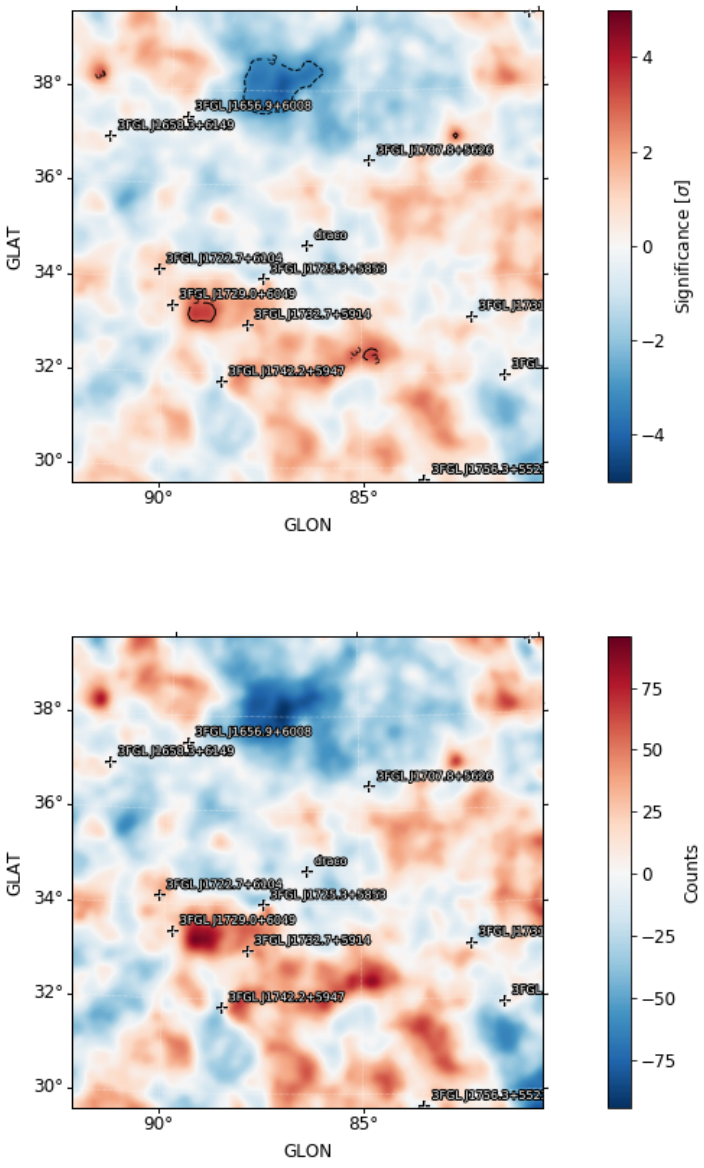

Figure 12: Significance map on top and excess map on bottom of the Draco ROI.

If there any significant sources that are not within the catalog used such as blazars or magnetars, it might be required to add and fix those sources in the source model

to help the second pass fit converge. In this case, the source PS J1705.4+5432 was detected and promptly taken account for, which can be seen in the top right of the following test statistic and number of predicted counts maps:

