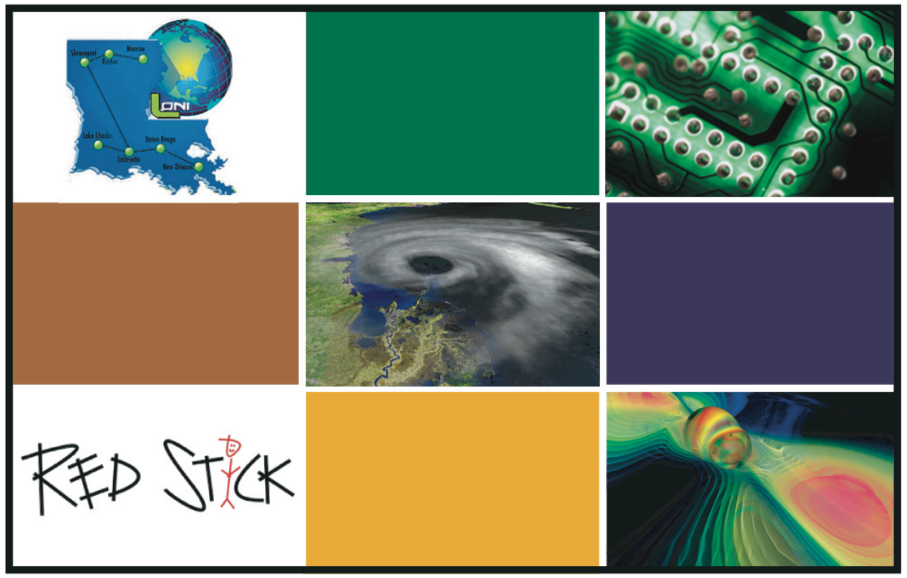
Kranc:
Automatic Code Generation for the Cactus Framework

Erik Schnetter, Ian Hinder

Baton Rouge, March 2008

CENTER FOR COMPUTATION
& TECHNOLOGY

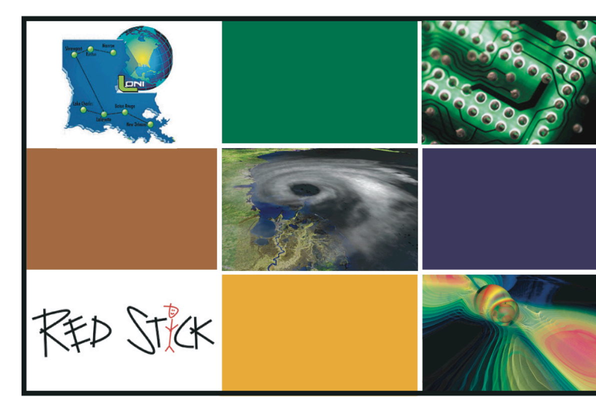


LSU

CENTER FOR COMPUTATION
& TECHNOLOGY

KRanc Assembles Numerical Code

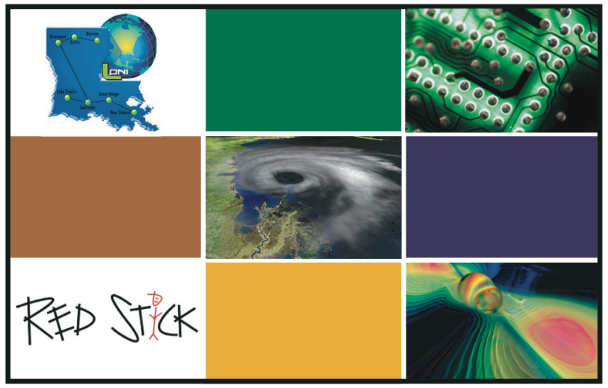
- Idea/dream: Input equations in a natural manner, *turn the crank*, output computer code ready to run on big machines
- Original goal: Analyse different formulations of the Einstein equations
- Originally developed 2002 by Sascha Husa (AEI), Christiane Lechner (AEI), Ian Hinder (Soton/PSU)



