GammaLib - Feature #1014

Implement GModelSpectralGauss

12/03/2013 10:39 PM - Deil Christoph

Status:	Closed	**Start date:**	12/03/2013
Priority:	Normal	**Due date:**	
Assigned To:	Owen Ellis	**% Done:**	100%
Category:		**Estimated time:**	0.00 hour
Target version:	00-08-00		

Description

To be able to test energy resolution (#1036) having a line-like spectral model is important.

The Fermi Science tools have a Gaussian: http://fermi.gsfc.nasa.gov/ssc/data/analysis/scitools/source_models.html#Gaussian

In the HESS software we have a TopHat:

```
/*! \class TopHat
    A "top-hat" FitFunction.
    \f[ f(E) = C \frac{1}{2} \left(Erf\left(\frac{\ln(E)-\mu_{On}}{\sigma_{On}}\right)+1\right)
    \left(1- \frac{1}{2} \left(Erf\left(\frac{\ln(E)-\mu_{Off}}{\sigma_{Off}}\right)+1\right) \f]
    Parameters :
    - #0 "Norm" C
    - #1 "OnLoc" \f$\mu_{On}\f$
    - #2 "OnWidth" \f$\sigma_{On}\f$
    - #3 "OffLoc" \f$\mu_{Off}\f$
    - #4 "OffWidth" \f$\sigma_{Off}\f$
  */
```

If no-one gets to it before, I could implemented and test this at the January coding sprint.

History

#1 - 12/11/2013 11:57 AM - Deil Christoph

- Description updated

- Assigned To set to Owen Ellis

- Priority changed from Low to Normal

#2 - 01/10/2014 10:48 AM - Deil Christoph

- Status changed from New to In Progress

- Assigned To changed from Owen Ellis to Deil Christoph

Ellis, Chia-Chun and I are currently implementing this here (work in progress):
https://github.com/cdeil/gammalib/compare/issue_1014

#3 - 01/10/2014 01:41 PM - Deil Christoph

Jürgen, how should we implement GModelSpectralGauss::mc?
(With Google we didn't find code that works for a given (min, max) energy range, just simple methods for all energies)

#4 - 01/10/2014 02:35 PM - Knödlseder Jürgen

Deil Christoph wrote:

> Jürgen, how should we implement GModelSpectralGauss::mc?

(With Google we didn't find code that works for a given (min, max) energy range, just simple methods for all energies)

I guess you then use a rejection method. Build a do-while loop around the random number generator and exit only if the energy is within the valid range. This can of course be costly in case that the energy range is narrow, but I guess very often min and max will enclose the bulk of the distribution.

#5 - 01/10/2014 03:02 PM - Owen Ellis

- Assigned To changed from Deil Christoph to Owen Ellis

- Target version set to 2nd coding sprint

#6 - 01/10/2014 06:11 PM - Owen Ellis

Progress update - implemented so far:

- GModelSpectralGauss::flux

Still to implement (in progress):

- GModelSpectralGauss::write (as in GModelSpectralExpPlaw::write)
- GModelSpectralGauss::flux (as used in Fermi Science Tools)
- GModelSpectralGauss::mc (no progress yet made)

#7 - 01/10/2014 06:36 PM - Deil Christoph

- Subject changed from Add GModelSpectralGaussian or GModelSpectralTopHat to Implement GModelSpectralGauss

#8 - 01/13/2014 07:03 PM - Owen Ellis

- Status changed from In Progress to Pull request

Could you please review this?

https://github.com/ellisowen/gammalib/compare/issue_1014

#9 - 01/13/2014 07:10 PM - Owen Ellis

- Status changed from Pull request to In Progress

I accidentally commited files that don't belong in the repo.
I'll make a new branch, re-add things and change the issue to "Pull request" once it's done.

#10 - 01/13/2014 07:33 PM - Owen Ellis

- Status changed from In Progress to Pull request

Please review this pull request:
https://github.com/ellisowen/gammalib/compare/issue_1014_B

#11 - 01/13/2014 07:57 PM - Owen Ellis

Also, I noticed the Doxygen docstrings in some cases aren't formatted correctly - I can correct these tomorrow if neccessary!