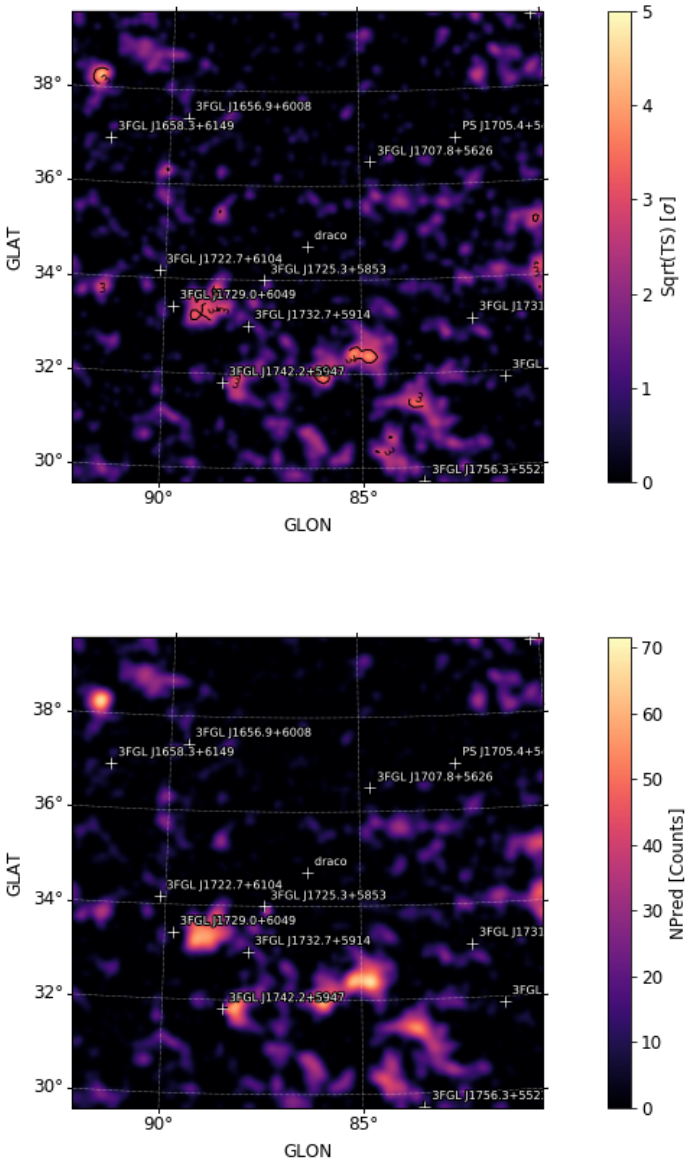

Figure 13: Test statistic map on top and number of predicted counts map on bottom of the Draco ROI.

Finally, what is done is the spectral energy distribution (SED) analysis. This takes the energy range and splits it up into uniform energy bins as specified by the user. A fit is then done with the new data set and parameters such as the test statistic, integrated flux [†] upper limit, or luminosity upper limit of the target event, to name a few. Note, in order to get parameters over the whole energy range instead of split bins, a single bin must be specified with the lower and upper bounds being the lower and upper energy values respectively. Aside from extracting the upper limit values that we have been seeking, it is also useful to plot integral flux values as a function of energy to visualize the upper limit values and conclude what is going on. The following is for 24 uniform logarithmically spaced energy bins for the target source Draco:

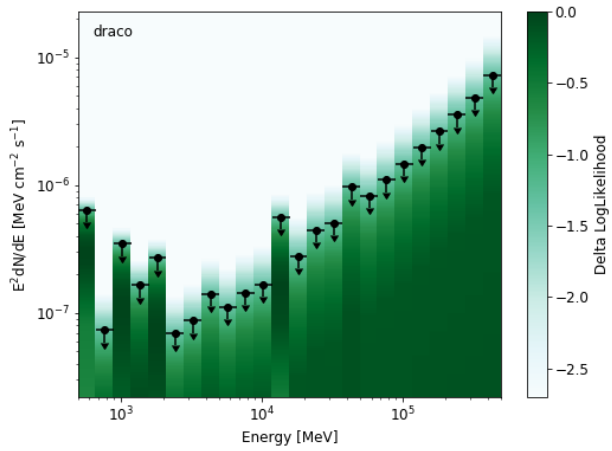

Figure 14: SED plot for Draco dSph galaxy.

Being finished, this leaves us with presenting results replicated from published and in preparation papers.

4 Comparative Results

4.1 Tidal Disruption Event

In this section we will be comparing results on an upper limit analysis based on a paper covering a tidal disruption event AT2019dsg that was coincident with a high energy neutrino IC191001A [18]. For reference, a tidal disruption event (TDE) or flare (TDF) occurs when a star approaches a supermassive black hole close enough that the tidal force pulls it apart as it undergoes spaghettification. The matter that isn't collected in the accretion disk gets emitted as a flare of electromagnetic radiation. [9] Some of this radiation and particles includes a small charge-less particle with

[†]The integrated flux upper limit, which will be the main source of interest in the comparative results section, takes the following form: $\phi_E = \int_{E_{min}}^{E_{max}} dE \, \frac{dN}{dE} E$. [5]

almost negligible mass known as a neutrino. Detected by the IceCube (ICE) Neutrino Observatory, the TDE is able to be located due to the state of the particle, i.e. its energy and velocity. The event was found to be a redshift [†] z of 0.0512, which may be converted to standard units of measurement, granted the assumed Hubble and cosmological constants are specified.

