GammaLib - Feature #810

Optimize computations

03/28/2013 02:53 AM - Knödlseder Jürgen

Status:	Closed	**Start date:**	03/28/2013
Priority:	Normal	**Due date:**	
Assigned To:		**% Done:**	50%
Category:		**Estimated time:**	0.00 hour
Target version:	1.0.0		

Description

GammaLib performs a lot of computations invoking mathematical functions (e.g. sin(), cos(), pow(), log() etc.). All these functions are time consuming, hence they should be called as few as possible.

This feature aims in optimizing the GammaLib code to speed up computations. The actions defined were derived by analyzing the code using a profiler.

Subtasks:

Action # 813: Optimize GSparseMatrix::stack_push_column	**Rejected**
Action # 811: Add sin() and cos() cache to GSkyDir	**Closed**

History

#1 - 03/28/2013 02:55 AM - Knödlseder Jürgen

- Description updated

#2 - 03/29/2013 01:03 AM - Knödlseder Jürgen

- Target version set to 00-08-00

#3 - 04/12/2013 02:16 PM - Knödlseder Jürgen

- Status changed from New to In Progress

#4 - 12/10/2013 12:16 AM - Knödlseder Jürgen

- Target version deleted (00-08-00)

#5 - 01/13/2014 10:39 AM - Deil Christoph

I had a look for C / C++ benchmark frameworks and found these two:

- https://github.com/DigitalInBlue/Celero
- https://github.com/nickbruun/hayai

Writing down benchmarks and running them on Jenkins would in the long run be better than "manual" optimization.
The advantages are that Jenkins results are visible for everyone and thus are an "objective" measure whether introducing optimizations into the code makes sense.

Also what is fast today (calling by const reference or caching) on some machine / compiler might not be true for most machines / compilers in 2015 - 2025, where most CTA analyses will be run, and having a benchmark suite in place that can be run simply with make benchmark will allow to evaluate GammaLib performance easily.

There's also this new tool that allows to benchmark GammaLib from Python and track how speed changes over time:
https://github.com/spacetelescope/asv

These all look very easy to use ... I think we could make make an example gammalib-benchmark using Celero and / or hayai and / or asv in a separate repo to try it out in a day (and use it e.g. to benchmark energy resolution convolution or other things).

#6 - 07/11/2014 04:12 PM - Knödlseder Jürgen

- Target version set to 1.0.0

#7 - 02/10/2015 11:16 PM - Knödlseder Jürgen

- Status changed from In Progress to Closed

I don't see what should be done immediately here. Propose to close this now.