GammaLib - Action #1278

Implement write() method in GCTAResponseTable

07/12/2014 05:53 PM - Lu Chia-Chun

Status:	Closed	**Start date:**	07/12/2014
Priority:	Normal	**Due date:**	
Assigned To:	Lu Chia-Chun	**% Done:**	100%
Category:		**Estimated time:**	0.00 hour
Target version:	00-09-00		

Description

I've implemented the write() method in GCTAResponseTable.
It works for one-row multi-vector table such as psf2d, background3d etc.

Todo:
Add extra keywords in header?
Implement write() method for multi-row tables such as Aeffarf.

History

#1 - 07/12/2014 05:56 PM - Lu Chia-Chun

pull link
https://github.com/chiachun/gammalib/tree/1278-improve-GCTAResponseTable

#2 - 07/13/2014 03:35 PM - Lu Chia-Chun

- File test_write.py added

- File newfile.fits added

I've added
GCTAResponseTable::add_axis()
GCTAResponseTable::add_par()
GCTAResponseTable::Write()
GCTABackground3D::Write()
GCTABackground3D::Save()
the .i files.

Please

- Check the get_item function in GCTAResponseTable.i. I copied and pasted from GSkymap.i. Not sure if I did a right thing, although it seems to work.

- GCTAResponseTable::Write() now works for writing columns, but some header keywords are missing. I don't know how to handle this part. Class-specific keywords should be added by GCTABackground3D, but I don't know the best way to do it. Now the extension name is hard-coded in GCTAResponseTable::Write()

The product of GCTAResponseTable::Write() looks fine. (See atteched newfile.fits. It's produced by the attached test_write.py, which reads the irf_file.fits you gave me and create the newfile.fits.)
But GCTABackground3D::operator() always gives zero after the new file is loaded. Probably due to missing keywords in the header?

#3 - 07/13/2014 07:42 PM - Lu Chia-Chun

Fix a (important) bug in GCTAResponseTable::Write()

new link
https://github.com/chiachun/gammalib/tree/1278.2-improve-GCTAResponseTable

#4 - 07/15/2014 02:39 AM - Knödlseder Jürgen

I would use a method

void axis(const int& index, const int& bin, const double& lo, const double& hi);

to set an axis bin value as this allows then an automatic computation of the nodes also.

Concerning adding an axis, you should use the method name append() which is gammalib standard, i.e.

void append(const std::string& name, const std::string& unit, const int& length);

to append an empty axis. You may also write the variant

void append(const std::string& name, const std::string& unit, const std::vector<double>& axis_lo, const std::vector<double>& axis_hi);

to fill axis values automatically. Also, write(), read() and save() should be lower case.

What is add_par doing?

#5 - 07/15/2014 07:15 AM - Lu Chia-Chun

- My add_axis is exactly as what you suggested except the naming and the order of parameters. Please check line 108 in GCTAResponseTable.C

- add_par() is to add a new parameter column. The idea is the following:

```
// Create GCTAResponseTable
 GCTAResponseTable table = GCTAResponseTable();
 // Add axes: detx, dety, energy to the table
 table.add_axis(detx_lo, detx_hi, "DETX_LO","DETX_HI", "deg");
 table.add_axis(dety_lo, dety_hi, "DETY_LO", "DETY_HI", "deg");
 table.add_axis(eng_lo, eng_hi, "ENERG_LO", "ENERG_HI", "TeV");
 // Add parameter column
 table.add_par("BGD","1/s/MeV/sr");
 table.add_par("BGD_RECO","1/s/MeV/sr");
```

Then the parameters are set by looping the axes.

- For changing an axis element, I find it a bit complicated and not very useful. We have to reset the nodes, as you said. And we usually want to change more than one bins in the axis or even want to change the number of axis bins. So, I would just add only the add_axis() and add_par() methods.

#6 - 07/16/2014 04:45 PM - Knödlseder Jürgen

Lu Chia-Chun wrote:

> - My add_axis is exactly as what you suggested except the naming and the order of parameters. Please check line 108 in GCTAResponseTable.C
>
> - add_par() is to add a new parameter column. The idea is the following:
>
> // Create GCTAResponseTable
> GCTAResponseTable table = GCTAResponseTable();
> // Add axes: detx, dety, energy to the table
> table.add_axis(detx_lo, detx_hi, "DETX_LO","DETX_HI", "deg");
> table.add_axis(dety_lo, dety_hi, "DETY_LO", "DETY_HI", "deg");
> table.add_axis(eng_lo, eng_hi, "ENERG_LO", "ENERG_HI", "TeV");
> // Add parameter column
> table.add_par("BGD","1/s/MeV/sr");
> table.add_par("BGD_RECO","1/s/MeV/sr");
>
> Then the parameters are set by looping the axes.
>
> - For changing an axis element, I find it a bit complicated and not very useful. We have to reset the nodes, as you said. And we usually want to change more than one bins in the axis or even want to change the number of axis bins. So, I would just add only the add_axis() and add_par() methods.

So we should rename the methods to append_axis() and append_par() to be compliant with GammaLib method naming conventions (add would somehow imply the + operator).

#7 - 07/16/2014 04:49 PM - Lu Chia-Chun

Agree. Tell me what you think after reviewing my code. I will do the modification accordingly.