CENTER FOR COMPUTATION
& TECHNOLOGY

Kranc Overview

- Written in Mathematica, and input equations in Mathematica
- Directly generate complete Cactus thorns, using standard Cactus toolkits
- Handle tensor expressions in abstract index notation
- Discretisation via finite differences



LSU CENTER FOR COMPUTATION & TECHNOLOGY

Cactus

- Framework for HPC: code development, simulation control, analysis
- Manage increased complexity with higher level abstractions, e.g. for inter-node communication, intra-node parallelisation
- Active user community, 10+ years old
- Supports collaborative development

LSU
CENTER FOR COMPUTATION
& TECHNOLOGY

Cactus Framework

parallelism

memory management

I/O

SOR solver

your computational tools

multigrid

interpolation

reduction

extensible APIs

ANSI C

parameters

schedule

grid variables

make system

error handling

coordinates

boundary conditions

AMR

CFD

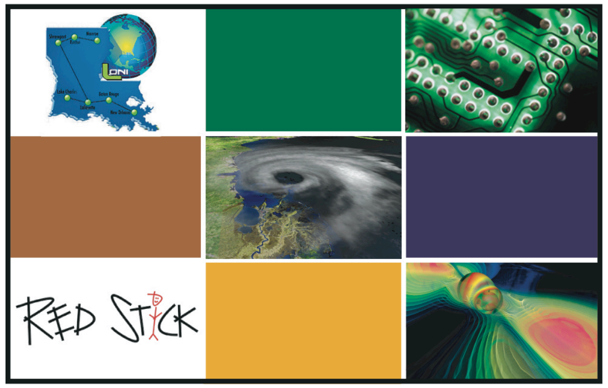
wave equation

Einstein equations

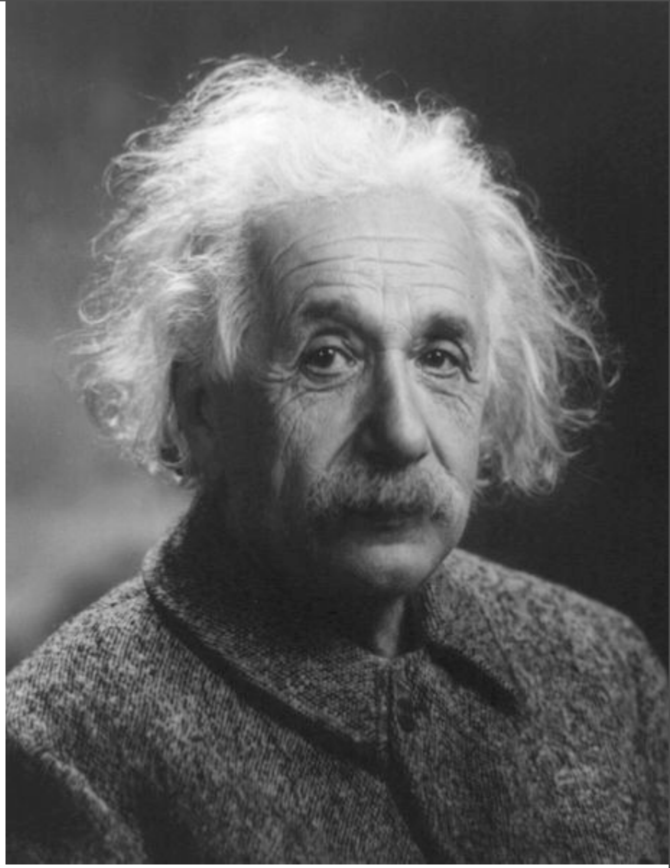
your physics

remote steering

Core *flesh* with plug-in *thorns*



Einstein Equations

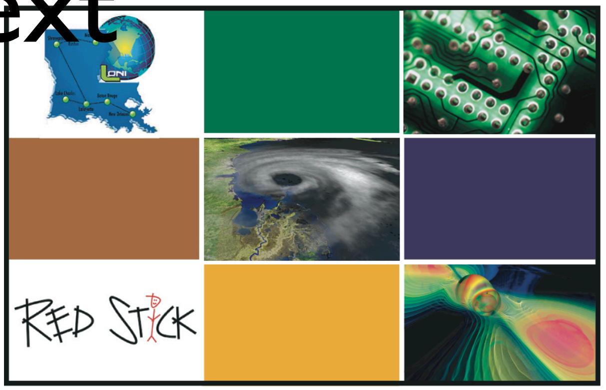


- 1915: General Relativity
- 10 coupled non-linear wave equations
- If written out explicitly, thousands of terms
- Can be written as hyperbolic PDE plus elliptic constraint equations ("3+1")
- If done naively, PDE are ill-posed

CENTER FOR COMPUTATION
