Exploring and Understanding the LAT Instrument Response Functions (IRFs)

Tyrel Johnson
(George Mason University/
@ US Naval Research Lab.)

Tutorial Goals

- Better understand the LAT IRFs
  - How they are derived
  - What they mean

- Know how to plot different IRF quantities
  - From IRFs FITS files directly
  - Using *pyIrfLoader* python module

What you need

- Software
  - *Fermi* STs
  - fv – ftools FITS viewer
- Custom python scripts (linked from schedule)
  - *customIRFplotter.py*
  - *plotIRFs.py*
- Pass7 performance paper (on USB drives)
  - Ackermann et al. 2012, ApJS, 203, 4 (arXiv:1206.1896)
  - Much of the info. in this talk gleaned from that paper
  - The LAT public performance page is also useful

http://www.slac.stanford.edu/exp/glast/groups/canda/lat_Performance.htm

The LAT Response

Measured Energy & Direction

Effective Area

Energy Dispersion

$$R(E', \hat{v}'; E, \hat{v}) = A_{eff}(E, \hat{v}) P(\hat{v}'; E, \hat{v}) D(E'; E, \hat{v})$$

Point-spread Function

True Energy & Direction

Expected Count Rate

Source Flux

$$\frac{dM(E', \hat{v}')}{dt} = \int \int R(E', \hat{v}'; E, \hat{v}) F(E, \hat{v}) d\hat{v} dE$$

Instrument Response

Likelihood fitting uses lots of information optimally.
This is a double-edged sword. Issues with any of our IRFs can affect fit and can be difficult to disentangle.

Slide shamelessly ripped off from E. Charles, FSS2013.

How Do We Derive IRFs?

- Average Response
  - Knowing event-by-event response is tough
  - Simulate a lot of γ-rays, apply cuts, calculate average response
    - dN/dE µ 1/E
    - 2e8 γ-rays, $\log_{10}(E/1\ \mathrm{MeV}) \in [1.25, 5.75]$, all-sky

- How to bin?
  - LAT gives us a lot of information...
  - Bin in "most-relevant" quantities
    - conversion layer, E, θ, ϕ
  - Much more goes into event classification

The Effective Area (A_{eff}) (I)

- Effective collecting area of LAT:
  - Depends on geometric cross section
  - Conversion probability and efficiency
  - Instrument livetime fraction

$$A_{\rm eff}(E_i,\theta_j,\phi_k) = (6\,{\rm m}^2)\left(\frac{n_{i,j}}{N_{\rm gen}}\right)\left(\frac{2\pi}{\Delta\Omega_j}\right) \times \left(\frac{\log_{10}E_{\rm max}-\log_{10}E_{\rm min}}{\log_{10}E_{{\rm max},i}-\log_{10}E_{{\rm min},i}}\right) \times R(E_i,\theta_j,\phi_k), \qquad (11)$$

