GammaLib - Feature #1525

Use effective area information as background for CTA

09/04/2015 09:46 AM - Mayer Michael

Status:	Closed	**Start date:**	09/04/2015
Priority:	Normal	**Due date:**	
Assigned To:	Mayer Michael	**% Done:**	100%
Category:		**Estimated time:**	0.00 hour
Target version:	1.0.0		

Description

In order to test different background models and shapes for IACT data, we could add the option to use the actual acceptance shape for the background model. The acceptance shape is basically already provided by the effective area information.
Technically, this could simply be achieved by adding a new class GCTABackground2D, which is also able to read files of the GCTAAeff2D format.
The proper scaling would then also be handled by GCTAModelIrfBackground, e.g via the XML interface by providing a spectral component.

In particular, this could be an option for CTA, where the background shape will be hard to determine. The gamma-ray acceptance might offer a good approximation for this. I am interested to see how well (or bad) this works for current IACTs.

History

#1 - 09/04/2015 09:50 AM - Knödlseder Jürgen

You can in principle do this already by using an isotrop diffuse model as background component. But I agree that this will be time consuming as it involves a convolution.

#2 - 09/04/2015 09:56 AM - Mayer Michael

Ah, I understand your point and haven't thought about this option. But you are right: especially for a large datasets this could cause computing problems. I am also thinking about the binned/stacked analysis where we need to compute a background cube first. Currently, only background models and no sky models are used in ctbkgcube.
Do you think creating GCTABackground2D is a reasonable approach then?

#3 - 09/04/2015 10:10 AM - Knödlseder Jürgen

I'm not sure we need a new GCTABackground... class for that, won't it be sufficient to create a new model, e.g. GCTAModelAeffBackground (which should be pretty similar to GCTAModelIrfBackground)?

#4 - 09/04/2015 10:22 AM - Mayer Michael

Actually, I had this in mind as well (with exactly that classname smile.png). Going through the code however, I realised that the GCTAModelAeffBackground::mc-method would require some thinking as GCTAAeff doesn't contain a method to create randomised event directions. But I guess this can be easily adapted from GCTABackground?

#5 - 09/04/2015 10:26 AM - Knödlseder Jürgen

Mayer Michael wrote:

> Actually, I had this in mind as well (with exactly that classname smile.png). Going through the code however, I realised that the GCTAModelAeffBackground::mc-method would require some thinking as GCTAAeff doesn't contain a method to create randomised event directions. But I guess this can be easily adapted from GCTABackground?

Indeed, the mc method would need to be coded explicitly in GCTAModelAeffBackground.

#6 - 09/04/2015 10:27 AM - Mayer Michael

Ok, I will give it a try ...

#7 - 09/04/2015 10:34 AM - Mayer Michael

And I agree, using GCTAModelAeffBackground allows for more flexibility if the effective area format gets changed or extended eventually.

#8 - 09/04/2015 02:15 PM - Mayer Michael

- % Done changed from 0 to 20

I am thinking whether it makes sense to create an abstract GCTAAeff::mc() method for this purpose. The function could then be implemented by the individual GCTAAeff..-classes.
Creating a generic GCTAModelAeffBackground::mc method might be difficult as we know nothing about the effective area shape and possible parameter dependencies which may be added later on.
What do you think?

#9 - 09/04/2015 02:31 PM - Knödlseder Jürgen

As we do not need this for the other functionalities I would not do this (it creates an unnecessary burden to implement something for all classes that is generally not needed).

Can't we copy basically the mc method from GCTABackground3D?

But I see your problem: you can't have an computation stack as we're living in model space. But we could implement a simple rejection method, right? It would only require that we know the maximum Aeff value.

#10 - 09/04/2015 03:12 PM - Mayer Michael

> Can't we copy basically the mc method from GCTABackground3D? But I see your problem: you can't have an computation stack as we're living in model space. But we could implement a simple rejection method, right? It would only require that we know the maximum Aeff value.

You are right. I was wondering how to sample the Aeff function to find the maximum value. I guess for the energy range we could simply use the GEbounds that come with the observation, while for the offset, we could use the GCTAPointing and the GCTARoi to find the maximum range. The only thing we need is then a hard-coded sampling granularity in energy and offset.

#11 - 09/04/2015 03:21 PM - Knödlseder Jürgen

Mayer Michael wrote:

> > Can't we copy basically the mc method from GCTABackground3D? But I see your problem: you can't have an computation stack as we're living in model space. But we could implement a simple rejection method, right? It would only require that we know the maximum Aeff value.