The upper limit analysis was performed on three time ranges in 2019 which are G1, G2, and G3, which have respective time ranges of April 4 to August 12, August 12 to November 20, and April 4 to January 31. The energy range is from 100 MeV to 800 GeV, with a $15° \times 15°$ ROI, maximum zenith angle cut of 90°, 0.1° bin scaling, 10 logarithmically-spaced bins per energy decade, and P8R3 SOURCE IRFs. The galactic diffuse and isotropic models are the most recent editions from the FSSC website, and it is evaluated at a 95 % confidence level. Finally, the assumed power law spectrum index Γ [‡] is 2.

The following table compares upper limits for energy fluxes over the entire energy range from the paper and what we computed [¶]:

Interval	TDE UL (erg cm^{-2} s^{-1})	Our UL (erg cm^{-2} s^{-1})
G1	2.6e-12	2.253e-12
G2	1.2e-11	1.094e-11
G3	2.0e-12	1.777e-12

Table 1: Energy flux upper limits integrated over the energy range 100 MeV to 800 GeV.

As can be seen, the results agree within a certain range. These discrepancies are possibly due to further parameters in the configuration file which haven't been specified in the paper and thus have not been accounted for in the external analysis, which has proven to drastically change results in the past.

We move onto the final section of the paper aside from the annex which is an upper limit analysis performed on two superluminous supernovae events.

4.2 Superluminous Supernovae

Finally we have an upper limit analysis computed based on an in preparation paper by Deivid Ribeiro from Columbia University on superluminous supernovae [§], that has been shared in private communications. The objective of the paper is to reconcile the discrepancy between the observed and predicted optical spectrum. Being that the observed curve is stepper, there is some energy missing, and up until now energy ranges below gamma-rays have been checked without finding anything and so they are expecting to find the missing energy within in the gamma-ray range. This section won't focus on the main goals of the paper, and instead fixates on the reproduction of

[†] The amount a wave's wavelength has increased. It is proportional to the ratio of the observed and emitted wavelength, subtracted by 1. In other words: $z = \frac{\lambda_o}{\lambda_e} - 1$.

[‡] $dN/dE \propto E^{-\Gamma}$ [18]

[¶] 1 erg = 624.151 GeV

[§] Brighter class of supernova that are produced by external interactions such as a magnetar interacting with the ejecta of a supernova which results in it heating up the ejecta and producing light.

the energy flux upper limits performed on the two primary superluminous supernovae: SN2015bn and SN2017egm.

The analysis was performed using a $15^\circ \times 15^\circ$ square [†] ROI, with an energy range of 612 MeV to 500 GeV. A maximum zenith angle cut of 90° was selected, with a 95 % confidence interval, an assumed power law spectral index of 2, bin scaling of 0.2°, 2.5 bins per decade, NEWMINUIT optimizer, and finally the latest galactic diffuse and isotropic models. SN2015bn was observed May 15, 2015 to May 15, 2018, while SN2017egm was observed May 23, 2017 to November 23, 2019. During both time intervals the upper limit analysis on the luminosity was performed over the whole time range, and 6 month time bins. The following tables compare our results to the results of the paper:

Start Time MJD	End Time MJD	Paper Luminosity erg s^{-1}	Our Luminosity erg s^{-1}
57192.0	57374.7	1.39e+44	1.207e+44
57374.7	57557.3	1.82e+44	1.476e+44
57557.3	57740.0	4.62e+44	3.168e+44
57740.0	57922.7	2.31e+44	1.814e+44
57922.7	58105.3	2.42e+44	1.686e+44
58105.3	58288.0	3.39e+44	5.696e+44
57192.0	58288.0	1.32e+44	1.301e+44

Table 2: *Fermi*-LAT Upper limits for SN2015bn for three years of observation.