$$A_{\rm eff}(E,F_l) = A_{\rm eff}(E)\cdot(c_0(E)F_l + c_1(E)) \qquad (14)$$

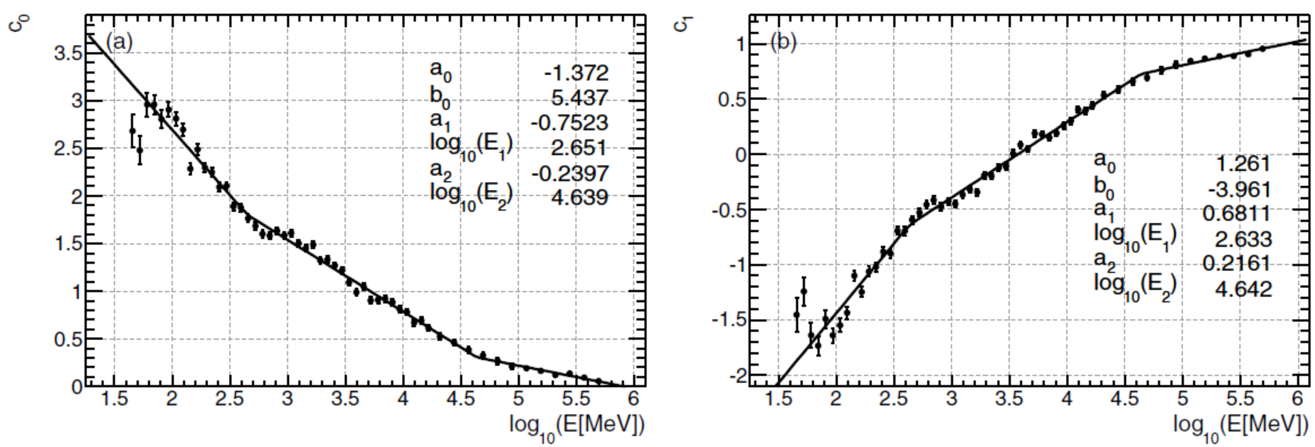

Figure 35.

The Effective Area (A_{eff}) (II)

- Generally, sample entire ϕ phase space well
  - True over long time scales
  - Variations as much as ~10% on shorter time scales
    - few orbits
    - ToOs and ARRs

$$\xi = \frac{4}{\pi}\left|\left(\phi \bmod \frac{\pi}{2}\right) - \frac{\pi}{4}\right| \qquad f(\xi) = 1 + q_0 \xi^{q_1}$$

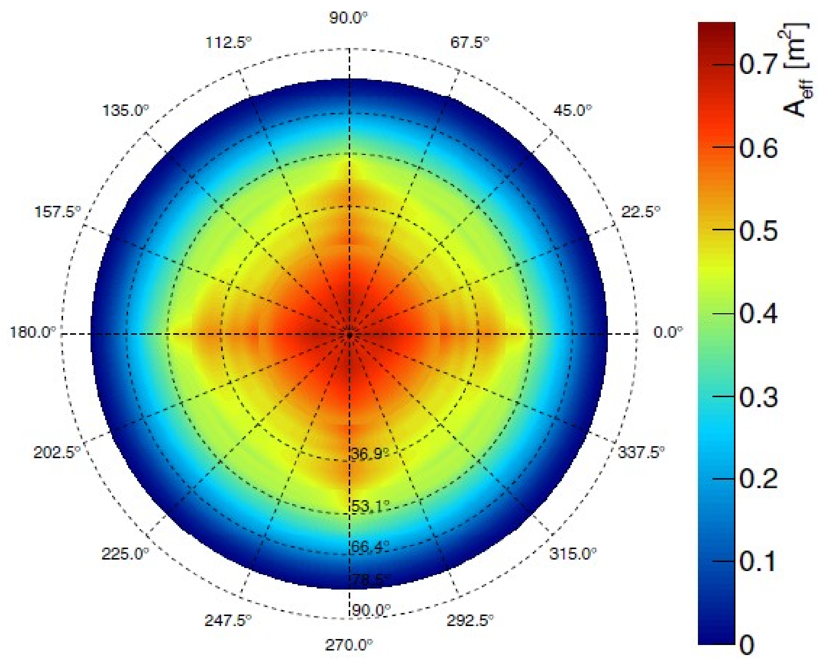

Figure 36. Total effective area at 10 GeV as a function of the incidence angle θ and the azimuthal angle ϕ for the P7SOURCE event class. The plot is shown in a