> You are right. I was wondering how to sample the Aeff function to find the maximum value. I guess for the energy range we could simply use the GEbounds that come with the observation, while for the offset, we could use the GCTAPointing and the GCTARoi to find the maximum range. The only thing we need is then a hard-coded sampling granularity in energy and offset.

Why do you need any granularity for a rejection method?

#12 - 09/04/2015 03:30 PM - Mayer Michael

Sorry, I wasn't expressing myself clearly. I need the granularity for finding the maximum effective area:

```
 // Initialise maximum effective area
double max_aeff = 0.0;

// Granularity
int m_onodes = 20;

// Loop over offsets to find maximum effective area
for (int j = 0; j < m_onodes; ++j) {
    double theta = j * max_theta / (double(m_onodes) - 1.0);
    double aeff_theta = (*aeff)(logE, theta)
    if (aeff_theta > max_aeff) {
        max_aeff = aeff_theta;
    }
}
```

For randomly drawing an event energy, I would also use the GModelSpectralNodes::mc method, where I also need to define a number of nodes.

#13 - 09/04/2015 06:23 PM - Knödlseder Jürgen

I guess we can use the granularity of the Aeff matrix. Values are interpolated bilinearily, hence we have to simply find the largest value by looping for a given energy over all offaxis angles.

#14 - 09/04/2015 08:18 PM - Mayer Michael

> I guess we can use the granularity of the Aeff matrix. Values are interpolated bilinearily, hence we have to simply find the largest value by looping for a given energy over all offaxis angles.

That is true for the case of GCTAAeff2D. However, in the code, we do not know what kind of GCTAAeff we are dealing with (GCTAAeffArf, GCTAAeffPerfTable, GCTAAeff2D, or any future Aeff format). In principle, it should work for any given effective area which implements operator()(logE, theta, phi, zenith, azimuth, etrue), right? Or would you limit this feature to GCTAAeff2D?

#15 - 09/04/2015 08:51 PM - Knödlseder Jürgen

This is true. I'm wondering whether we may implement an GCTAAeff::max(GEnergy) method (or something like that) so that we do not have to recompute the value every time.

#16 - 09/07/2015 05:29 PM - Mayer Michael

- Status changed from New to Feedback

- % Done changed from 20 to 80

Good point to implement GCTAAeff::max().
I did this and I also basically finished the implementation of this feature on branch *1525-use_aeff-as_background*.

Simulating works fine so far. I have also added an example XML file (including a unit test): $GAMMALIB/inst/cta/test/data/cta_model_aeff_bgd.xml