Start Time MJD	End Time MJD	Paper Luminosity erg s^{-1}	Our Luminosity erg s^{-1}
57896.0	58168.7	1.80e+43	1.669e+43
58168.7	58351.3	9.61e+42	8.6489+42
58351.3	58534.0	1.00e+43	1.002e+43
58534.0	58716.7	2.87e+42	3.366e+42
58716.7	58808.5	4.29e+42	4.041e+42
57896.0	58808.5	4.95e+42	4.300e+42

Table 3: *Fermi*-LAT Upper limits for SN2017egm for two and a half years of observation.

As can be deduced from the tables, the results agree within a certain range, which could be due to minimizers using randomized noise. Thus concludes this brief introduction to Fermi LAT data analysis.

[†]Note this denotes the side length of the square embedded in the ROI, using simple identities reveals the search radius to be approximately 11.1 square degrees.

5 Annex

5.1 Extended Analysis Code

5.1.1 Preparation of Template Data

To isolate the central source, we will perform the following manipulations via Python, imported from astropy:

```
#!/usr/bin/env python3

#importing necessities
import astropy.io.fits as pyfits
import numpy as np
#opening WMAP image + info
wmap_image=pyfits.open('./skv.fits')
print(wmap_image[0].header)
#background suppression: set any pixels less that 0.5mK to 0K
wmap_image[0].data[wmap_image[0].data < 0.5] = 0.0
wmap_image.writeto('CenA_wmap_k_above5.fits')
#creating meshgrid for 14 deg image w/ pixel size 0.1
x = np.arange(-7,7,0.1)
y = np.arange(-7,7,0.1)
xx, yy = np.meshgrid(x, y, sparse=True)
dist = np.sqrt(xx**2 + yy**2)
#set all pixels in WMAP with distance greater than 5 to 0
wmap_image[0].data[dist > 5] = 0.0
wmap_image.writeto('CenA_wmap_k_nobkgrnd.fits')
#subtracting out core of fits image (Cen A)
wmap_image[0].data[dist < 1.0] = 0.0
wmap_image.writeto('CenA_wmap_k_nocenter.fits')
#making purely north map with a normalized flux of 1
wmap_image = pyfits.open('CenA_wmap_k_nocenter.fits')
wmap_image[0].data[0:69,0:140] = 0 #0:69 is the y-data and 0:140 is the
    x-data
norm = np.sum(wmap_image[0].data) * (np.pi/180)**2 * (0.1**2) #norming
    flux
wmap_image[0].data = wmap_image[0].data / norm
wmap_image.writeto('CenA_wmap_k_nocenter_N.fits')
wmap_image.close()
#making purely south map with a normalized flux of 1
wmap_image = pyfits.open('CenA_wmap_k_nocenter.fits')
wmap_image[0].data[70:140,0:140] = 0 #0:69 is the y-data and 0:140 is
    the x-data
norm = np.sum(wmap_image[0].data) * (np.pi/180)**2 * (0.1**2) #norming
    flux
wmap_image[0].data = wmap_image[0].data / norm
wmap_image.writeto('CenA_wmap_k_nocenter_S.fits')
wmap_image.close()
```

The dissection of the template file is visualized via the following figure:

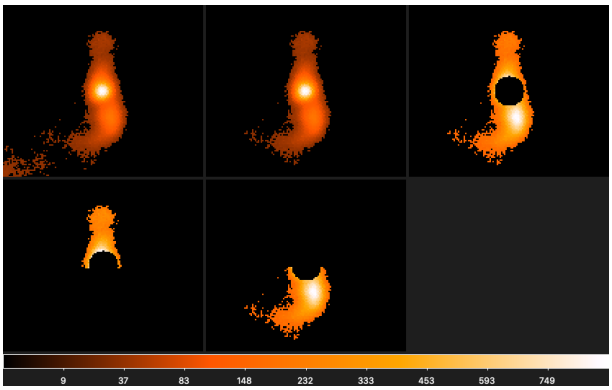

Figure 15: Dissection of template file via a list of Python commands, imported from astropy.