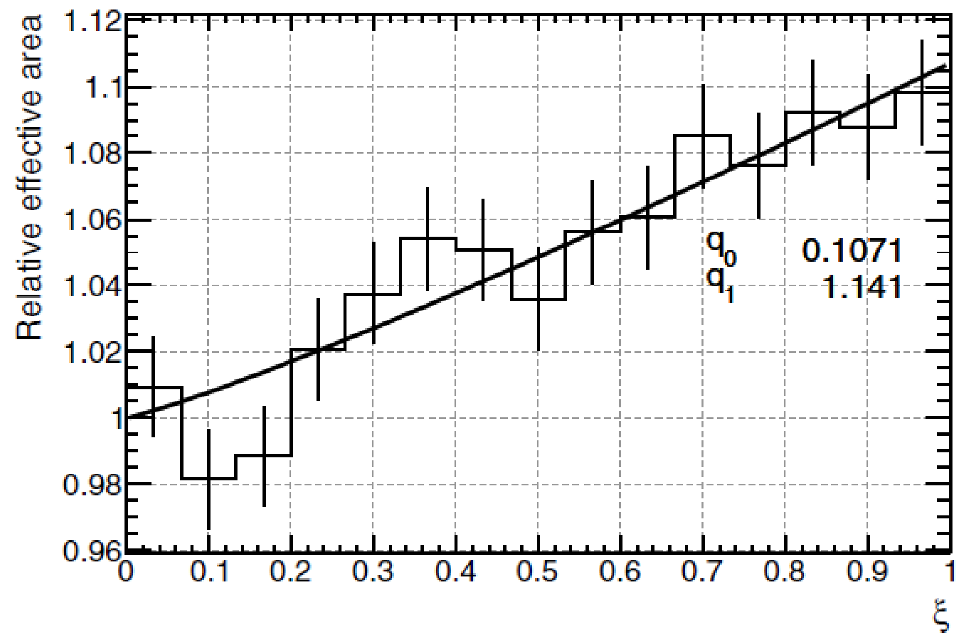

Figure 37. Example of A_{eff} azimuthal dependence fit. The plot refers to the bin centered at 7.5 GeV and 30° for the P7SOURCE class, front section—a similar

Acceptance

- Acceptance is A_{eff} integrated over solid angle
  - Units of m^2 sr

$$\mathcal{A}(E) = \int A_{\rm eff}(E,\theta,\phi)\,d\Omega = \int_0^{\frac{\pi}{2}}\int_0^{2\pi} A_{\rm eff}(E,\theta,\phi)\,\sin\theta\,d\theta\,d\phi, \qquad (12)$$