I have however one problem I don't quite understand yet:
The problem occurred when testing GCTAModelAeffBackground with ctlike, i.e. fitting the simulated background. The spectral parameters of GCTAModelAeffBackground don't get optimised at all. I get the following output:

```
2015-09-07T15:02:00: +==================================+
2015-09-07T15:02:00: | Maximum likelihood optimisation |
2015-09-07T15:02:00: +==================================+
2015-09-07T15:02:01:    Parameter "Prefactor" has zero curvature. Fix parameter.
2015-09-07T15:02:01:    Parameter "Index" has zero curvature. Fix parameter.
2015-09-07T15:02:01:  >Iteration   0: -logL=39972.876, Lambda=1.0e-03
2015-09-07T15:02:01:  GOptimizerLM::iteration: All curvature matrix elements are zero.
2015-09-07T15:02:01:  >Iteration   1: -logL=39972.876, Lambda=1.0e-03, delta=0.000, max(|grad|)=0.000000
2015-09-07T15:02:01:    Free parameter "Prefactor" after convergence was reached with frozen parameter.
2015-09-07T15:02:01:    Free parameter "Index" after convergence was reached with frozen parameter.
2015-09-07T15:02:01:
2015-09-07T15:02:01:  +==================+
2015-09-07T15:02:01:  | Curvature matrix |
2015-09-07T15:02:01:  +==================+
2015-09-07T15:02:01:  === GMatrixSparse ===
2015-09-07T15:02:01:   Number of rows .....: 4
2015-09-07T15:02:01:   Number of columns .....: 4
2015-09-07T15:02:01:   Number of nonzero elements : 0
2015-09-07T15:02:01:   Number of allocated cells .: 512
2015-09-07T15:02:01:   Memory block size .....: 512
2015-09-07T15:02:01:   Sparse matrix fill .....: 0
2015-09-07T15:02:01:   Pending element .....: 0
2015-09-07T15:02:01:   Fill stack size .....: 0 (none)
2015-09-07T15:02:01:   0, 0, 0, 0
2015-09-07T15:02:01:   0, 0, 0, 0
2015-09-07T15:02:01:   0, 0, 0, 0
2015-09-07T15:02:01:   0, 0, 0, 0
2015-09-07T15:02:01:  Non-Positive definite curvature matrix encountered.
2015-09-07T15:02:01:  Load diagonal elements with 1e-10. Fit errors may be inaccurate.
2015-09-07T15:02:01:  Non-Positive definite curvature matrix encountered, even after diagonal loading.
2015-09-07T15:02:01:
2015-09-07T15:02:01: +=========================================+
2015-09-07T15:02:01: | Maximum likelihood optimization results |
2015-09-07T15:02:01: +=========================================+
2015-09-07T15:02:01: === GOptimizerLM ===
2015-09-07T15:02:01:  Optimized function value ..: 39972.876
2015-09-07T15:02:01:  Absolute precision .....: 0.005
2015-09-07T15:02:01:  Acceptable value decrease .: 2
2015-09-07T15:02:01:  Optimization status .....: curvature matrix not positive definite
2015-09-07T15:02:01:  Number of parameters .....: 4
2015-09-07T15:02:01:  Number of free parameters .: 2
2015-09-07T15:02:01:  Number of iterations .....: 1
```

```
2015-09-07T15:02:01:  Lambda .....: 0.001
2015-09-07T15:02:01:  Maximum log likelihood ....: -39972.876
2015-09-07T15:02:01:  Observed events  (Nobs) ...: 4123.000
2015-09-07T15:02:01:  Predicted events (Npred) ..: 1456.479 (Nobs - Npred = 2666.52)
2015-09-07T15:02:01: === GModels ===
2015-09-07T15:02:01:  Number of models .....: 1
2015-09-07T15:02:01:  Number of parameters ......: 4
2015-09-07T15:02:01: === GCTAModelAeffBackground ===
2015-09-07T15:02:01:  Name .....: Background
2015-09-07T15:02:01:  Instruments .....: CTA
2015-09-07T15:02:01:  Instrument scale factors ..: unity
2015-09-07T15:02:01:  Observation identifiers ...: all
2015-09-07T15:02:01:  Model type .....: "PowerLaw" * "Constant"
2015-09-07T15:02:01:  Number of parameters ......: 4
2015-09-07T15:02:01:  Number of spectral par's ..: 3
2015-09-07T15:02:01:   Prefactor .....: 1e-14 +/- 0 [1e-18,1e-10] ph/cm2/s/MeV (free,scale=1e-05,gradient)
2015-09-07T15:02:01:   Index .....: -2.7 +/- 0 [5,-5]  (free,scale=-1,gradient)
2015-09-07T15:02:01:   PivotEnergy .....: 1e+06 MeV (fixed,scale=1e+06,gradient)
2015-09-07T15:02:01:  Number of temporal par's ..: 1
2015-09-07T15:02:01:   Normalization .....: 1 (relative value) (fixed,scale=1,gradient)
```

It seems the parameters do not affect the fit, but manually changing the start values results in different npred values. One assumption might be that the factor_gradient might become too large, as the spatial component (which uses the effective area), which is in the order 1e12 or so.
Maybe that causes numerical problems in the curvature computation. However, I also tested with a large scaling factor in the code, which did not solve the problem. So I am a bit stuck at the moment.
Maybe you have an idea?