5.1.2 Computation of Binned Likelihood Analysis

Running through the full optimization is as following through these Python commands:

```
#!/usr/bin/env python

#importing necessities
import pyLikelihood
from BinnedAnalysis import *
#defining some variable sets
obs =
    BinnedObs(srcMaps='extTUT_srcmaps.fits',expCube='extTUT_ltcube.fits',
binnedExpMap='extTUT_expcube.fits',irfs='P8R3_SOURCE_V2')
#looser tolerance pass w/ MINUIT
like1 = BinnedAnalysis(obs,'CenA_model.xml',optimizer='MINUIT')
#setting tolerance and performing and saving fit results
like1.tol = 0.1
like1obj = pyLike.Minuit(like1.logLike)
print("logLike of first pass")
print(like1.fit(verbosity=0,covar=True,optObject=like1obj))
like1.logLike.writeXml('CenA_fit1.xml')
#checking convergence (check numbers on ext side, should be 3)
print("Convergence of fit")
print(like1obj.getQuality())
#performing second, more strict fit with lower tolerance
# like1 = BinnedAnalysis(obs,'CenA_model.xml',optimizer='MINUIT')
#tighter (smaller) tolerance pass w/ NewMINUIT
like2 = BinnedAnalysis(obs,'CenA_fit1.xml',optimizer='NEWMINUIT')
like2.tol = 1e-8
like2obj = pyLike.NewMinuit(like2.logLike)
print("logLike of second pass")
print(like2.fit(verbosity=0,covar=True,optObject=like2obj))
```

```
#checking if newMinuit converged (should be 0)
print("Convergence of fit")
print(like2obj.getRetCode())
#saving fitted model
like2.logLike.writeXml('CenA_fit2.xml')

#printing all results
print(like2.model)
#checking values
print("TS of fit 2")
print(like2.Ts('Cen A'))
print("Model info of fit 2")
print(like2.model['Cen A'])
print("Flux")
print(like2.flux('Cen A',emin=100,emax=100000))
print("Flux error")
print(like2.fluxError('Cen A',emin=100,emax=100000))
#checking and printing more values
print("TS of North lobe")
print(like2.Ts('CenA_NorthLobe'))
print("North lobe model info")
print(like2.model['CenA_NorthLobe'])
print("Flux of North lobe")
print(like2.flux('CenA_NorthLobe',emin=100,emax=100000))
print("Flux error of North lobe")
print(like2.fluxError('CenA_NorthLobe',emin=100,emax=100000))
print("TS of South lobe")
print(like2.Ts('CenA_SouthLobe'))
print("South lobe model info")
print(like2.model['CenA_SouthLobe'])
print("Flux of South lobe")
print(like2.flux('CenA_SouthLobe',emin=100,emax=100000))
print("Fluex error of south lobe")
print(like2.fluxError('CenA_SouthLobe',emin=100,emax=100000))
```

5.2 Configuration File

Here we present a template configuration file in full:

```
logging :

  verbosity : 3 # how extensive the text output of the commands are

data:

   evfile : example_gti . fits  # event file
   scfile : spacecraft . fits  # spacecraft file
  ltcube : example_ltcube . fits  # livetime cube file
```

```
binning:

  roiwidth :    10.0 # square length of square embedded in ROI
  npix :        null # pixel length of square
  binsz :       0.1 # spatial bin size in deg
  binsperdec : 8    # number of energy bins per decade
  coordsys :    'GAL' # galactic coordinate system

selection:

  emin :    500 # lower energy bin
  emax :    500000 # upper energy bin
  zmax :    100 # maximum zenith angle cut
  evclass : 128 # event class
  evtype :  3 # event type
  tmin :    239557414 # lower time bin
  tmax :    428903014 # upper time bin
  filter :  null # gtmktime filter
  # radius  : 15 # ROI radius
  ra :      260.05167 # right ascension of target event
  dec :     57.91528 # declination of target event

gtlike:

  edisp : True # energy dispersion correction
  irfs  : 'P8R2_SOURCE_V6' # IRF
  edisp_disable : ['isodiff','galdiff'] # disabling corrections to
      model sources

model:

  src_roiwidth : 15.0 # square length of source region

  galdiff  :  'gll_iem_v07.fits' # galactic model
  isodiff  :  'iso_P8R3_SOURCE_V2_v1.txt' # isotropic model

  catalogs :
    - '3FGL' # 3FGL catalog

  sources :
    - { name :'target', ra : 230, dec : 72,
        SpectrumType :'PowerLaw', Index : 2.0,  Prefactor  : { 'value' :
            0.0, 'scale' : 1e-13 } } # target source

components: null # only for independent subsections of data
```

Note that depending on the analysis, modifying the configuration file is suggested, and this should only be seen as a suggested template.