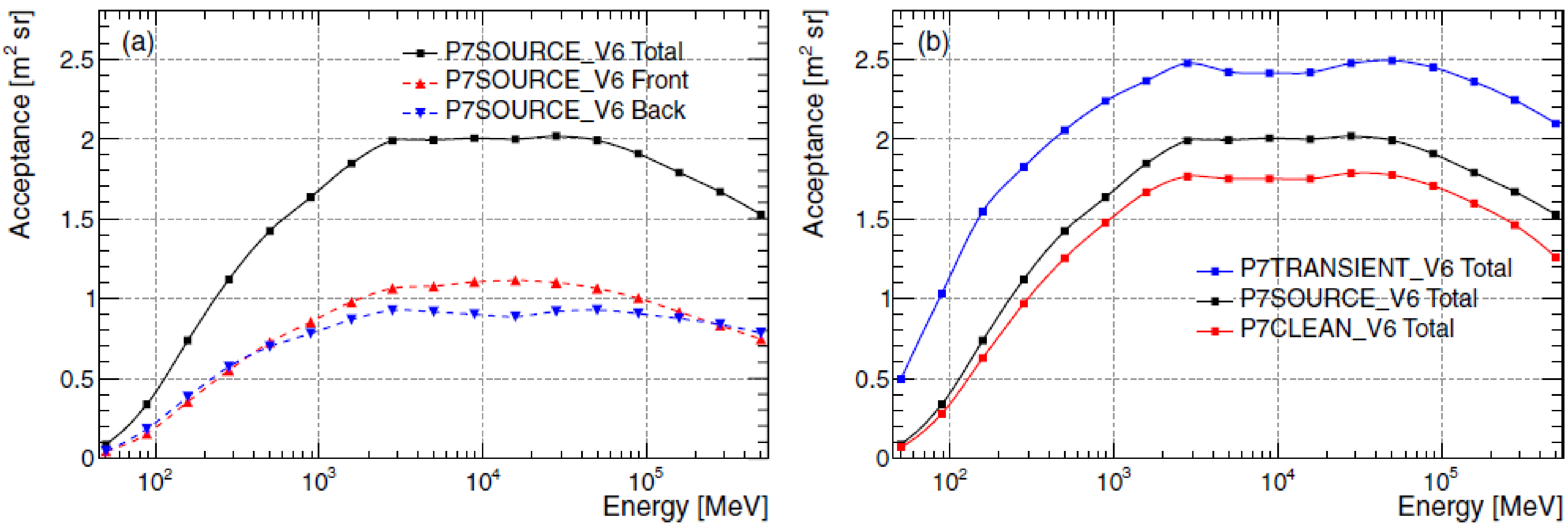

Figure 32. Acceptance as a function of energy for the P7SOURCE class (a) and for the other standard γ-ray classes (b).

Using the A_{eff} FITS file

- Where are they?
  - In CALDB…
    - $CALDB/data/glast/lat/bcf/ea

- How to open?
  - Favorite FITS viewer (e.g., fv)
  - With pyfits

Plotting A_{eff}

- What if you want to build your own plots
  - Check effect of livetime for different energies
    - Different from performance page
  - Curiosity
  - ...
- Things to remember
  - These are average responses
    - average when combining bins or quantities
  - Unless you're adding front and back A_{eff}

The Point-Spread Function (PSF)

- Probability density to reconstruct an event with angular deviation δν from the true direction:
  - Depends on conversion layer, energy, θ, and ϕ
    - Ignore ϕ
  - Simulations binned in energy and θ
  - Deviations in data from simulated PSF
    - Stack bright puslars and AGN -> in-flight PSF
  - Combining event types requires joint cumulative distribution function
    - FRONT+BACK averaging "ok" for 68% containment
    - 95% containment closer to BACK value

Form of the PSF

$$P(x) = f_{\rm core} K(x, \sigma_{\rm core}, \gamma_{\rm core}) + (1 - f_{\rm core}) K(x, \sigma_{\rm tail}, \gamma_{\rm tail}). \quad (38)$$

$$K(x, \sigma, \gamma) = \frac{1}{2\pi\sigma^2}\left(1 - \frac{1}{\gamma}\right) \cdot \left[1 + \frac{1}{2\gamma} \cdot \frac{x^2}{\sigma^2}\right]^{-\gamma}, \quad (36)$$

$$S_P(E) = \sqrt{\left[c_0 \cdot \left(\frac{E}{100\,\mathrm{MeV}}\right)^{-\beta}\right]^2 + c_1^2}. \quad (34) \qquad x = \frac{\delta v}{S_P(E)}. \quad (35)$$

$$\int_0^\infty K(x, \sigma, \gamma)\, \underline{2\pi x\, dx} = 1; \quad (37)$$

PSF cumulative distribution function →

$$2\pi \int_0^x P(x')\, x'\, dx' = f_{core} * \left(1 - \left(1 + \frac{x'^2}{2\gamma_{core}\sigma_{core}^2}\right)^{1-\gamma_{core}}\right) + (1 - f_{core}) * \left(1 - \left(1 + \frac{x'^2}{2\gamma_{tail}\sigma_{tail}^2}\right)^{1-\gamma_{tail}}\right)$$

Using the PSF FITS file

- Where are they?
  - In CALDB…
    - $CALDB/data/glast/lat/bcf/psf
- How to open?
  - Favorite FITS viewer (e.g., fv)
  - With pyfits

The σ and γ values are stored in tables of PSF parameters as SCORE, STAIL, GCORE and GTAIL respectively. Because of the arbitrary normalization used in fitting the PSF function, $f_{\rm core}$ must be extracted from the NTAIL table parameter, in conjunction with SCORE and STAIL:

$$f_{\rm core} = \frac{1}{1 + \texttt{NTAIL} \cdot \texttt{STAIL}^2/\texttt{SCORE}^2}. \qquad (39)$$

The Energy Dispersion (E_{disp}) (I)

- Probability density to reconstruct an event with energy deviation (E'-E)/E from the true energy E:
  - Depends on conversion layer, energy, and θ
    - Generally ignored in likelihood fits
    - More important at low energy
    - More important in pass8
  - When finding E_{disp} for a superset of events, can't average
    - Need to manually construct cumulative distribution function
    - Even for FRONT+BACK in same event class