#12 - 01/13/2014 08:44 PM - Deil Christoph

One more thing that needs to be done that we forgot today: add GModelSpectralGauss to the user manual:
http://gammalib.sourceforge.net/user_manual/modules/model.html?highlight=curvature#spectral-models
https://github.com/gammalib/gammalib/blob/devel/doc/source/user_manual/modules/model.rst

#13 - 01/13/2014 09:58 PM - Knödlseder Jürgen

- % Done changed from 0 to 50

I'm about to merge in this one.

While going over the code I found that there are no analytical gradients coded for the Gaussian. I guess it won't be too difficult to include the relevant code. For the performance of the code and the precision of the results this would be crucial. Numerical gradients should only be used when there is no obvious way how to compute the analytical ones.

#14 - 01/13/2014 10:13 PM - Knödlseder Jürgen

- Status changed from Pull request to In Progress

Merged into devel.

I put the status back to In Progress to not forget the implementation of the analytical gradients.

What we should also do is a pull distribution test (using for example the cspull script) to make sure that the MC and the model are consistent. See for example https://cta-redmine.irap.omp.eu/projects/gammalib/wiki/GModelSpectralPlaw for a pull distribution for the power law.

#15 - 01/14/2014 02:44 PM - Deil Christoph

The GModelSpectralGauss::mc() method needs to be improved.

This shows that currently it sometimes generates negative energies (even though I pass a positive emin) and that it sometimes goes into an infinite loop:

```
import gammalib
model = gammalib.GModelSpectralGauss()
print(model)
emin = gammalib.GEnergy(1, 'MeV')
emax = gammalib.GEnergy(1e6, 'MeV')
time = gammalib.GTime()
ran = gammalib.GRan()
model.mc(emin, emax, time, ran)
print(model.mc(emin, emax, time, ran)) # call this repeatedly
```

#16 - 01/14/2014 02:54 PM - Knödlseder Jürgen

I was checking on the web: http://www.design.caltech.edu/erik/Misc/Gaussian.html

The algorithm given is

```
float x1, x2, w, y1, y2;

      do {
            x1 = 2.0 * ranf() - 1.0;
            x2 = 2.0 * ranf() - 1.0;
            w = x1 * x1 + x2 * x2;
      } while ( w >= 1.0 );

      w = sqrt( (-2.0 * log( w ) ) / w );
      y1 = x1 * w;
      y2 = x2 * w;
```

The implement algorithm is

```
  // Get uniform value between 0 and 1
  double u = ran.uniform();

  // Do ...
  do {
     double x1;
     double w;
     do{
        x1       = 2.0 * u - 1.0;
        double x2 = 2.0 * u - 1.0;
        w        = x1 * x1 + x2 * x2;
     } while (w >= 1.0);

     // Do ...
     w = std::sqrt( (-2.0 * std::log( w ) ) / w );

     // Compute ...
     double val = x1 * w;

     // Map into [emin,emax] range
     energy = (xmax - xmin) * val + xmin;

  } while ((xmin <= energy) && (energy < xmax));
```

One obvious difference is that the implemented version uses the same random number u while the Box-Muller transformation is expecting two independent random numbers. I'm also not sure that the mapping is correct. I think the value should be multiplied by the Gaussian sigma and adding the mean as offset. I guess the code should thus be:

```
  // Do ...
  do {
     double x1;
     double w;
     do{
        x1       = 2.0 * ran.uniform() - 1.0;
        double x2 = 2.0 * ran.uniform() - 1.0;
        w        = x1 * x1 + x2 * x2;
     } while (w >= 1.0);

     // Do ...
     w = std::sqrt( (-2.0 * std::log( w ) ) / w );

     // Compute ...
     double val = x1 * w;

     // Map to Gaussian parameters
     energy = m_sigma.value() * val + m_mean.value();

  } while ((xmin <= energy) && (energy < xmax));
```

Can you please check?

