GammaLib - Feature #1790

Add PowerLaw3 spectral model

06/14/2016 04:28 PM - Mayer Michael

Status:	Closed	**Start date:**	06/14/2016
Priority:	Normal	**Due date:**	
Assigned To:	Mayer Michael	**% Done:**	100%
Category:		**Estimated time:**	0.00 hour
Target version:	1.2.0		

Description

In addition to the PowerLaw2 spectral model, which uses the integral flux as free parameter, we could add a new spectral model: GModelSpectralPlaw3. This model stores the integral energy flux in units of [erg / cm2 / s] instead of the photon flux. The interface could be similar to GModelSpectalPlaw2 but using the parameter name EFlux or EnergyFlux instead of Integral.
One of the use case for me is the following: I often wanted to plot an SED with spectra where I only have the energy flux in a certain range provided. This model would allow to simplify the conversion from energy flux to an SED or differential spectrum.
I am still not certain about the name (same is true for GModelSpectralPlaw2). We could rename both into: GModelSpectralPlawFlux and GModelSpectralPlawEFlux. In the XML interface these models should also be renamed to be more intuitiv: "FluxPowerLaw", "EFluxPowerLaw" or "PowerLawFlux", "PowerLawEFlux".
Of course we should still support "PowerLaw2" for the Fermi notation.

History

#1 - 06/20/2016 10:03 AM - Mayer Michael

Would such a model be useful? Could we come up with a more meaningful name than "PowerLaw3"?

#2 - 06/21/2016 03:43 PM - Knödlseder Jürgen

This is certainly useful.

In terms of class names, how about

- GModelSpectralPlawPhotonFlux
- GModelSpectralPlawEnergyFlux

And for the XML we could have

```
<spectrum type="PowerLawPhotonFlux">
   <parameter scale="1e-07" name="PhotonFlux" min="1e-07" max="1000.0"    value="1.0" free="1"/>
   <parameter scale="1.0"   name="Index"      min="-5.0"  max="+5.0"      value="-2.0" free="1"/>
   <parameter scale="1.0"   name="LowerLimit" min="10.0"  max="1000000.0" value="100.0" free="0"/>
   <parameter scale="1.0"   name="UpperLimit" min="10.0"  max="1000000.0" value="500000.0" free="0"/>
</spectrum>
```

(where PowerLawPhotonFlux is a proxy for PowerLaw2 and PhotonFlux is a proxy for Integral), and

```
<spectrum type="PowerLawEnergyFlux">
   <parameter scale="1e-07" name="EnergyFlux" min="1e-07" max="1000.0"    value="1.0" free="1"/>
   <parameter scale="1.0"   name="Index"      min="-5.0"  max="+5.0"      value="-2.0" free="1"/>
   <parameter scale="1.0"   name="LowerLimit" min="10.0"  max="1000000.0" value="100.0" free="0"/>
   <parameter scale="1.0"   name="UpperLimit" min="10.0"  max="1000000.0" value="500000.0" free="0"/>
</spectrum>
```

#3 - 06/21/2016 03:46 PM - Mayer Michael

sounds good! I will try to make the changes (also to PowerLaw2)

#4 - 06/21/2016 09:57 PM - Knödlseder Jürgen

- Target version set to 1.2.0

#5 - 07/07/2016 11:45 AM - Mayer Michael

- Status changed from New to In Progress

I have started the excercise and renamed "GModelSpectralPlaw2" in "GModelSpectralPhotonFlux". This seems to work, however, I stumbled upon a problem when changing the type from "PowerLaw2" to "PowerLawPhotonFlux". The compatibility with the Fermi-LAT notation which uses "PowerLaw2" cannot be easily accomplished:
The GModelSpectalRegistry searches for a model instance by the type string. Therefore, when reading a Fermi-LAT XML file, the type "PowerLaw2" will not be found. I am not sure how to circumvent this problem without hard-coding in GModelSpectralRegistry sth like:

```
std::string modelname = "";
if (name == "PowerLaw2") {
   modelname == "PowerLawPhotonFlux";
}
else {
   modelname = name;
}
```

Do you have better ideas for a workaround?

#6 - 07/07/2016 01:03 PM - Mayer Michael

- % Done changed from 0 to 20

I also realised that the member attributes of m_last_g_integral and m_last_integral are actually not used in the code at all in GModelSpectralPlawPhotonFlux

#7 - 07/08/2016 10:42 AM - Mayer Michael

Another thing I was wondering about was in which unit we want to store the energy flux. I guess these are the two options:

1. MeV / cm2 / s
2. erg / cm2 / s

The first one would be more compliant with the GammaLib overall energy notation. The latter would be more convenient to astronomers, since erg are widely used in publications.