The Energy Dispersion (E_{disp}) (II)

$$x = \frac{(E' - E)}{S_D(E, \theta) E}$$

$$S_D(E,\theta) = c_0(\log_{10} E)^2 + c_1(\cos\theta)^2 + c_2 \log_{10} E + c_3 \cos\theta + c_4 \log_{10} E \cos\theta + c_5. \quad (48)$$

$$D(x) = \begin{cases} N_L R(x, x_0, \sigma_L, \gamma_L) & \text{if } (x - x_0) < -\tilde{x} \\ N_l R(x, x_0, \sigma_l, \gamma_l) & \text{if } (x - x_0) \in [-\tilde{x}, 0] \\ N_r R(x, x_0, \sigma_r, \gamma_r) & \text{if } (x - x_0) \in [0, \tilde{x}] \\ N_R R(x, x_0, \sigma_R, \gamma_R) & \text{if } (x - x_0) > \tilde{x}. \end{cases} \quad (51)$$

$$R(x, x_0, \sigma, \gamma) = N \exp\left(-\frac{1}{2}\left|\frac{x - x_0}{\sigma}\right|^{\gamma}\right)$$

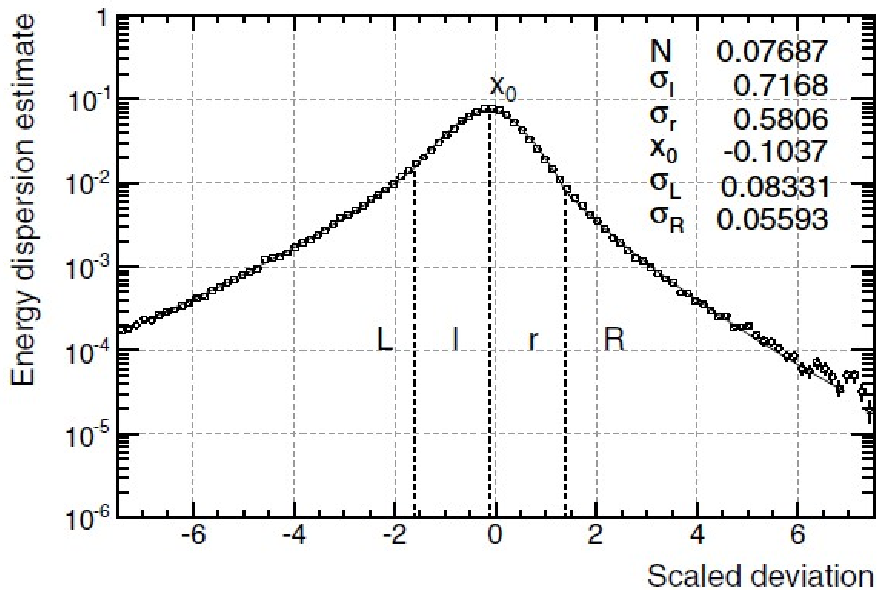

Figure 66. Histogram of the scaled energy deviation, as defined in Equation (49), fitted with the function $D(x)$ in Equation (51). The plot refers to the (E, θ) bin