#17 - 01/14/2014 06:12 PM - Owen Ellis

Work in progress: https://github.com/ellisowen/gammalib/tree/issue_1014_gradients

Implemented analytical gradients (I still need to check/test this)

Also made changes suggested by Jürgen to the mc algorithm in the GModelSpectralGauss::mc() method (although I may make further changes due to the issues Christoph mentioned)

I also still need to add GModelSpectralGauss to the user manual and work on the pull distribution test(tomorrow)

#18 - 01/17/2014 12:48 AM - Knödlseder Jürgen

I'd like now to freeze the code for the gammalib-00-08-00 release. Any progress on the remaining issues? Do you think things will be finished by the end of this week?

#19 - 01/17/2014 02:07 AM - Knödlseder Jürgen

- File gauss.png added

I went ahead and integrated your actual branch in integration.

I recognized that there was an error in the mc() method that led to the endless loops: the while condition checked whether the energy was within the valid range, and not outside. So it continued looping until an energy outside [emin, emax] was found. The correct code is

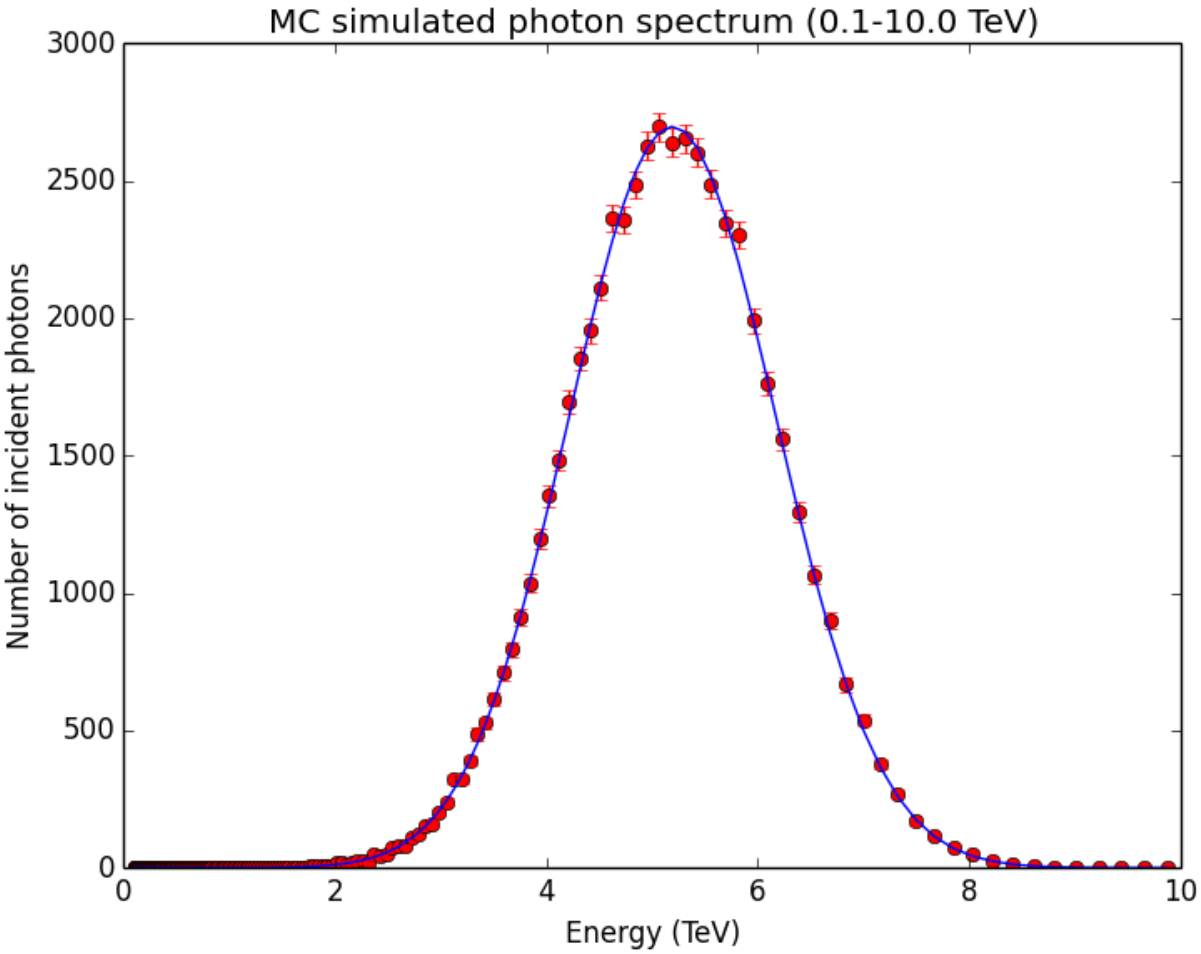
```
// Sample until we find a value within the requested energy range
do {
   double x1;
   double w;
   do {
      x1       = 2.0 * ran.uniform() - 1.0;
      double x2 = 2.0 * ran.uniform() - 1.0;
      w        = x1 * x1 + x2 * x2;
   } while (w >= 1.0);

   // Compute random value
   double val = x1 * std::sqrt((-2.0 * std::log(w)) / w);

   // Scale to specified width and shift by mean value
   energy = m_sigma.value() * val + m_mean.value();

} while (energy < xmin || energy > xmax);
```