#8 - 07/08/2016 02:17 PM - Mayer Michael

- Assigned To set to Mayer Michael

- % Done changed from 20 to 80

I have finished the implementation of this feature. The changes include:

- Renamed PowerLaw2 into PowerLawPhotonFlux and GModelSpectralPlaw2 into GModelSpectralPlawPhotonFlux
- Added condition to GModelSpectralRegistry to support the Fermi-LAT notation (which uses PowerLaw2)
- Added class GModelSpectralPlawEnergyFlux
- Added unit test for GModelSpectralPlawEnergyFlux
- Added example file in $GAMMALIB/test/data/model_point_plaw_eflux.xml to be used by the unit tests
- Adapted the documentation (sphinx and doxygen)

I decided to use the unit erg/cm2/s for the energy flux.
All changes are available on branch *1790-GModelSpectralPlawEnergyFlux*.

I am currently computing pull distributions to verify the implementation.

#9 - 07/08/2016 03:44 PM - Mayer Michael

- File Pull_PlawEflux_Eflux.png added

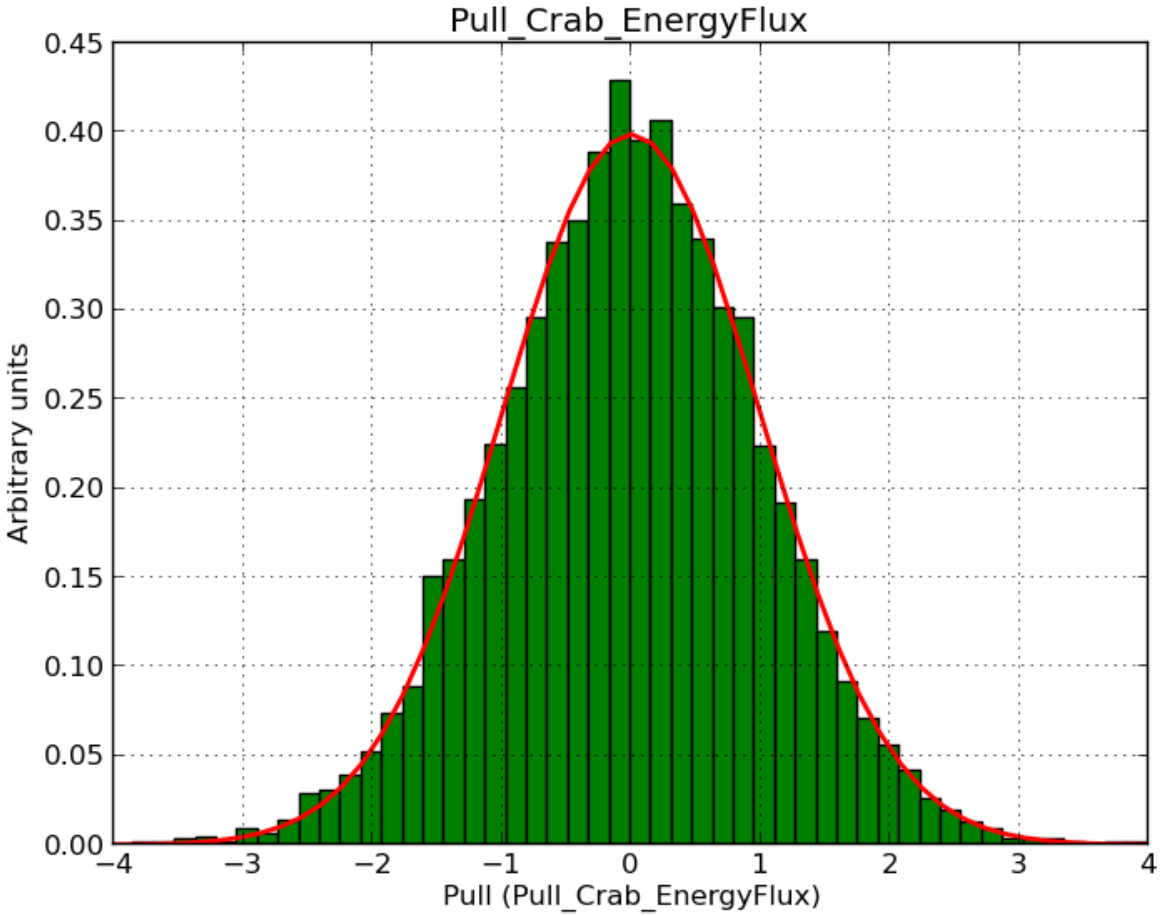
- File Pull_PlawEflux_Index.png added

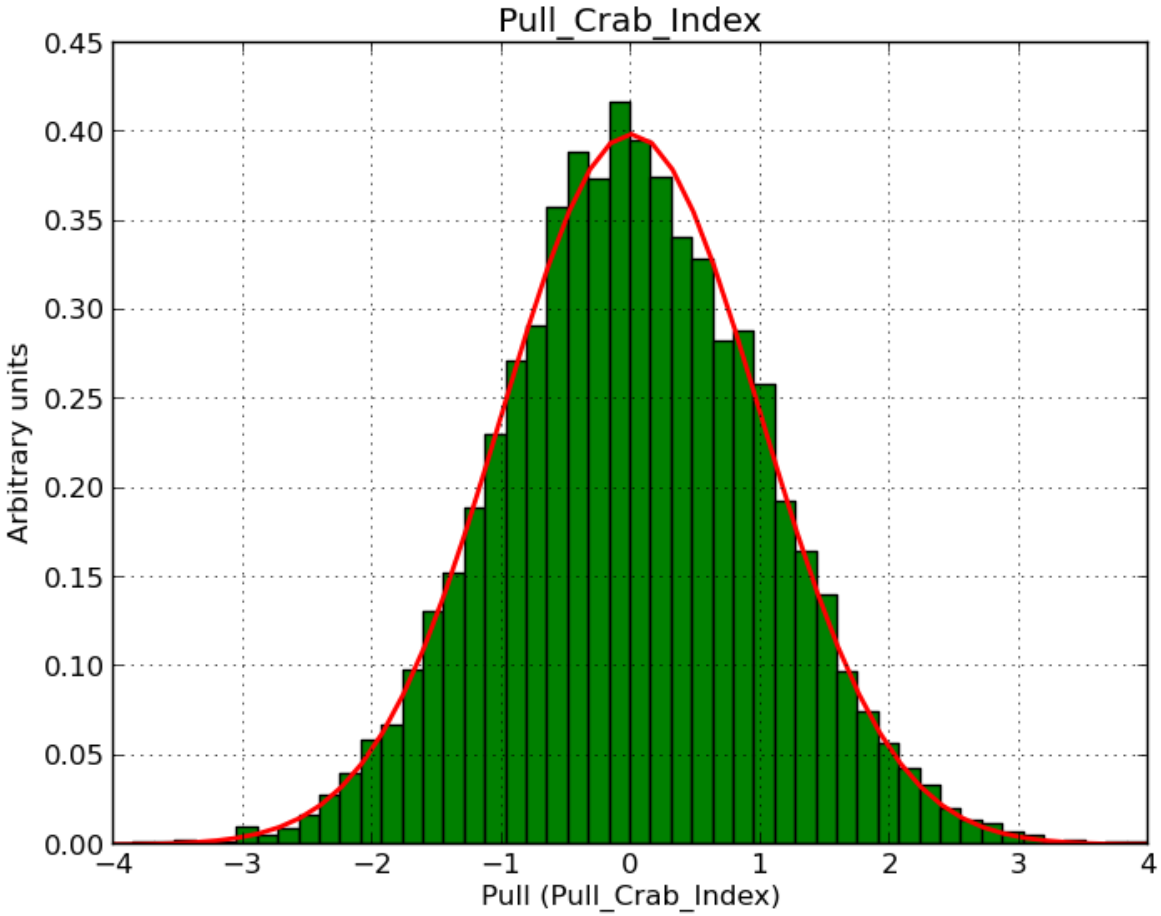
- Status changed from In Progress to Pull request

- Target version changed from 1.2.0 to 1.1.0

- % Done changed from 80 to 100

The pull distributions look good (see below). I have created 10000 trials using "CTA_South_50h" IRFs for an observation time of 5h per trial. The feature would thus be ready for integration. Does this still go into 1.1.0 or are you already working on the science verification?


Pull_Crab_EnergyFlux
Arbitrary units
0.45
0.40
0.35
0.30
0.25
0.20
0.15
0.10
0.05
0.00
−4
−3
−2
−1
0
1
2
3
4
Pull (Pull_Crab_EnergyFlux)


Pull_Crab_Index
Arbitrary units
0.45
0.40
0.35
0.30
0.25
0.20
0.15
0.10
0.05
0.00
−4
−3
−2
−1
0
1
2
3
4
Pull (Pull_Crab_Index)

#10 - 07/11/2016 12:26 PM - Knödlseder Jürgen

- Target version changed from 1.1.0 to 1.2.0

Thanks for implementing this. I would however prefer to defer this to release 1.2. Experience has shown that last minutes adds may be subject to trouble. I'm about to finish the package up for release, worked essentially on the out of source builds and release procedures to streamline the release and hence potentially increase the release frequency.