References

[1] Hadiseh Alaeian. An introduction to cherenkov radiation. http://large.stanford.edu/courses/2014/ph241/alaeian2/, 2014.

[2] David Band and Chuck Patterson. Lat irfs (instrument response functions). https://www.slac.stanford.edu/exp/glast/wb/prod/pages/sciTools_overview/overview_IRF.htm, 2009.

[3] CEN Bordeaux-Gradignan Benoît Lott. Fermi-lat likelihood analysis. https://fermi.gsfc.nasa.gov/science/mtgs/workshops/da2010_india/Benoit_ST_Bangalore_0210.pdf, 2010.

[4] VERITAS Collaboration. Very energetic radiation imaging telescope array system. https://veritas.sao.arizona.edu/about-veritas-mainmenu-81, 2018.

[5] Columbia University Deivid Ribeiro. Fluxes: A linguistic description, 2020.

[6] M. Ackermann et al. The fermi large area telescope on orbit: Event classification, instrument response functions, and calibration. *The Astrophysical Journal Supplement Series*, 203(1):4, October 2012.

[7] J V Jelley. Cerenkov radiation and its applications. *British Journal of Applied Physics*, 6(7):227–232, 1955.

[8] Massachusetts Institute of Technology John Houck, Center for Space Research. An introduction to exposure maps. https://space.mit.edu/ASC/docs/expmap_intro.pdf.

[9] NASA Lee Mohon. Tidal disruption. https://www.nasa.gov/mission_pages/chandra/tidal-disruption.html, 2017.

[10] NASA N. Mirabal. Extended source analysis. https://fermi.gsfc.nasa.gov/ssc/data/analysis/scitools/extended/extended.html, 2018.

[11] NASA N. Mirabal. Source model definitions for gtlike. https://fermi.gsfc.nasa.gov/ssc/data/analysis/scitools/source_models.html, 2018.

[12] NASA N. Mirabal. Binned likelihood tutorial. https://fermi.gsfc.nasa.gov/ssc/data/analysis/scitools/binned_likelihood_tutorial.html, 2019.

[13] NASA N. Mirabal. Unbinned likelihood tutorial. https://fermi.gsfc.nasa.gov/ssc/data/analysis/scitools/likelihood_tutorial.html, 2019.

[14] NASA. gtltcube. https://raw.githubusercontent.com/fermi-lat/fermitools-fhelp/master/gtltcube.txt.

[15] NASA. gtsrcmaps. https://raw.githubusercontent.com/fermi-lat/fermitools-fhelp/master/gtsrcmaps.txt.

[16] NASA. Gamma rays. https://web.archive.org/web/20120502232209/http://missionscience.nasa.gov/ems/12_gammarays.html, 2010.

[17] NASA. Fermi-lat likelihood analysis. https://glast.sites.stanford.edu/, 2012.

[18] et al. Robert Stein, Sjoert van Velzen. A high-energy neutrino coincident with a tidal disruption event, 2020.

[19] Stanford. Fgst: Fermi gamma-ray space telescope. https://fgst.slac.stanford.edu/, 2010.

[20] Swinburne University Study Astronomy Online. Equatorial coordinate system. https://astronomy.swin.edu.au/cosmos/E/Equatorial+Coordinate+System.

[21] B A Van Tiggelen, A Tip, and A Lagendijk. Dwell times for light and electrons. *Journal of Physics A: Mathematical and General*, 26(7):1731–1748, 1993.

[22] Wikipedia. Cherenkov radiation. https://en.wikipedia.org/wiki/Cherenkov_radiation, 2020.