McLachlan

Erik Schnetter

Center for Computation & Technology, Louisiana State University, USA[*]

(Dated: August 18, 2009)

McLachlan is a free Einstein solver that uses the Cactus framework and the Einstein toolkit. This document describes the basic features of the code, and also how to obtain, build, and use the code.

I. MCLACHLAN

McLachlan is a free Einstein solver that uses the Cactus framework and the Einstein toolkit. McLachlan was developed by Erik Schnetter and Peter Diener with the help of Jian Tao and Ian Hinder. It was first described in [9], where (to our knowledge) the first fully fourth order accurate black hole evolution with adaptive mesh refinement is presented. The McLachlan web pages are located at [10].

II. OBTAINING MCLACHLAN

McLachlan uses the Cactus Software Framework and the Einstein Toolkit. Cactus organises applications into *thorns* (modules) that can be maintained independently of each other. In order to use McLachlan, it is necessary to obtain Cactus as well as a set of supporting thorns.

The subsections below describe how to obtain Cactus and other necessary thorns. McLachlan contains also a shell script `checkout.sh` that attempts to automate this. However, this script is very basic and does not handle errors well.

A. Tools

Cactus thorns are ususall stored in *repositories* that are managed by version control systems such as CVS [1], SVN (subversion) [2], or git [3].

Before getting the code, you will need to install the following software on your local system:

1. wget (or curl)
2. CVS
3. SVN
4. git
5. Perl

These are standard packages, and they should be easily available for all Linux systems.

---
[*]URL: http://www.cct.lsu.edu/~eschnett/McLachlan/; Electronic address: schnetter@cct.lsu.edu

B. Cactus

Cactus [4, 5] is a software framework that makes it possible to maintain parts of applications independent of each other, and combine them into an efficient code when building the application. Cactus is described at http://www.cactuscode.org/, and a new version of the of Cactus web site is currently being prepared at http://preview.cactuscode.org/download/.

To obtain Cactus itself as well as a set of basic thorns, follow the instructions at http://preview.cactuscode.org/download/. (Additional thorns specific to numerical relativity are also located elsewhere.) Don't use a particular thorn list for this; instead, download all the basic thorn that Cactus offers.

For reference, here is a brief overview over the commands to do this:

1. `wget http://preview.cactuscode.org/download/GetCactus`
2. `chmod a+x GetCactus`
3. `./GetCactus`

   This checks out Cactus itself (the *flesh*) into a new subdirectory `Cactus`. When asked, choose the *development version* of Cactus; use the default answer for all other questions.
4. `cd Cactus`
5. `make checkout`

   This checks out some arrangements with basic thorns for Cactus, including the important CactusEinstein arrangement. Again, use the default answer for all questions, except when you are asked for the second time whetyer you want to quit. In this case, quit.

C. Carpet

Carpet [6–8] is a *driver* for Cactus. A driver manages memory, handles parallelism, and performs I/O on behalf of the application. Carpet supports adaptive mesh refinement (AMR) and multi-block methods. Carpet is described at http://www.carpetcode.org/.

To obtain Carpet, follow the instructions at http://www.carpetcode.org/get-carpet.html. Please check out the *Development Version*, which is currently quite stable. (We are planning to release a new stable version soon.)

In particular, the commands to obtain the development version are:

1. `cd Cactus`
2. `git clone -o carpet git://carpetcode.dyndns.org/carpet.git`
3. `cd arrangements`
4. `ln -s ../carpet/Carpet* .`

(Don't miss the dot after the `Carpet*` in the last line.) Note that Carpet should be checked out into the main Cactus directory, and the `arrangements` subdirectory needs to contain symbolic links pointing into the `carpet` directory.

D. McLachlan

McLachlan [9, 10] in an Einstein solver. It uses one of the BSSN formulations of the Einstein equations. McLachlan is described at http://www.cct.lsu.edu/~eschnett/McLachlan/, which is where you may have obtained this documentation.

To obtain McLachlan, issue the following commands:

1. `cd Cactus`
2. `cd arrangements`
3. `git clone git://carpetcode.dyndns.org/McLachlan.git`

Note that McLachlan needs to be checked out directly into the `arrangements` subdirectory.

McLachlan uses the Kranc code generation package [11–14]. Kranc also contains some thorns that McLachlan needs. (However, it is not necessary to run Kranc in order to use McLachlan. It is only necessary to run Kranc if McLachlan is modified.)

To obtain Kranc, issue the following commands:

1. `cd Cactus`
2. `git clone http://www.aei.mpg.de/~ianhin/kranc.git`
3. `cd arrangements`
4. `ln -s ../kranc/Auxiliary/Cactus/KrancNumericalTools .`

(Don't miss the dot at the end of the last line.) Note that Kranc needs to be check out into the main Cactus directory, and the `arrangements` subdirectory needs to contain symbolic links pointing into the `kranc` directory.

E. Other Thorns

All other thorns, including the public Whisky thorns, can be obtained via the GetCactus script that was downloaded above (see section II B):

1. `cd Cactus`
2. `cd ..`
3. `./GetCactus Cactus/arrangements/McLachlan/doc/mclachlan-public.th`

   Use the default answer for all questions.

F. Consistency Check

The *thorn list* `mclachlan-public.th` lists the thorns that are necessary for a simple spacetime evolution. The thorns are grouped into arrangements. All thorns listed in this file must now be present in the `arrangements` subdirectory of the main Cactus directory.

III. BUILDING MCLACHLAN

Building McLachlan and the other thorns requires C, C++, and Fortran 90 compilers, MPI, as well as the BLAS, GSL, HDF5, and LAPACK libraries.

