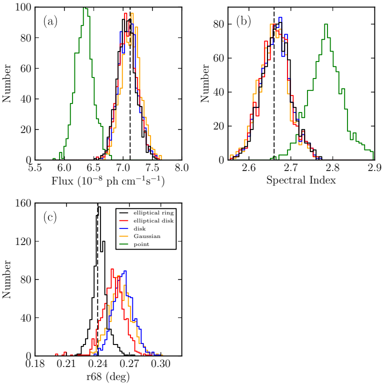
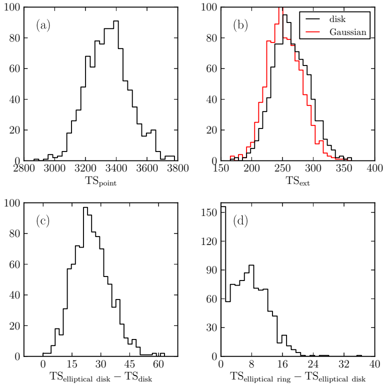
pointlike's MC Simulation Package

Joshua Lande

June 27, 2012

Motivation (Ext. Srs. Search paper)





- Simulation of W44 + Galactic & Isotropic Diffuse $\sim 1,000$ times
- Look at distribution of TS values fitting with different spatial models

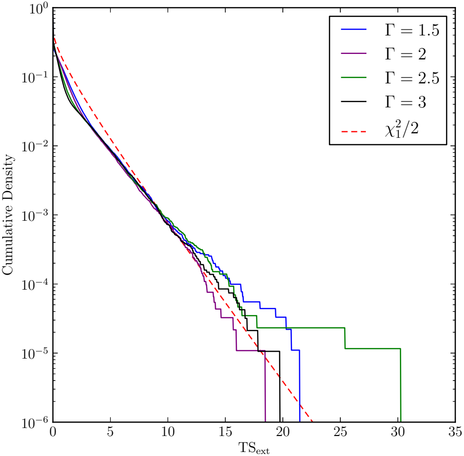
Table 1. Monte Carlo Spectral Parameters

Spectral Index	Flux[(a)] (ph cm^{-2} s^{-1})	$N_{1-100\mathrm{GeV}}$	$\langle\mathrm{TS}\rangle_{1-100\mathrm{GeV}}$	$N_{10-100\mathrm{GeV}}$	$\langle\mathrm{TS}\rangle_{10}$
	Isotropic Background				
1.5	3×10^{-7}	18938	22233	18938	
	10^{-7}	19079	5827	19079	
	3×10^{-8}	19303	1276	19303	
	10^{-8}	19385	303	19381	
	3×10^{-9}	18694	62	12442	
2	10^{-6}	18760	22101	18760	
	3×10^{-7}	18775	4913	18775	
	10^{-7}	18804	1170	18803	
	3×10^{-8}	18836	224	15256	
	10^{-8}	17060	50	⋯	
2.5	3×10^{-6}	18597	19036	18597	
	10^{-6}	18609	4738	18608	
	3×10^{-7}	18613	954	15958	
	10^{-7}	18658	203	⋯	
	3×10^{-8}	14072	41	⋯	
3	10^{-5}	18354	19466	18354	
	3×10^{-6}	18381	4205	15973	
	10^{-6}	18449	966	⋯	
	3×10^{-7}	18517	174	⋯	
	10^{-7}	13714	41	⋯	
	Galactic Diffuse and Isotropic Background[(b)]				
1.5	2.3×10^{-8}	90741	63	⋯	
2	1.2×10^{-7}	92161	60	⋯	
2.5	4.5×10^{-7}	86226	47	⋯	
3	2.0×10^{-6}	94412	61	⋯	

[(a)] Integral 100 MeV to 100 GeV flux.

