GammaLib - Action #745

Feature # 736 (Closed): Implement SPI background model

Implement simple background model for SPI

02/06/2013 04:48 PM - Knödlseder Jürgen

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

Description

History

#1 - 12/11/2013 10:20 PM - Knödlseder Jürgen

- Target version deleted (SPI sprint #1)

#2 - 04/05/2020 10:31 PM - Knödlseder Jürgen

- Assigned To set to Knödlseder Jürgen

- Target version set to 1.7.0

#3 - 04/07/2020 08:48 PM - Knödlseder Jürgen

- Status changed from New to In Progress

- % Done changed from 0 to 50

I implemented the GSPIModelDataSpace class that is a generic model for fitting of data space models, similar to what is provided by the spi_fitlib and used for example by spi_obs_fit. For the moment the orbit and dete methods are implemented, I still need to work to implement these other methods.

I did a comparison of the spi_obs_fit results and the ctlike results for fitting the same data (the 511 keV dataset using a Gaussian component for the bulge and three background components, where the first is GEDSAT fitted with orbit,dete, the second is a constant fitted with dete and the third is TIME1 fitted with dete). Below a comparison of the results.

The agreement is excellent, although there seem to be some differences in the fitted values of the TIME1 component, where spi_obs_fit gives values that differ from 1 while ctlike gives values very close to 1, as would be expected (the test was a refit of a fitted model, hence all model values should be 1.0). Despite these small differences, the statistical uncertainties agree very well.

Fitting with ctlike is significantly slower than with spi_obs_fit which is likely due to the more generic nature of the ctlike fit. It should be investigated whether some speed-up is possible (maybe the bookeeping in GSPIEventCube is too important).

Parameter	**spi_obs_fit**	**ctlike**
Observed events		49071010
Predicted events		49071010.000
Pointings	7082	7082
Model parameters	1806	1806
-logL	-249848994.151	-249848994.151
MLR or TS	264.76440	264.764407
Sky model	1.00000 +/- 0.06148	1.00000 +/- 0.06148
GEDSAT D000 Rev0019	1.00000 +/- 0.10410	1.00000 +/- 0.10410
GEDSAT D008 Rev0030	1.00000 +/- 0.12312	1.00000 +/- 0.12312
CONST D005	0.99995 +/- 1.19135	0.99999 +/- 1.19135
TIME1 D015	0.78317 +/- 3.08255	0.99999 +/- 3.08196
TIME1 D016	-1.33474 +/ - 16.68483	1.00000 +/- 16.68124

CPU seconds	9	30.7
H1 fitting	1	10
H0 fitting	2	14

#4 - 04/07/2020 08:51 PM - Knödlseder Jürgen

I just check: reducing temporarily the bookkeeping in GSPIEventCube to a minimum does not reduce the computing time, hence bookkeeping should not be an issue.

I guess the possible reason for the difference is the large number of parameters. GammaLib does not handle efficiently the gradients for a large number of parameters. For every event bin, all gradients have to be initialised to zero (which means that for every bin several 1000 zeros are written) and then the several 1000 gradient values are checked to fish out the gradient value that did change for a given event (it's just a single one!).

I cross-checked this by implementing an eval() method that directly operates on the gradient vector, avoiding to use the gradients stored in the model parameters. This gave the same fitting results after only 8 seconds, hence slightly faster than spi_obs_fit. This calls for a redesign of the code that allows using such an alternative eval() method.

#5 - 04/07/2020 10:33 PM - Knödlseder Jürgen

I added the methods GModel::has_eval_indices() and GModel::eval_indices() to allow for a more efficient computation.

GModel::has_eval_indices() echos a boolean flag m_has_eval_inx that can be set to inform the GObservation::model() method that a list of parameter indices that were updated by eval() is available. This list can then be access via the GModel::eval_indices() method that returns the vector of integers m_eval_inx. Both members, m_has_eval_inx and m_eval_inx, have to be managed by the clients, i.e. they need to set or reset the flag and the vector.

I implemented handling of these flags in GSPIModelDataSpace::eval(). Now ctlike for the above example takes less than 8 seconds. This is really great!

#6 - 04/07/2020 10:49 PM - Knödlseder Jürgen

- % Done changed from 50 to 70

Here the updated table after the above mentioned implementations:

Parameter	**spi_obs_fit**	**ctlike**
Observed events		49071010
Predicted events		49071010.000
Pointings	7082	7082
Model parameters	1806	1806
-logL	-249848994.151	-249848994.151
MLR or TS	264.76440	264.764407
Sky model	1.00000 +/- 0.06148	1.00000 +/- 0.06148
GEDSAT D000 Rev0019	1.00000 +/- 0.10410	1.00000 +/- 0.10410
GEDSAT D008 Rev0030	1.00000 +/- 0.12312	1.00000 +/- 0.12312

CONST D005	0.99995 +/- 1.19135	0.99999 +/- 1.19135
TIME1 D015	0.78317 +/- 3.08255	0.99999 +/- 3.08196
TIME1 D016	-1.33474 +/ - 16.68483	1.00000 +/- 16.68124
CPU seconds	9	7.8
H1 fitting	1	2
H0 fitting	2	3

#7 - 04/14/2020 10:40 AM - Knödlseder Jürgen

I added the POINT model.

What is still missing are:

- DATE
- EVCLASS
- GEDFAIL
- GEDANNEAL

#8 - 04/14/2020 10:19 PM - Knödlseder Jürgen

- Status changed from In Progress to Closed

- % Done changed from 70 to 100

I implemented the remaining methods and added some unit tests.

The code was merged into devel.