GammaLib - Action #3202

Implement virtual default irf(GEvent&, const GSource&, GObservation& obs) method

04/09/2020 08:03 PM - Knödlseder Jürgen

Status:	Closed	**Start date:**	04/09/2020
Priority:	Normal	**Due date:**	
Assigned To:	Knödlseder Jürgen	**% Done:**	100%
Category:		**Estimated time:**	0.00 hour
Target version:	1.7.0		

Description

The method irf(GEvent&, GSource&, GObservation& obs) should be implemented as virtual method at the level of the GResponse class so that by default, all model types are available to an instrument class. The structure of the method should follow

```
double GResponse::irf(const GEvent&       event,
                      const GSource&      source,
                      const GObservation& obs) const
{
   // Initialise IRF value
   double irf = 0.0;

   // Select IRF depending on the spatial model type
   switch (source.model()->code()) {
      case GMODEL_SPATIAL_POINT_SOURCE:
         irf = irf_ptsrc(event, source, obs);
         break;
      case GMODEL_SPATIAL_RADIAL:
         irf = irf_radial(event, source, obs);
         break;
      case GMODEL_SPATIAL_ELLIPTICAL:
         irf = irf_elliptical(event, source, obs);
         break;
      case GMODEL_SPATIAL_DIFFUSE:
         irf = irf_diffuse(event, source, obs);
         break;
      case GMODEL_SPATIAL_COMPOSITE:
         irf = irf_composite(event, source, obs);
         break;
      default:
         break;
   }

   // Return IRF value
   return irf;
}
```

and default virtual methods should be implemented for irf_ptsrc, irf_radial, irf_elliptical, irf_diffuse and irf_composite that only make use of the irf(GEvent&, GPhoton&, GObservation&) method. These default methods can be overloaded when needed.

History

#1 - 04/11/2020 12:53 AM - Knödlseder Jürgen

- Status changed from New to In Progress

- % Done changed from 0 to 40

I implemented the new structure in GResponse and cleaned up the GCTAResponse class that contained before the irf_composite() member function, meaning that composite spatial models were only accessible to CTA. Now they should be accessible to all instruments.

I then implemented the GResponse::irf_radial() method that performs an integration over a radial model. The method works well for SPI data, results can be found in #742.

I also move the GCTAResponseCache class to obs module, renaming it in GResponseCache. It turned out that the energy encoding that worked for CTA is not possible for SPI due to the different energy scale of the instrument. Also, there was no guarantee that the encoding scheme implemented for CTA actually works, since a double precision value is a bit limited to hold the information of 5 floating point variables. I therefore reworked the caching scheme, implementing a three-level cache, with the first level being the source name (and observation ID), the second level the true and reconstructed event energies, and the third level the instrument direction.

The two energies are now converted into floats, meaning that the information is compressed into 2*32=64 Bits. The two floats are then mapped to a unsigned 64 Bit integer, which is used as key for the second level.

The same procedure is applied to the instrument direction. Here eventually three floating point variables will be combined in a 64 Bit integer, the details are left to the actual implementation for each instrument. a GInstDir::hash() method was added to provide the proper encoding of the instrument direction.

#2 - 04/11/2020 03:49 AM - Knödlseder Jürgen

Just for the record, to make the hashing into an unsigned int work on all platforms I had to use the C-style u_int64_t since uint64_t is only supported on C++11, which is not what is currently on the Continuous Integration platform. However, on openSolaris, u_int64_t did not exist, and there I had to use uint64_t. The following code in the relevant header files did the job:

```
#include <sys/types.h>
#if defined(__sun) && (defined(__svr4__) || defined(__SVR4)) // SOLARIS
   typedef uint64_t u_int64_t;
#endif // SOLARIS
```

#3 - 04/11/2020 10:59 PM - Knödlseder Jürgen

- % Done changed from 40 to 80

For clarity I renamed the

double GResponse::irf(const GEvent& event, const GSource& source, const GObservation& obs) const

method to

double GResponse::irf_spatial(const GEvent& event, const GSource& source, const GObservation& obs) const

so that irf is now only used for the photon response method. This should avoid the complex overloading where at the end it is not clear which method is called.

#4 - 04/13/2020 10:29 AM - Knödlseder Jürgen

- Status changed from In Progress to Pull request

- % Done changed from 80 to 100

I completed the implementation of the virtual default irf_spatial() method. The method is now used for INTEGRAL/SPI data analysis, results can be found on issue #742.

The default methods are not very fast, and may be inaccurate, but at least they provide some functionality that can be easily used. The results should however be checked carefully for any new instrument, and probably, it will be better to implement custom methods for a new instrument.

In particular, the current convolution of the spatial model with the IRF does not take into account the properties of the PSF. Consequently, the sampling may not be very efficient, and for small IRFs and large extended models, the PSF may be actually missed. Hence proper convolution should take into account the PSF, making sure that it is properly sampled.

But it's difficult to take into account the PSF at a general level, unless new methods are added that provide information about PSF properties. One possibility would be to return for an instrument direction the spatial PSF centre, since any instrument direction will have at some level an association sky direction, even for a non-imaging instrument (for example the telescope pointing direction for SPI or the scatter direction (Chi,Psi) for COMPTEL). The pointing direction together with a maximum radius and some information about the angular resolution can be used by a general convolution method to give accurate results. I created the issue #3204 to cover this issue.

#5 - 04/14/2020 02:13 PM - Knödlseder Jürgen

- Status changed from Pull request to Closed

Merged into devel.