GammaLib - Action #3405

Implement vectorised response computation for energy dispersion

10/18/2020 10:57 AM - Knödlseder Jürgen

Status:	In Progress	**Start date:**	10/18/2020
Priority:	Normal	**Due date:**	
Assigned To:	Knödlseder Jürgen	**% Done:**	20%
Category:		**Estimated time:**	0.00 hour
Target version:			

Description

For the moment the vectorised response computation brings only a speed-up without energy dispersion. The code needs to be restructured to achieve also a speed-up with energy dispersion enabled.

Related issues:

Related to GammaLib - Action # 3390: Implement vectorised computation of ener...	**Rejected**	**10/14/2020**

History

#1 - 10/18/2020 02:06 PM - Knödlseder Jürgen

One possibility to do this at high level would be to add methods that GEventCube that return the number of energy bins and that allow iterating over all event bins for a given energy bin. This will allow for instruments where data are binned in energy to process energy bin after energy bin, using high-level energy dispersion information.

What would be needed is something like this:

```
GEventBin* GEventCube::first_event_bin(const int& iebin);
GEventBin* GEventCube::next_event_bin(const int& iebin);
```

Note that a

```
GEbounds   GEvents::ebounds(void) const;
```

method exists already at the level of the GEvents class, which allows accessing how many energy bins exist.
The methods can then be used as follows to loop over all events in an energy bin:

```
for (GEventBin* bin = first_event_bin(iebin); bin != NULL; bin = next_event_bin(iebin)) {
   ...
}
```

#2 - 10/19/2020 09:06 AM - Knödlseder Jürgen

I'm not sure that such methods are actually useful, since what counts is the minimisation of the spatial transformations.

#3 - 10/21/2020 09:22 PM - Knödlseder Jürgen

Here is the current performance of the code using energy dispersion. The values were taken from issue #3203.

Code	CPU	Iterations	logL
Reference	11385.7 s	2	122531.429
Using vector response, still event-by-event evaluation	10871.2 s	2	122531.429

#4 - 10/22/2020 12:01 PM - Knödlseder Jürgen

- Status changed from New to In Progress

- Assigned To set to Knödlseder Jürgen

- % Done changed from 0 to 10

Before starting the vectorisation I wrote down the current code status in the TN0003 and restructured a bit the GResponse class so that four virtual methods are now dealing with event-wise and vectorised event probability computation for the two cases of neglecting or using the energy dispersion. This allows overloading of the various computations by instrument-specific implementations.

```
virtual double  eval_prob_no_edisp(const GModelSky&    model,
                                   const GEvent&       event,
                                   const GEnergy&      srcEng,
                                   const GTime&        srcTime,
                                   const GObservation& obs,
                                   const bool&         grad) const;
virtual double  eval_prob_edisp(const GModelSky&    model,
                                const GEvent&       event,
                                const GTime&        srcTime,
                                const GObservation& obs,
                                const bool&         grad) const;
virtual GVector eval_probs_no_edisp(const GModelSky&    model,
                                    const GObservation& obs,
                                    GMatrixSparse*      gradients) const;
virtual GVector eval_probs_edisp(const GModelSky&    model,
                                 const GObservation& obs,
                                 GMatrixSparse*      gradients) const;
```

#5 - 10/22/2020 12:21 PM - Knödlseder Jürgen

- % Done changed from 10 to 20

As a first modification, the vector computation of analytical spatial model gradients was added. So far there is no support for the event wise spatial model gradients in the CTA interface, but at least the code structure makes such a support possible, and support can be implemented at CTA level.

Before doing so, we should make sure that the code works still as expected for all spatial models. Note that the make check is successful.

#6 - 10/22/2020 04:09 PM - Knödlseder Jürgen

Here the test results, where **_b** refers for former results obtained with the old code and **_a** refers to new results obtained with the new code.

Model	**CPU_b**	**Iter_b**	**logL_b**	**CPU_a**	**Iter_a**	**logL_a**	**Comments**
Disk	14.81	2	156240.662	14.80	2	156240.662	identical result
Gaussian	16.55	2	118106.615	16.25	2	118106.615	identical result
Ring	257.40	31	125762.471	263.59	31	125762.471	identical result
Shell	48.47	4	127541.952	49.38	4	127541.952	identical result
Profile	97.40	2	118004.032	105.36	2	118004.032	identical result
Gaussian (edisp)	10871.2	2	122531.429	10848	2	122531.429	identical result
Gaussian (unbinned)	3313.09	2	-11118399.291	3261.26	2	-11118399.291	identical result
Gaussian (binned)	1974.85	2	118107.706	1938.22	2	118107.706	identical result

#7 - 11/18/2020 11:03 PM - Knödlseder Jürgen

- Related to Action #3390: Implement vectorised computation of energy dispersion added

#8 - 03/14/2022 12:28 PM - Knödlseder Jürgen

- Target version deleted (2.0.0)