#8 - 07/16/2014 05:43 PM - Knödlseder Jürgen

I'm about to check the code. Here some comments:

You added a method

```
GCTAResponseTable rsp_table(void) const;
```

to GCTABackground3D to retrieve the response table. First, I propose to rename the method to table() for simplification (short method names without underscore should be used for public methods in GammaLib). Second, the interface definition should be

```
const GCTAResponseTable& table(void) const;
```

to avoid creating a copy. If you need a method to set a response table, you should add

```
void table(const GCTAResponseTable& table);
```

Why do you need the

```
    double mc_spatial_resolution(void) const;
    double mc_spectral_resolution(void) const;
    void   mc_spatial_resolution(const double& binsize);
    void   mc_spectral_resolution(const double& binsize);
```

methods? This is dangerous as you're exposing internal cache variables to the outside world. Internal cache should not be visible to the outside, that's just stuff the class uses for it's own purpose. I removed these for the moment (I guess there is a better solution for your specific problem).

I renamed in GCTAResponseTable add_axis() to append_axis() and changed the argument list so that only a single name is provided. Axis names are then constructed internally by appending "_LO" and "_HI", which is response table convention. I renamed in GCTAResponseTable add_par() to append_parameter(). I also changed the arguments to const references, which is GammaLib style (avoiding memory copies).

I also removed the cast in GCTABackground3D.i and added an automatic type conversion section to GCTABackground.i. Now Python should always return the correct type, no casting is needed.

#9 - 07/16/2014 05:50 PM - Lu Chia-Chun

double mc_spatial_resolution(void) const;
double mc_spectral_resolution(void) const;
void   mc_spatial_resolution(const double& binsize);
void   mc_spectral_resolution(const double& binsize);

These stuff were not added by me. I even don't know what they are!!! Weren't they in the devel branch?

And what should I do now ? Or you have done all the job?

#10 - 07/16/2014 05:59 PM - Knödlseder Jürgen

Lu Chia-Chun wrote:

> double mc_spatial_resolution(void) const;
> double mc_spectral_resolution(void) const;
> void   mc_spatial_resolution(const double& binsize);
> void   mc_spectral_resolution(const double& binsize);
>
> These stuff were not added by me. I even don't know what they are!!! Weren't they in the devel branch?
>
> And what should I do now ? Or you have done all the job?

Ok, no worries, then there isn't any problem. I'm about to merge in the code. Will tell you when it's done.

Just another thing: I see that all operators have been removed and the *getitem* and *setitem* methods implemented. This makes sense for the

```
    const double&      operator()(const int& element) const;
    double&            operator()(const int& element);
    const double&      operator()(const int& index, const int& element) const;
    double&            operator()(const int& index, const int& element);
```

operators that take integer arguments, but not for those taking doubles and that are used for interpolations. I'll put them back.

#11 - 07/16/2014 06:02 PM - Lu Chia-Chun

About the operators: Yes, you are right!

#12 - 07/16/2014 06:04 PM - Lu Chia-Chun

I copied and pasted all getitem and setitem methods from GSkymap.i. I actually don't know the swig language. Things seem to work.

#13 - 07/16/2014 06:09 PM - Knödlseder Jürgen

Lu Chia-Chun wrote:

> I copied and pasted all getitem and setitem methods from GSkymap.i. I actually don't know the swig language. Things seem to work.

That's called lucky programming biggrin.png

#14 - 07/16/2014 06:25 PM - Knödlseder Jürgen

- Status changed from Pull request to Resolved

- Target version set to 00-09-00

- % Done changed from 80 to 100

Merged into devel.

I propose to close this issue now, and open eventually a new one if more code changes are needed. You probably need to adapt slightly to the above mentioned interface changes.

We also should add some unit tests for the new functionalities. It's always good to do this immediately, so that we don't get an increasing fraction of untested code.

#15 - 07/16/2014 07:18 PM - Lu Chia-Chun

Hey! This is not fair! Not a matter of pure luck.
We should call it 'smart and wise programming'! wink.png

Knödlseder Jürgen wrote:

> Lu Chia-Chun wrote:
>
> > I copied and pasted all getitem and setitem methods from GSkymap.i. I actually don't know the swig language. Things seem to work.
>
> That's called lucky programming biggrin.png

#16 - 07/16/2014 07:22 PM - Lu Chia-Chun

- agree to close this issue.
- I am willing to write some unit tests but I don't really know what unit tests mean. What should be tested? Could you give a list of to-test things and an example?

Knödlseder Jürgen wrote:

> Merged into devel.
>
> I propose to close this issue now, and open eventually a new one if more code changes are needed. You probably need to adapt slightly to the above mentioned interface changes.
>
> We also should add some unit tests for the new functionalities. It's always good to do this immediately, so that we don't get an increasing fraction of untested code.

#17 - 07/16/2014 09:41 PM - Knödlseder Jürgen

Lu Chia-Chun wrote:

> - I am willing to write some unit tests but I don't really know what unit tests mean. What should be tested? Could you give a list of to-test things and an example?

In a unit test you want to test in principle all functionalities of a class. This is pretty ambitious, and we likely never will cover all test cases. The typical thing you want to do is to build a simple response table from scratch, then check whether you read back the correct values (the one that you used for the definition), then save the response table and load it back again, and check if you again have the same values. This is already quite comprehensive. You may also want to test the interpolation methods to check that you get the expected value.

For the testing purpose you would create a very simple response table, maybe a 2D table with 2-3 bins in each axis.

The test should be added to the test_CTA.cpp / .hpp files.

#18 - 07/20/2014 11:21 PM - Knödlseder Jürgen

- Status changed from Resolved to Closed

- Remaining (hours) set to 0.0

Files

test_write.py	1.17 KB	07/13/2014	Lu Chia-Chun
newfile.fits	73.1 KB	07/13/2014	Lu Chia-Chun