[(b)] **For the Galactic simulations, the quoted fluxes are the fluxes for sources the Galactic center. The actual fluxes are scaled by Equation 12.**

Note. — A list of the spectral models of the simulated point-like sources which were extension. For each model, the number of statistically independent simulations and the ave of TS is also tabulated. **The top rows are the simulations on top of an isotropic ba and the bottom rows are the simulations on top of the Galactic diffuse and background.**



$\sim 90,000$ simulations/model!

gtobssim Overview

- Input to gtobssim:
  - XML File
  - Ft2 file/source list
  - templates for certain spectral and spatial models (more soon. . .)
- After running gtobssim
  - Remove bad time intervals from simulated data
  - Apply zenith angle cut to simulated data
- *Building the* gtobssim *XML file can be error prone*
- Cutting simulated data can be error prone

pointlike's MC Simulation package

- I developed a wrapper around gtobssim to automate otherwise time consuming, tedious, or error-prone tasks
- Built around pointlike, an alternate maximum likelihood package written in python
  - Uses as input a list of pointlike sources
  - Builds the XML file for gtobssim
  - Converts unsupported models into required templates.
  - Automatically removes bad time intervals + zmax cut
- Code is in pointlike package: uw.like.roi_monte_carlo.py.
- http://www-glast.stanford.edu/cgi-bin/viewcvs/pointlike/python/uw/like/roi_monte_carlo.py

Point Sources

- power law point sources are easy:

```
<source name="source" flux="0.03">
  <spectrum escale="MeV">
    <particle name="gamma">
      <power_law emin="20" emax="1000000" gamma="1.9"/>
    </particle>
    <celestial_dir ra="193.98" dec="-5.82"/>
  </spectrum>
</source>
```

- `gtobssim` supports power law, (dark matter) line, broken powerlaw
  - `http://fermi.gsfc.nasa.gov/ssc/data/analysis/scitools/other_sources.html`
- No LogParabola, ExpCutoff. . .

Point Sources (cont)

- Any spectrum can be simulated using `FileSpectrum`:

```
<source name="FileSpectrum">
    <spectrum escale="MeV" >
        <SpectrumClass name="FileSpectrum" params="flux=0.,
         specFile=$(FERMI_DIR)/spectrum.dat"/>
        <celestial_dir ra="194.04" dec="-5.789"/>
    </spectrum>
</source>
```

- Requires generating a 1D data file of spectral model
- `roi_monte_carlo` uses `pointlike` to automatically build the `FileSpectrum`
- `gtobssim` requires integral of spectral model (done automatically by `roi_monte_carlo`)
- WARNING! `FileSpectrum` objects cannot contain 0 pixels (stripped out by `roi_monte_carlo`)

Diffuse Sources Sources

- MapCube model to simulate diffuse background:

```
<source name="map_cube_source">
  <spectrum escale="MeV">
    <SpectrumClass name="MapCube" params="1.,
      $(FERMI_DIR)/mapcube.fits "/>
    <use_spectrum frame="galaxy"/>
  </spectrum>
</source>
```

- Requires 3D integral of fits file
- Integration automatic by `roi_monte_carlo`

Isotropic Diffuse

- `FileSpectrumMap` for simulation the isotropic diffuse:

```
<source name="isotropic">
   <spectrum escale="GeV" flux="1.">
      <SpectrumClass name="FileSpectrumMap"
      params="flux=17,
        fitsFile=$(FERMI_DIR)/iso_spatial.fits,
        specFile=$(FERMI_DIR)/iso_spectral.dat,
        emin=100,emax=1e5"/>
      <use_spectrum frame="galaxy"/>
   </spectrum>
</source>
```

- Must compute flux for isotropic spectrum
- Must generate allsky spatial `fits` file predicting 1
- Must include energy range from isotropic file
- All done automatically by `roi_monte_carlo`

Extended Sources

```
<source name="gaussian_source">
  <spectrum escale="MeV">
    <SpectrumClass name="GaussianSource"
      params="0.1,2.1,45,30,3,0.5,45,30,2e5"/>
    <use_spectrum frame="galaxy"/>
  </spectrum>
</source>
```

- `gtobssim` only natively supports an elliptical Gaussian spatial model with a power law spectral model.
- WARNING, the ellipse angle is defined west of celestial north)!

