GammaLib - Action #3333

Implement numerical integration of gradients for energy dispersion

09/08/2020 09:29 AM - Knödlseder Jürgen

Status:	Closed	**Start date:**	09/08/2020
Priority:	Normal	**Due date:**	
Assigned To:	Knödlseder Jürgen	**% Done:**	100%
Category:		**Estimated time:**	0.00 hour
Target version:	2.0.0		

Description

Currently the spectral gradients for energy dispersion are computed numerically. A faster approach is to numerically integrate the analytical spectral gradients. This can now be done using the GIntegrals and the GFunctions classes.

For this, the following changes need to be implemented:

- derive the GResponse::edisp_kern from GFunctions instead of GFunction
- implement GResponse::edisp_kern::eval(const double& etrue) method that returns an GNdarray object with value and function gradients
- use GIntegrals instead of GIntegral in GResponse::convolve() method and update m_grad member

History

#1 - 09/10/2020 12:54 PM - Knödlseder Jürgen

- Status changed from New to In Progress

- % Done changed from 0 to 20

I started implementing the numerical integration of gradients for energy dispersion, but when testing the code I got a segmentation fault. This segmentation fault seems to be related to the response cache:

```
$ ctlike edisp=yes
...
#0  0x000000010015b7be in ?? () from /usr/local/gamma/lib/libgamma.8.dylib
#1  0x00007ffeefbfdb50 in ?? ()
#2  0x0000000100246dd5 in std::__1::__tree<std::__1::__value_type<unsigned long long, std::__1::map<unsigned long long, double,
 std::__1::less<unsigned long long>, std::__1::allocator<std::__1::pair<unsigned long long const, double> > > >,
 std::__1::__map_value_compare<unsigned long long, std::__1::__value_type<unsigned long long, std::__1::map<unsigned long long, double,
 std::__1::less<unsigned long long>, std::__1::allocator<std::__1::pair<unsigned long long const, double> > > >, std::__1::less<unsigned long long>,
 true>, std::__1::allocator<std::__1::__value_type<unsigned long long, std::__1::map<unsigned long long, double, std::__1::less<unsigned long long>,
 std::__1::allocator<std::__1::pair<unsigned long long const, double> > > > >::__insert_node_at (this=<optimized out>, __parent=<optimized out>,
 __child=<optimized out>,
    __new_node=0x100c0a2e0)
    at /Applications/Xcode.app/Contents/Developer/Toolchains/XcodeDefault.xctoolchain/usr/bin/../include/c++/v1/__tree:2134
#3  std::__1::map<unsigned long long, std::__1::map<unsigned long long, double, std::__1::less<unsigned long long>,
 std::__1::allocator<std::__1::pair<unsigned long long const, double> > >, std::__1::less<unsigned long long>,
 std::__1::allocator<std::__1::pair<unsigned long long const, std::__1::map<unsigned long long, double, std::__1::less<unsigned long long>,
 std::__1::allocator<std::__1::pair<unsigned long long const, double> > > > >::operator[] (this=0x100c096b8, __k=<optimized out>)
    at /Applications/Xcode.app/Contents/Developer/Toolchains/XcodeDefault.xctoolchain/usr/bin/../include/c++/v1/map:1525
#4  GResponseCache::set (this=<optimized out>, name=..., dir=..., ereco=..., etrue=..., value=<optimized out>) at GResponseCache.cpp:254
```

#2 - 09/10/2020 09:35 PM - Knödlseder Jürgen

- % Done changed from 20 to 30

I checked that the segfault does not occur with the code version 1.7.1. It seems to be related to the use of the GIntegrals class.

#3 - 09/11/2020 03:04 PM - Knödlseder Jürgen

- Status changed from In Progress to Pull request

- % Done changed from 30 to 100

There was a bug in the code, and after fixing the bug, things work as expected. Code is ready to get merged into devel.

#4 - 09/11/2020 03:43 PM - Knödlseder Jürgen

- Status changed from Pull request to Closed

Merged into devel.