Automatic code generation.

Peter Diener

Center for Computation and Technology
and
Department of Physics and Astronomy

Louisiana State University

Collaborators: Erik Schnetter, Jian Tao, Frank Löffler.

MICRA 2009

Nils Bohr Institutet

Copenhagen

October 27, 2009

Punchline.

Automatic code generation can save you a lot of time and hassle.

Kranc.

Kranc is a set of mathematica scripts developed initially by Sascha Husa and Christiane Lechner and currently further developed by Ian Hinder for converting a set of tensorial evolution equations into a complete Cactus thorn.

Originally it was created in order to allow easy experimentation with different formulations of the Einstein equations.

Kranc produces a complete Cactus thorn including the configuration files.

Kranc interfaces with MoL and one of it's main functions is to produce the RHS evaluation routine for the evolution equations.

The user has to use Kranc mathematica routines to define tensors and their properties and how they relate to the Cactus grid functions.

In addition there are routines to define Cactus parameters.

The user defines "Calculations" to operate on the tensors along with scheduling information.

Kranc performs a number of optimizations before the actual C-code is generated.

Kranc currently does the finite differencing for you but we also have an interface to externally defined finite differencing operators (incomplete/untested).

McLachlan.

We have implented the BSSN evolution equations using Kranc.

The Kranc source for McLachlan is 782 lines long (excluding comments and empty lines). This expands out to 5874 lines of C-code.

The advantage of McLachlan compared to the previous code (CCATIE):

1. It is much easier to maintain and extend.
2. High order finite differencing can be generated easily.
3. It was designed to support a mixed MPI/OpenMP programming model.
4. It is easier to experiment with architecture specific optimizations.

McLachlan.

An example:

```
Gt[ua,lb,lc] -> 1/2 gtu[ua,ud]
                 (PD[gt[lb,ld],lc] + PD[gt[lc,ld],lb]
                  - PD[gt[lb,lc],ld]),
```

becomes:

```
Gt111  =  khalf*(gtu11*PDstandardNth1gt11
       + 2*(gtu21*PDstandardNth1gt12
       + gtu31*PDstandardNth1gt13)
       - gtu21*PDstandardNth2gt11
       - gtu31*PDstandardNth3gt11);

Gt211  =  khalf*(gtu21*PDstandardNth1gt11
       + 2*(gtu22*PDstandardNth1gt12
       + gtu32*PDstandardNth1gt13)
       - gtu22*PDstandardNth2gt11
       - gtu32*PDstandardNth3gt11);
```

plus 16 similar expressions for the remaining components of `Gt[ua,lb,lc]`.

McLachlan.

McLachlan is freely available.

Contact me (diener@cct.lsu.edu) or Erik (schnetter@cct.lsu.edu) for information on how to download it.