A. Documentation

It is best to begin building Cactus with building documentation:

1. `cd Cactus`
2. `make UsersGuide`

This creates the users' guide as `doc/UsersGuide.pdf`.

All Cactus commands are listed with

1. `make help`

B. Option List

To build a Cactus application one needs to create an *options list.* This is a text file containing the configuration options that tell Cactus what compilers and compiler options to use, and where MPI and the auxiliary libraries are installed. This process is very specific to each machine and may require some trial and error.

We distribute options lists for a range of machines that we are using on the Cactus web site at http://preview.cactuscode.org/download/configfiles/. Option lists are also available together with the Simulation Factory [15] at https://svn.cct.lsu.edu/repos/numrel/simfactory/optionlists/.

C. Building

To build a Cactus application one starts with an option list and a thorn list. The text below assumes that you have an option list called `einstein-redshift-gcc.cfg`.

To configure an application

1. `cd Cactus`

2.
   ```
   make sim-config options=redshift-gcc.cfg \
   THORNLIST=arrangements/McLachlan/doc/mclachlan-public.th
   ```
3. `make sim`

This is necessary only once, or when the configuration options change. This will create an application called `sim`; of course, the name could also be different. Different applications, e.g. with different options or different thorn lists, can exist side by side.

To build the application:

1. `cd Cactus`
2. `make sim -j4`

The make option `-j4` builds 4 files at the same time. Use this option if you have several processors available; this will speed up building the application. The executable is called `cactus_sim` and is placed in the `exe` subdirectory.

D. Cleaning

You can also clean the application, removing all object files but keeping the configuration options and thorn list:

1. `cd Cactus`

2. `make sim-realclean`

IV. RUNNING MCLACHLAN

To run the McLachlan code, one needs a *parameter file.* Parameter files select which thorns are activated at run time, and what values the thorns' run-time parameters have. They typically have a `.par` suffix. Below, we use the parameter file `ks-mclachlan-public.par`.

Cactus applications are started like a regular MPI application. The exact mechanism depends on the particular MPI implementation. On Redshift, the command is

1. `cd Cactus`

2. `mkdir simulations`

3. `cd simulations`

4.
   ```
   env OMP_NUM_THREADS=1 mpirun -np 1 ../exe/cactus_sim \
       ../arrangements/McLachlan/doc/ks-mclachlan-public.par
   ```

This parameter file simulates a single, stationary, spinning black hole in Kerr-Schild coordinates. It requires about 4 GByte of RAM to run.

Note: If the options list enables OpenMP, then the Cactus application will be multi-threaded. Multi-threading can improve performance and reduce memory consumption, especially when many (> 100) cores are used. However, it is usually a bad idea to over-subscribe cores by having too many threads per node. It is usually best to choose both the number of MPI processes per node and the number of OpenMP threads per process such that their product equals the number of cores on a node. The way in which these numbers are chosen depend on the MPI implementation.

[1] CVS: Concurrent Versions System, URL http://www.nongnu.org/cvs/.
[2] SVN: Subversion, URL http://subversion.tigris.org/.
[3] Git: Fast version control system, URL http://www.git-scm.org/.
[4] T. Goodale, G. Allen, G. Lanfermann, J. Massó, T. Radke, E. Seidel, and J. Shalf, *The Cactus framework and toolkit: Design and applications*, in *Vector and Parallel Processing – VECPAR'2002, 5th International Conference, Lecture Notes in Computer Science* (Springer, Berlin, 2003).
[5] Cactus, Cactus Computational Toolkit, URL http://www.cactuscode.org/.
[6] E. Schnetter, S. H. Hawley, and I. Hawke, *Evolutions in 3D numerical relativity using fixed mesh refinement*, Class. Quantum Grav. **21**, 1465 (2004), arXiv:gr-qc/0310042, URL http://arxiv.org/abs/gr-qc/0310042.
[7] E. Schnetter, P. Diener, E. N. Dorband, and M. Tiglio, *A multi-block infrastructure for three-dimensional time-dependent numerical relativity*, Class. Quantum Grav. **23**, S553 (2006), arXiv:gr-qc/0602104, URL http://arxiv.org/abs/gr-qc/0602104.
[8] Carpet, mesh Refinement with Carpet, URL http://www.carpetcode.org/.

[9] D. Brown, P. Diener, O. Sarbach, E. Schnetter, and M. Tiglio, *Turduckening black holes: an analytical and computational study*, Phys. Rev. D **79**, 044023 (2009), arXiv:0809.3533 [gr-qc], URL http://arxiv.org/abs/0809.3533.

[10] McLachlan, McLachlan, a Public BSSN Code, URL http://www.cct.lsu.edu/~eschnett/McLachlan/.

[11] C. Lechner, D. Alic, and S. Husa, *From tensor equations to numerical code — computer algebra tools for numerical relativity*, in *SYNASC 2004 — 6th International Symposium on Symbolic and Numeric Algorithms for Scientific Computing, Timisoara, Romania* (2004), cs.SC/0411063, URL http://arxiv.org/abs/cs.SC/0411063.

[12] S. Husa, I. Hinder, and C. Lechner, *Kranc: a Mathematica application to generate numerical codes for tensorial evolution equations*, Comput. Phys. Comm. **174**, 983 (2006), gr-qc/0404023.

[13] Kranc: Automated Code Generation, URL http://numrel.aei.mpg.de/Research/Kranc/.

[14] Kranc, Kranc: Automated Code Generation, URL http://www.cct.lsu.edu/~eschnett/Kranc/.

[15] SimFactory, SimFactory: Herding Numerical Simulations, URL http://www.cct.lsu.edu/~eschnett/SimFactory/.