Extended Sources (cont)

- Any extended source can be represented by a `FileSpectrumMap`:

```
<source name="filespectrummap_test">
  <spectrum escale="GeV" flux="1.">
    <SpectrumClass name="FileSpectrumMap" params="
      flux=17,
      fitsFile=$(FERMI_DIR)/spatial.fits,
      specFile=$(FERMI_DIR)/spectral.dat,
      emin=100, emax=1e5"/>
    <use_spectrum frame="galaxy"/>
  </spectrum>
</source>
```

- Have to:
  - Build fits template for spatial model
  - Build text file for spectral model
  - Calculate flux of spectral model
- XML built automatically by `roi_monte_carlo` for any of `pointlike`'s extended sources (disk, Gauss, NFW,

Common Gotcha's (Energy Dispersion)

- Energy dispersion means photons with energies outside simulation range can disperse into simulation
  - All spectral models must be simulated for energies well outside simulation range!
- Handled automatically by `roi_monte_carlo`
  - Parameter `roi_pad` (default=2) will pad a given amount to energy to every spectral model:

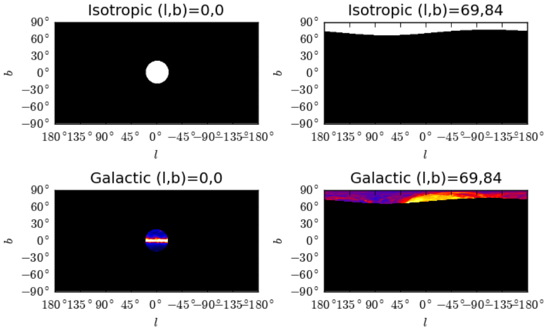
```
actual_emin = simulation_emin/roi_pad
actual_emax = simulation_emax*roi_pad
```

More Common Gotcha's

- All fluxes be in ph $m^{-2}s^{-1}$
- Must remove bad time intervals
  - `gtobssim` takes as input an FT2 file (no info on GTIs)
  - `gtlike` analysis uses `ltcube` sumed over GTIs
  - Can only use `gtmktime` if you know exact cuts applied to data (problematic)
  - No Science Tool for applying GTIs from another file
  - `roi_monte_carlo` uses a combination of `pyfits` and `gtmktime` to apply exact GTIs from an `ft1` or `ltcube` file.
- zenith angle cut must be consistent with `zmax` used when generating `ltcube`
  - Flag in `roi_monte_carlo` to automatically apply `zmax` after simulation.

All sky vs region simulations

- gtobssim will simulate over all sky for allsky MapCube files
- gtobssim's use_ac parameter is applied AFTER the simulation!
- This is very inefficient when simulating a particular region in the sky
- As far as I can tell, non-allsky MapCube fits files will cause strange projection effects
- lonMin and lonMax parameters for spatial models does not work correct.



- My solution for simulation small regions of the sky is to set to 0 pixels far away from ROI
- Done automatically by `roi_monte_carlo`
- Dramatic speedup for simulations of small regions
- Also, cut out energy bins in `MapCube` far away from simulation energy range.

roi_monte_carlo Usage

- First, build a list of pointlike sources
- Most easily, use pointlike's XML parser:

```
from uw.utilities.xml_parsers import parse_sources
ps,ds=parse_sources(xmlfile)
sources=ps+ds
```

- Also build source programmatically with pointlike:

```
from uw.like.pointspec_helpers import PointSource
from uw.like.Models import PowerLaw
skydir = SkyDir(34,-100, SkyDir.EQUATORIAL)
model = PowerLaw(norm=1e-10, index=2)
ps=PointSource(name='ps', model=model, skydir=skydir)
```

Run the Simulation

```
from skymaps import SkyDir
roi_dir = SkyDir(30, 0.5, SkyDir.GALACTIC)

from uw.like.roi_monte_carlo import MonteCarlo
mc = MonteCarlo(
    sources=sources,
    seed=0,
    emin=1e3,
    emax=1e4,
    roi_dir=roi_dir,
    maxROI=10,
    irf='P7SOURCE_V6',
    ft1='ft1.fits',
    ft2='ft2.fits',
    gtifile='ltcube.fits',
    zmax=100,
)
mc.simulate()
```

- (optional) `roi_dir` + `maxROI` for MapCube cutting

Conclusion

- `roi_monte_carlo` avoids otherwise time consuming, tedious, or error-prone tasks
- `roi_monte_carlo` part of `pointlike` and automatically distributed with the Science Tools
- Documented on Confluence: `https://confluence.slac.stanford.edu/x/MIAcBw`
- Python module can be easily built on top of:
  - For example, this package is the basis of MakeSkyModelsFromCatalog (see work by Stephan Zimmer)