& TECHNOLOGY

Example: McLachlan

- Einstein code implemented with Kranc
- 1,000 lines of Mathematica (short!) (generating 10k lines, replacing 25k lines)
- Implemented in two weeks (short!)
- Main problem: "read" equations in papers ("reading" requires unspecified context knowledge!)



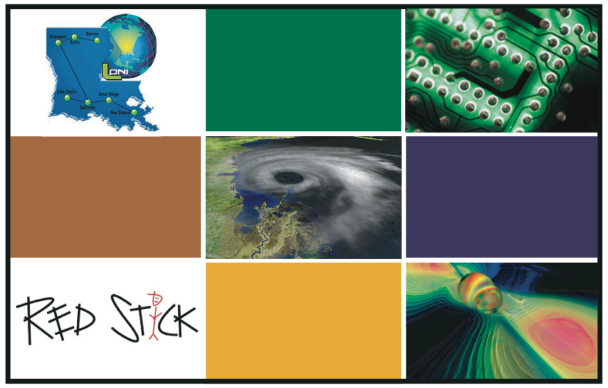
LSU
CENTER FOR COMPUTATION
& TECHNOLOGY

Example Input

```
initialCalc =
{
  Name -> "ML_ADM_Minkowski",
  Schedule -> {"IN ADMBase_InitialData"},
  ConditionalOnKeyword -> {"my_initial_data", "Minkowski"},
  Equations ->
  {
    g[la,lb] -> KD[la,lb],
    K[la,lb] -> 0,
    alpha    -> 1,
    beta[ua] -> 0
  }
}
```

$$
\begin{aligned}
g_{ab} &= \delta_{ab} \\
K_{ab} &= 0 \\
\alpha &= 1 \\
\beta^a &= 0
\end{aligned}
$$

Most simple routine in McLachlan:
Flat spacetime ADM initial data



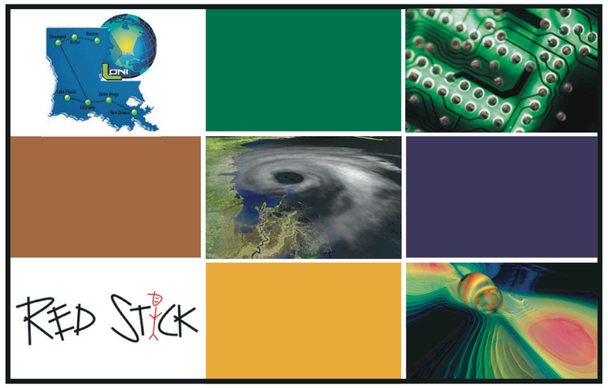
Example Output (C Code)

```
/* Loop over the grid points */
_Pragma ("omp parallel")
LC_LOOP3 (ML_ADM_Minkowski,
          i,j,k, min[0],min[1],min[2], max[0],max[1],max[2],
          cctk_lsh[0],cctk_lsh[1],cctk_lsh[2])
{
  index = CCTK_GFINDEX3D(cctkGH,i,j,k);

  /* Calculate temporaries and grid functions */
  g11L  =  1;
  g12L  =  0;
  g13L  =  0;
  g22L  =  1;
  g23L  =  0;
  g33L  =  1;
  K11L  =  0;
  K12L  =  0;
  K13L  =  0;
  K22L  =  0;
  K23L  =  0;
  K33L  =  0;
  alphaL  =  1;
  beta1L  =  0;
  beta2L  =  0;
  beta3L  =  0;
```