#11 - 07/11/2016 12:32 PM - Mayer Michael

Ok sounds reasonable, I agree with moving this to 1.2.0.
Unrelated I was adding some more documentation on #1646 in ctools, which might be nice to still make it into 1.1.0.

#12 - 07/18/2016 10:48 PM - Knödlseder Jürgen

user#77 wrote:

> Another thing I was wondering about was in which unit we want to store the energy flux. I guess these are the two options:
>
> 1. MeV / cm2 / s
> 2. erg / cm2 / s
>
> The first one would be more compliant with the GammaLib overall energy notation. The latter would be more convenient to astronomers, since erg are widely used in publications.

The eflux() methods return erg/cm2/s and I would tend to use the same units here for consistency.

#13 - 07/18/2016 10:51 PM - Knödlseder Jürgen

user#77 wrote:

> I have started the excercise and renamed "GModelSpectralPlaw2" in "GModelSpectralPhotonFlux". This seems to work, however, I stumbled upon a problem when changing the type from "PowerLaw2" to "PowerLawPhotonFlux". The compatibility with the Fermi-LAT notation which uses "PowerLaw2" cannot be easily accomplished:
> The GModelSpectalRegistry searches for a model instance by the type string. Therefore, when reading a Fermi-LAT XML file, the type "PowerLaw2" will not be found. I am not sure how to circumvent this problem without hard-coding in GModelSpectralRegistry sth like:
> [...]
> Do you have better ideas for a workaround?

What I typically to is that I append two instances of the same class to the registry, and example is the GCTAObservation class. This means however that the type string needs to be a member of the class, and that a constructor exist that allows setting the type string to a different value. You may add a string constructor for this purpose.

#14 - 07/19/2016 03:44 PM - Mayer Michael

Thanks for your feedback. I have implemented your suggestions:

1. The input energy flux is in erg/cm2/s
2. Added a dummy constructor to GModelSpectralPlawPhotonFlux using the std::string type (analogous to @GCTAObservation)
3. Added an instance using "PowerLaw2" type to the registry.
4. Added a unit test for "PowerLaw2" type.
5. Extended the file $GAMMALIB/test/data/model_point_plaw_pflux.xml to now contain both, a "PowerLawPhotonFlux" and a "PowerLaw2" model which can be used for testing purposes.

#15 - 07/19/2016 10:28 PM - Knödlseder Jürgen

Thanks a lot. I have now in preparation for release 1.1 already switched to the new naming convention and added an additional model to the registry for legacy. Legacy support is optional (enabled by default), and can be set via a ./configure option. You may check the code in the devel branch to see how this is setup.

Note that I have added special unit tests for legacy models with new XML files. When legacy support is switched off the test are skipped.

With having the old format in legacy XML files it is more obvious that this format should no longer be used.

#16 - 07/28/2016 02:38 PM - Mayer Michael

Thanks for the updates on the spectral registry. I have been trying to rebase on this branch on devel but it was a nightmare :).
Therefore, I created a new branch from devel with the modifications (which went much quicker).
The new branch is called *1790-new-GModelSpectralPlawEnergyFlux* and is ready to be merged. The changes include:

- Rename GModelSpectralPlaw2 into GModelSpectralPlawPhotonFlux
- Add GModelSpectralPlawEnergyFlux
- Add unit tests for GModelSpectralPlawEnergyFlux and adapt tests for GModelSpectralPlawPhotonFlux
- Add documentation for PowerLawEnergyFlux
- Both classes follow the latest update on the spectral registry parameter names

#17 - 07/28/2016 03:02 PM - Knödlseder Jürgen

Thanks, I could indeed imagine that the changes are not straight forward to merge in. I will look into that.

#18 - 07/28/2016 11:00 PM - Knödlseder Jürgen

- Status changed from Pull request to Closed

Looks that your rebasing was at the end quite successful. Integration of code was relatively smooth. Is now in devel.

Note that I renamed the photon_flux() and energy_flux() methods in flux() and eflux() for consistency with other class methods.

Files

Pull_PlawEflux_Eflux.png	46.3 KB	07/08/2016	Mayer Michael

Pull_PlawEflux_Index.png	45 KB	07/08/2016	Mayer Michael