#17 - 09/14/2015 06:20 PM - Mayer Michael

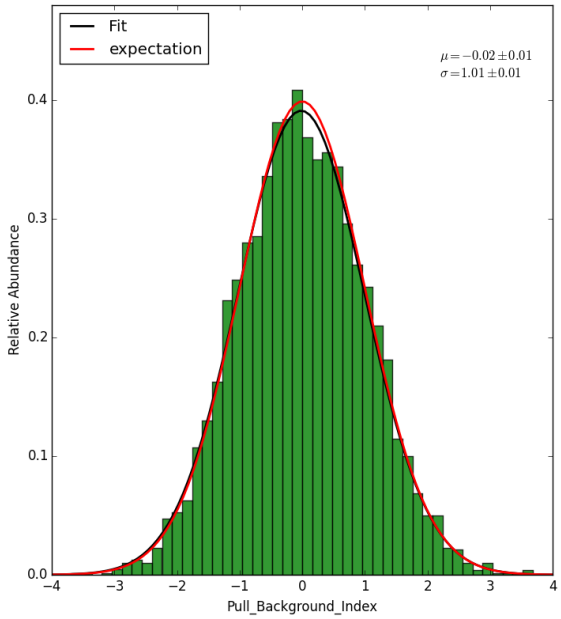
- File pull.png added

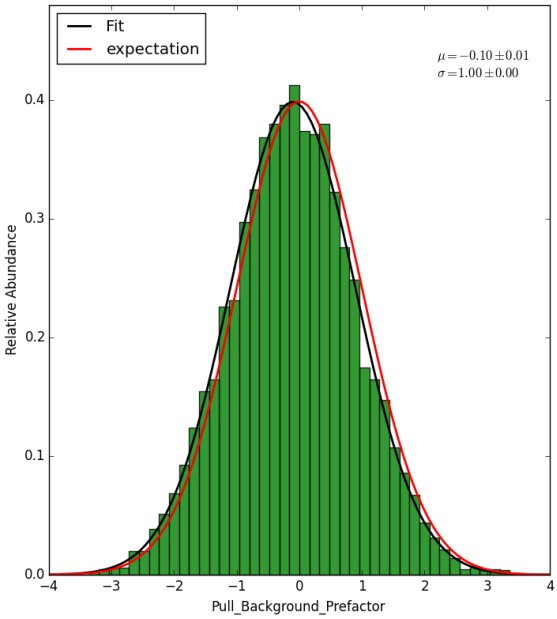
- Status changed from Feedback to Pull request

- Assigned To set to Mayer Michael

- % Done changed from 80 to 100

I have figured out the problem: it was a copy&paste error which caused that the parameter gradients where not calculated in the GCTAModelAeffBackground::eval_gradients() method.I fixed it and produced some pull distributions (see attachment)
All unit tests run successfully. The branch *1525-use_aeff-as_background* can thus be merged.





The slight shift of the pull distribution of the prefactor might be due to the approximation of gaussian statistics?

#18 - 10/02/2015 01:35 PM - Knödlseder Jürgen

I was looking into the code and I was wondering whether the Monte Carlo simulations of the background model were in fact correct. I see that you use:

```
offset = std::sqrt(ran.uniform()) * max_theta;
phi    = ran.uniform() * gammalib::twopi;
```

to draw a random offset angle and phi angle, but this does not give a uniform series of points in the DETX and DETY plane but a concentration towards small offset angles (the reasoning is that the area covered increases with offset angle). To generate a uniform density of points you would need to replace this by

```
cosrad = std::cos(max_theta);
offset = std::acos(1.0 - ran.uniform() * (1.0 - cosrad));
phi    = ran.uniform() * gammalib::twopi;
```

#19 - 10/02/2015 02:56 PM - Knödlseder Jürgen

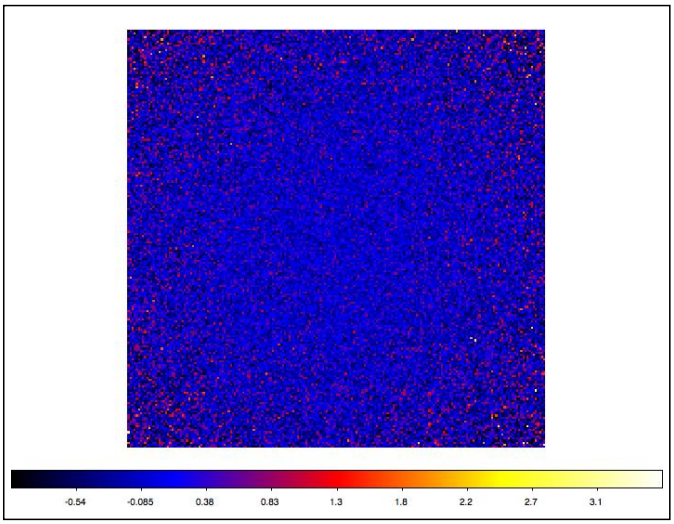
I overlooked the sqrt-term in your equation, which corresponds to the small angle approximation of my formula. Sorry for that.

I nevertheless propose to use the correct trigonometric formula.

#20 - 10/02/2015 02:57 PM - Knödlseder Jürgen

- File resmap-corrected.png added

Here a residual map obtained using the trigonometric formula. This looks okay.



#21 - 10/02/2015 05:07 PM - Knödlseder Jürgen

- Status changed from Pull request to Closed

- Target version set to 1.0.0

Merged into devel.

#22 - 10/05/2015 10:00 AM - Mayer Michael

I agree with you regarding the correct trigonometric formula. Thanks for merging.

Files

pull.png	95.8 KB	09/14/2015	Mayer Michael
resmap-corrected.png	284 KB	10/02/2015	Knödlseder Jürgen