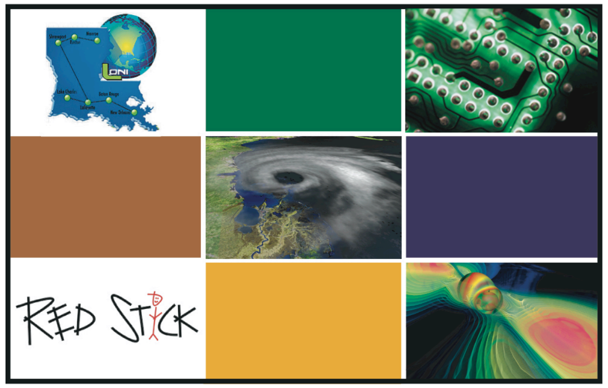
```
  /* Copy local copies back to grid functions */
  alpha[index] = alphaL;
  beta1[index] = beta1L;
  beta2[index] = beta2L;
  beta3[index] = beta3L;
  g11[index] = g11L;
  g12[index] = g12L;
  g13[index] = g13L;
  g22[index] = g22L;
  g23[index] = g23L;
  g33[index] = g33L;
  K11[index] = K11L;
  K12[index] = K12L;
  K13[index] = K13L;
  K22[index] = K22L;
  K23[index] = K23L;
  K33[index] = K33L;
}
LC_ENDLOOP3 (ML_ADM_Minkowski);
```

Tensor indices have been expanded, loop over grid points has been added. Not shown: finite difference macros, Cactus interface declarations.



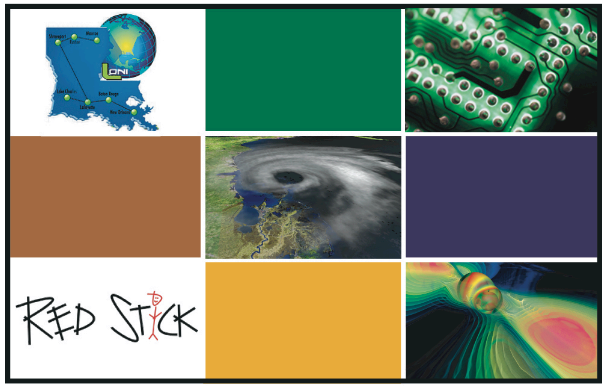
Original Kranc Design Considerations

- Equations should be input in a "natural manner", i.e., in a way in which people already write them (Mathematica!)
- It must be easy to modify the equations and generate new code – no manual tinkering
- The resulting code must be fast, on par with hand-written code



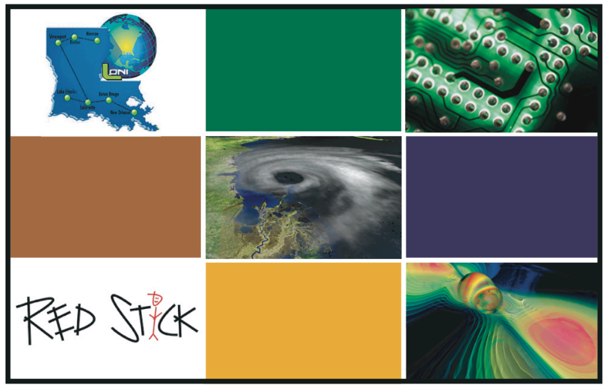
Current Kranc Design Considerations

- Well-posed formulations are now known (still many interesting mathematical aspects)
- As large simulations become common, proper experimental methods (“lab books”) have become important
- Surprisingly, HPC hardware architecture has begun to change dramatically



