

GammaLib - Feature #605

Add log parabola spectral model

11/30/2012 11:47 AM - Knödlseider Jürgen

<b>Status:</b>	Closed	<b>Start date:</b>	11/30/2012
<b>Priority:</b>	Normal	<b>Due date:</b>	
<b>Assigned To:</b>	Mayer Michael	<b>% Done:</b>	100%
<b>Category:</b>		<b>Estimated time:</b>	6.00 hours
<b>Target version:</b>	00-07-00		
<b>Description</b>			
A log parabola spectral model should be added, following the conventions and formula implemented for Fermi-LAT.			
To start implementation, just copy one of the other spectral models (e.g. GModelSpectralPlaw into GModelSpectralLogParabola) and adapt the code.			

History

#1 - 11/30/2012 11:53 AM - Knödlseider Jürgen

- Description updated

#2 - 11/30/2012 05:24 PM - Mayer Michael

- Assigned To set to Mayer Michael

#3 - 11/30/2012 05:39 PM - Mayer Michael

- Start date set to 11/30/2012

- % Done changed from 0 to 10

I started to produce and edit the files GModelSpectralLogParabola.hpp and GModelSpectralLogParabola.cpp

#4 - 11/30/2012 05:45 PM - Knödlseider Jürgen

Great! Note that there should also be a GModelSpectralLogParabola.i in the pyext folder which will create the Python bindings for the model component. You also have to add the .i file to the list of includes in pyext/gammlib/model.i.

To compile the GModelSpectralLogParabola.cpp file, you need to add the file to the src/model/Makefile.am file. For the include file, you should do the same in include/Makefile.am.

I recommend launching ./autogen.sh after editing the Makefile.am files so that things will be clean.

#5 - 12/03/2012 04:27 PM - Mayer Michael

Ok great, thanks for this important note.

Adapting the code, I encountered two issues we should discuss how to deal with:

- In the Fermi Science Tools, the LogParabola model uses the parameters "norm", "alpha" and "beta". This seems to be inconsistent with other spectral models, where the normalization parameter is always named "Prefactor". I think that "index" and "curvature" would be more meaningful names than "alpha" and "beta". However, changing the parameter names would complicate crosschecks when using with xml-files produced by the Science Tools.
- The function "GModelSpectralLogParabola::flux(const GEnergy& emin, const GEnergy& emax) const" is not analytically to calculate (at least I did not find a way to do it). Therefore, we would need to implement numerical integration of the function (this is probably similar to the ExpPlaw). Should it be implemented directly in this function, or did you already think of another solution? In addition: I am curios if there is a reason why the function "plaw\_photon\_flux(double emin, double emax, double pivot, double index)" for the PowerLawModel was outsourced to "GTools"?



Michael Mayer wrote:

Ok great, thanks for this important note.

Adapting the code, I encountered two issues we should discuss how to deal with:

- In the Fermi Science Tools, the LogParabola model uses the parameters "norm", "alpha" and "beta". This seems to be inconsistent with other spectral models, where the normalization parameter is always named "Prefactor". I think that "index" and "curvature" would be more meaningful names than "alpha" and "beta". However, changing the parameter names would complicate crosschecks when using with xml-files produced by the Science Tools.

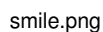
Right, things are unfortunately not done coherently in the ScienceTools. What I recommend here (and what I did already for other model components) is that two different parameter names are supported (e.g. GModelSpectralConst which supports Normalization and Value, see GModelSpectralConst::read and GModelSpectralConst::write). This ensures compatibility with Fermi XML files (which I think is very important), but also allows to move towards a new standard. Maybe we should at some point discuss this with Jim Chiang to see whether we can move towards a common standard.

- The function "GModelSpectralLogParabola::flux(const GEnergy& emin, const GEnergy& emax) const" is not analytically to calculate (at least I did not find a way to do it). Therefore, we would need to implement numerical integration of the function (this is probably similar to the ExpPlaw). Should it be implemented directly in this function, or did you already think of another solution? In addition: I am curious if there is a reason why the function "plaw\_photon\_flux(double emin, double emax, double pivot, double index)" for the PowerLawModel was outsourced to "GTools"?

Numerical integration is already available in GammaLib. To see how this works you may check the method GCTAResponse::irf\_extended. You first have to define a class that derives from GIntegrand and which implements an eval method with a single parameter (in the above mentioned case, this is done in the file GCTAResponse\_helpers.hpp. The class implements the kernel over which you want to integrate. To perform the integration, use something like

```
GIntegral integral(&integrand);
integral.eps(m_eps);
irf = integral.romb(rho_min, rho_max);
```

where integrand is an instance of the kernel class. GIntegral::eps allows to set the precision (is set to 1e-6 by default). And the GIntegral::romb performs the integral, where the 2 arguments are the minimum and maximum integration boundary.

I guess I should paste this small guide into a Wiki Howto section for developers 

Concerning plaw\_photon\_flux, the reason for outsourcing was that it's used by several classes (e.g. GModelSpectralFunc and GModelSpectralNodes, which make piecewise integrations).