In eval_gradients(), the gradients were formally correct, but there were two issues. First, gradients have to be computed with respect to the value factor and not the factor. A parameter value is the product between a scale and a value factor. The fit changes the value factor, hence gradients are needed for this component. This comes down to multiplying all gradients by the parameter scale. Furthermore, parameters which should not be fit need a zero gradient. Thus, the correct code is

```
double g_norm  = (m_norm.is_free()) ? term2 * eterm3 * m_norm.scale() : 0.0;
double g_mean  = (m_mean.is_free()) ? value * term4 * m_mean.scale() : 0.0;
double g_sigma = (m_sigma.is_free()) ? -term5 * eterm3 * (1.0 - (2.0 * term3)) * m_sigma.scale() : 0.0;
```

I also noted that you called the normalization of the Gaussian "Prefactor" in the XML file and gave it a unit of ph/cm2/s/MeV. However, as it's coded, the parameter is the normalization in units of ph/cm2/s. I thus renamed the parameter to "Normalization" and changed the unit.

I checked the new code by doing a simulation of a Gaussian at 5 TeV. Here the result:


MC simulated photon spectrum (0.1-10.0 TeV)
Number of incident photons
Energy (TeV)
0
500
1000
1500
2000
2500
3000
0
2
4
6
8
10

Looks quite reasonable. Now it remains to be see whether the fit works properly. This is done using a pull distribution.

#20 - 01/17/2014 02:31 AM - Knödlseder Jürgen

- % Done changed from 50 to 60

Merged into devel. Please start from there if you do further changes ...

#21 - 01/17/2014 10:43 AM - Knödlseder Jürgen

- % Done changed from 60 to 90

I started doing some pull distributions, and they look good. Will post when finished, but I think we can consider the feature as settled.

#22 - 01/17/2014 11:42 AM - Owen Ellis

Many thanks for your helpful comments! I think from my side there were no further changes or additions I planned to make.

#23 - 01/17/2014 12:02 PM - Knödlseder Jürgen

Okay. I guess the only thing still missing is a description of the model in the User Manual. A sub section would be needed in

doc/source/user_manual/modules/model.rst

Could you quickly write something up?

#24 - 01/17/2014 12:04 PM - Owen Ellis

Sure - I will do it this afternoon

#25 - 01/17/2014 01:38 PM - Owen Ellis

- Status changed from In Progress to Pull request

Done

https://github.com/ellisowen/gammalib/compare/issue_1014_usr_manual

#26 - 01/17/2014 02:09 PM - Knödlseder Jürgen

- Status changed from Pull request to Feedback

- Target version changed from 2nd coding sprint to 00-08-00

- % Done changed from 90 to 100

Great. Merged this in.

On my side, the pull distributions finished. Everything looks good. They are at [[GModelSpectralGauss]].

You model will go in the GammaLib-00-08-00 release. Congrats.

#27 - 01/19/2014 02:01 AM - Knödlseder Jürgen

- Status changed from Feedback to Closed

Files

gauss.png	47.7 KB	01/17/2014	Knödlseder Jürgen