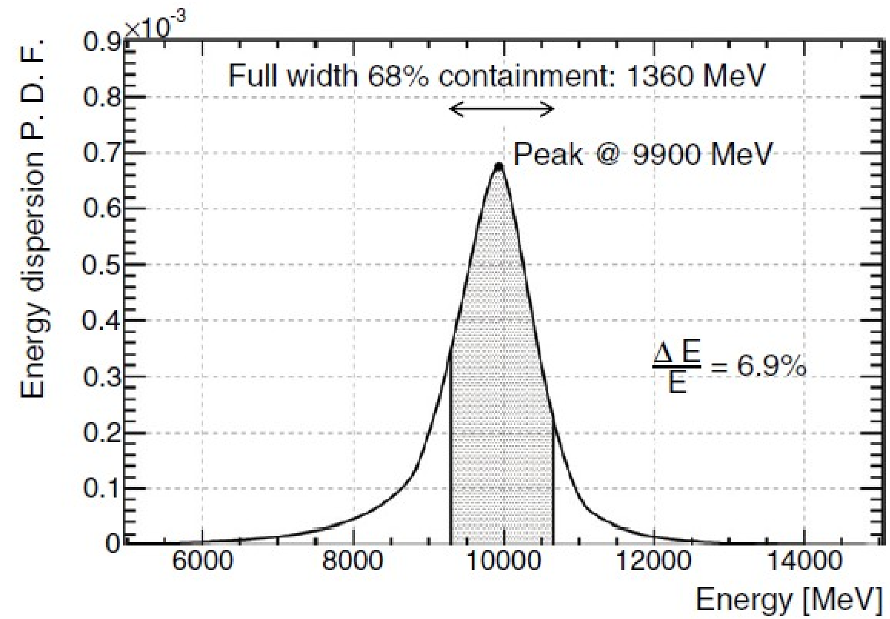

Figure 67. Energy dispersion at 10 GeV for front-converting P7_SOURCE

Using the E_{disp} FITS file

- Where are they?
  - In CALDB…
    - $CALDB/data/glast/lat/bcf/edisp
- How to open?
  - Favorite FITS viewer (e.g., fv)
  - With pyfits

The values of the split point $\tilde{x}$ and of the four exponents γ of the energy dispersion parameterization in Equation (51) are fixed as specified in Table 19. Moreover, the relative normalizations are set by requiring continuity at $x = x_0$ and $|x - x_0| = \tilde{x}$ and therefore the fit is effectively performed with a total of six free parameters, which are stored in the IRF FITS files: the overall normalization $N_r = N_l$ (`NORM`), the centroid position x_0 (`BIAS`), the two core scales σ_r (`RS1`) and σ_l (`LS1`), and the two tail scales σ_R (`RS2`) and σ_L (`LS2`).

Using the *pyIrfLoader* (I)

Basics:

```
>>> import pyIrfLoader
#get available IRFs
>>> pyIrfLoader.Loader_go()
#get your IRFs of choice, note this must be FRONT or BACK version
>>> irfs=pyIrfLoader.IrfsFactory.instance().create('P7REP_SOURCE_V15::FRONT')
```

A_{eff}:

```
>>> ae=irfs.aeff()
#turn phi dependence on or off
>>> ae.setPhiDependence(0)
#check if phi dependence is on or off
>>> ae.getPhiDependence()
#get effective area for specific energy, theta, and phi
#units are cm^2
>>> ae.value(energy,theta,phi)
```

Using the *pyIrfLoader* (II)

PSF:

```
>>> psf=irfs.psf()
#for a given energy and theta, what containment fraction does an angular separation
#s, in degrees, correspond to?
>>> psf.angularIntegral(energy,theta,phi,s)
#can get value of the PSF for a specific set of parameters
#but recall this is the probability density
>>> psf.value(s,energy,theta,phi)
#no phi dependence in PSF, but codes wants it anyway
```

Using the *pyIrfLoader* (III)

E_{disp}:

```
>>> ed=irfs.edisp()
#energy resolution is more complicated than psf containment
#for 68% cont. want the half width of interval from edisp peak
#containing +/-34% from the peak quantile
#first need peak of edisp for a given true energy, theta, and phi (no phi dependence)
>>> peakE=somethingclever
#then what quantile is that
#note, for the edisp.integral function, I don't know why the first entry is always 0
>>> qpeak=edisp.integral(0.,peakE,trueEnergy,theta,phi)
>>> qmin=qpeak-0.34
>>> qmax=qpeak+0.34
#now you need some method to find the energies where edisp.integral=qmin,qmax
>>> energyRes=(emin-emax)/2./trueEnergy
```

NOTE: my slapped-together method of getting energy resolution in *plotIRFs.py* seems to be approximately right, okay for demonstrative purposes, but isn't what is officially used, hope to have documentation soon.