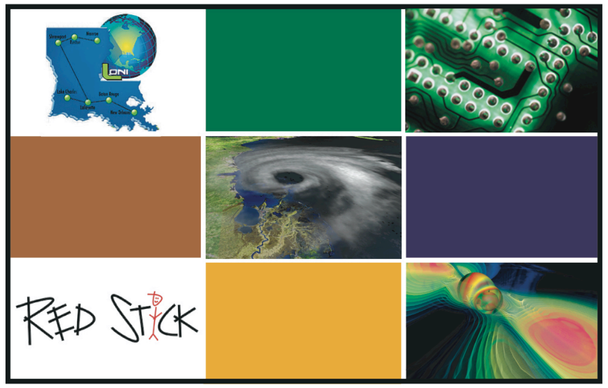
Provenance, Correctness, Believability

- Large simulation cannot be repeated; they are experiments, not calculations
- Can test continuum equations in Mathematica before discretisation
- Can add "annotators" to generated code
- Automatically generated code is more likely to be correct



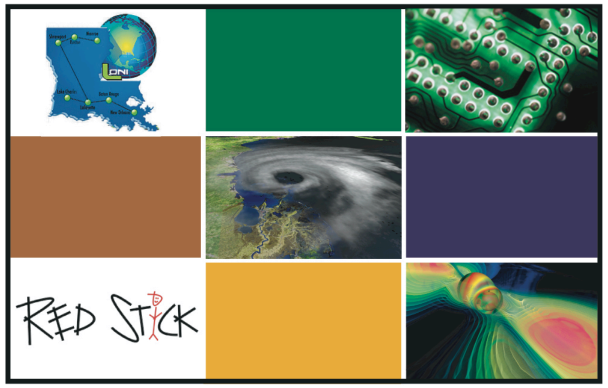
Higher Performance

- Fact: These days, compiled code is faster than assembler
- Surmise: These days, automatically generated code is faster than hand-written code
- Code can be automatically restructured, transformed, adapted to new programming models, far better than the average programmer would



Debugging, Profiling

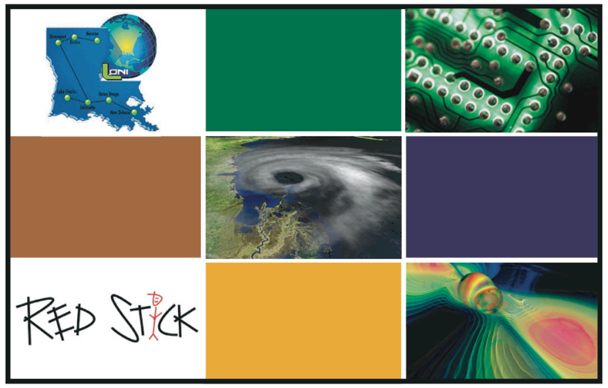
- Aspect oriented programming: Modify certain patterns (not just certain places)
- For example, "Check each assignment statement for NaN"
- Can insert annotations describing the code (performance statistics)
- Can give hints to run-time system



LSU
CENTER FOR COMPUTATION
& TECHNOLOGY

Summary

- Kranc generates code, directly from Mathematica to supercomputers
- Beside the obvious (ease of programming, correctness):
    - high level code can be more efficient
    - high level code is future proof
- Everybody in the community uses some kind of code generator



Further Information

- Kranc on the web: http://numrel.aei.mpg.de/Research/Kranc/
- Development tree: http://www.aei.mpg.de/~ianhin/kranc.git
- XiRel, Cyberinfrastructure for Numerical Relativity: http://www.cct.lsu.edu/xirel/