**#7 - 12/03/2012 05:46 PM - Knödlseider Jürgen**

Started a [[Development HowTo]] in the Wiki and added a first item on [[How to integrate a function]].

**#8 - 12/04/2012 08:53 AM - Mayer Michael**

- % Done changed from 10 to 20

Thanks for the quick reply and solutions. It seems really easy to implement the integral following the tutorial. In which file should I implement the classes "integral\_logparabola\_photonflux" and "integral\_logparabola\_energyflux" (that's my suggestion to name them)? In the GModelSpectralLogParabola.hpp itself or also outsourced to GTools and with the function "double logparabola\_photon\_flux(...)"?

**#9 - 12/04/2012 11:52 AM - Knödlseider Jürgen**

I generally put these things directly in the class declaration (unless they have multiple usages), see for example GObservation which has class npred\_temp\_kern defined in the protected area.

Concerning the naming, I often use \_kern as suffix (referring to "kernel"). As the classes are subclasses of GModelSpectralLogParabola it's probably not necessary to add logparabola to the name, so you I would propose the shorter names flux\_kern and eflux\_kern (as they are used by the flux and eflux methods).

**#10 - 12/15/2012 09:41 PM - Knödlseider Jürgen**

- Target version set to 00-07-00

**#11 - 12/17/2012 04:25 PM - Mayer Michael**

- % Done changed from 20 to 90

- Estimated time set to 6.00

Hi Jürgen,

I finished the LogParabola model. After testing the new model extensively, I wanted to push it into the devel branch. I get the following error message:

```
git push
remote: [POLICY] Pushing to branch 'devel' is not allowed for user 'mmayer'.
remote: error: hook declined to update refs/heads/devel
To https://cta-git.irap.omp.eu/gammalib
! [remote rejected] devel -> devel (hook declined)
error: failed to push some refs to 'https://cta-git.irap.omp.eu/gammalib'
```

Maybe I am just to unexperienced with git. I did the following:

```
git checkout -b logparabola_model
git add <every file I changed>
git commit -m "my comment"
git checkout devel
git pull
git merge logparabola_model
git push
```

I also tried the following command: `git push https://mmayer@cta-git.irap.omp.eu/gammalib` which did not work either. Do you have an idea which mistake I made?



#12 - 12/17/2012 08:42 PM - Knödlseider Jürgen

Hi Michael,

After discussions with Christoph (see <https://cta-redmine.irap.omp.eu/boards/6/topics/22>) I changed the Git workflow. Now, pushing to devel is not permitted anymore. You should create a specific feature branch that you push to the repository. Then, you should put the status of the feature to Pull request.

Read more on the page [\[\[Git workflow for GammaLib\]\]](#).

If you have comments about the workflow, don't hesitate to post them on the forum <https://cta-redmine.irap.omp.eu/boards/6/topics/22>.

#13 - 12/18/2012 12:34 PM - Knödlseider Jürgen

- Status changed from New to Pull request
- % Done changed from 90 to 100

Code available in branch 605-Added-LogParabola-spectrum.

#14 - 12/18/2012 12:38 PM - Knödlseider Jürgen

The mc() method is not yet implemented.

I added action #654 for this reason. We should try to implement the method before shipping GammaLib-00-07-00.

#15 - 12/18/2012 12:50 PM - Knödlseider Jürgen

Integration is finished, the new code is now available in the devel branch.

Congratulations, Michael, for your first contribution to GammaLib!

#16 - 12/18/2012 12:50 PM - Knödlseider Jürgen

- Status changed from Pull request to Closed

#17 - 12/18/2012 03:42 PM - Mayer Michael

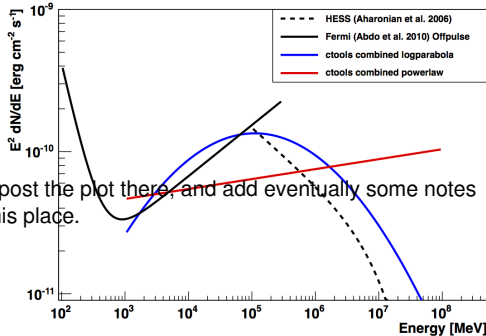
- File fermihess.png added

Thanks for your help and support to make this possible. I attached a preliminary plot showing first results with this spectral model. For this specific reason Fermi data (1yr) and HESS data (1 run), prepared by A. Schulz, were fitted with ctlike.

#18 - 12/18/2012 04:36 PM - Knödlseider Jürgen

Looks great!

I added a Wiki page [\[\[GModelSpectralLogParabola\]\]](#) under [\[\[Development notes\]\]](#). Could you post the plot there, and add eventually some notes related to the implementation? Ideally, we want to have all stuff related to code validation in this place.



Files

fermihess.png	104 KB	12/18/2012	Mayer Michael
---------------	--------	------------	---------------