

Adaptable Geophysics Models and Solvers

Bill Appelbe, Steve Quenette, Patrick Sunter
Victorian Partnership for Advanced Computing, Melbourne, Australia

Louis Moresi
Monash University, Melbourne, Australia

Abstract

One key problem that faces geophysics is the high cost of constructing, adapting, and maintaining numerical models for large scale simulations. Tools such as Matlab or Mathematica can be used to rapidly model and numerically solve PDEs, but they are not efficient for large scale parallel computations. Libraries such as PETSc and NAG provide some adaptability in parallel implicit solvers, but adapting and testing different solvers is still a tedious process that requires recompilation and recoding much of the application. Our work has focused on building a lightweight software framework on top of libraries such as PETSc, NAG, and HYPRE, that supports development of new models with minimal recoding or loss of performance. The framework supports both implicit and explicit solvers, and is in use for DNS in geophysics at several sites in the USA and Australia for 2D and 3D CitCom and FLAC style solvers for visco-elastic flow.

Traditional Code Adaptation

Traditional scientific software is adapted to support new models by one of two means:

1. Cut and paste
Create a new code that is a copy of the old code with changes such as new solvers or rheology.
2. Modification
Modify the existing code to incorporate new features (solvers or geophysics). Where an extension is inconsistent with the existing code, use conditional compilation or conditional statements to choose the appropriate feature set.

Neither of these approaches scale well. Cut and paste can soon create dozens of incompatible versions of a program - any change or improvement in one version soon gets out of sync with the other versions of the program. Soon users are faced with a major maintenance problem: different versions of the code are supported on different platforms with different quirks and features. By contrast, modification tries to maintain one monolithic version of the application. But such monolithic code soon becomes harder and harder to understand and maintain, as the structure of the code is obscured by a thicket of interdependent conditional statements and compilation directives.

In modern systems and business programming, adaptation is provided through OO technology in Java and C++ and frameworks and plug-ins. A framework is a set of extensible library classes and interface that can be specialized for a particular problem without altering the underlying. A plug-in is an extension that can be added to the framework without recompilation of the framework.

Neither Java or C++ and their brand of OO technology have made significant inroads into scientific computing. The reason is simply that neither language is very efficient for large-scale parallel numerical models and frameworks.

The solution to this conundrum is to apply OO style technology (frameworks and plug-ins), but use traditional high-performance C and Fortran programming techniques, make parallelism implicit (in the framework) and build on top of existing numerical libraries and solvers such as PETSc, NAG, and HYPRE.

The StGermain framework has been developed and evolved over the past three years so that it supports a range of numerical models for lithospheric and mantle flow, using both implicit and explicit solvers and coupled models. The key abstractions in the framework include support for: grids, meshes, particles, and particle ensembles.

StGermain programs are built around the idea of a Context. A Context has two main roles:

- bringing together all the data structures a program requires, and
- controlling what is run where through a set of EntryPoints.

An EntryPoint is simply a modifiable stack of named function pointers, that can be run on command. An example entry point is outputting visualization information at the end of every solver time step. The combined group of EntryPoints a program defines forms an API (Applications Programming Interface) that the plugin developer can use to build new numerical models.

StGermain Contexts have three distinct stages:

1. setting up what the application will do, by filling in the entry points with function pointer hooks;
2. allocating memory for data structures and initializing them;
3. doing the real scientific work, by iteratively running the solver.

So how can plugins change how a program runs? The plugin dynamic libraries are loaded at the start of stage 2 above (in Context_Build()), so they can modify the entry points and extend the data structures before the actual main program (solver) runs. Thus, a basic output plugin may simply register a function pointer in the AbstractContext_Dump entry point. A more complex plugin which adds the ability to model material with a particular rheology may extend the particle type with additional information, and modify several key entry points within the solver. Complete flexibility is provided.

The current set of plug-ins include:

Materials: rectangular box initial conditions, slab initial conditions (for subduction)

Solvers: gravity, viscous rheology (the strong temperature dependence of viscosity in mantle convection for Newtonian and non-Newtonian fluids non-dimensional Arrhenius viscosity law, or Frank-Kamenetskii approximation).

Output: boundary layer thickness, iso-surfaces, temperature profiles, surface mobility

Plug-ins also support changing the underlying solver for implicit methods. A key performance issue is often the choice of linear solver or preconditioner. The choice of linear solver or preconditioner should almost invariably not be hard-wired into an application, as any change in the boundary conditions or equations will affect the choice of solver. Libraries like PETSc support dynamic choice of solver, and we have extended that to allow link-time binding of solver libraries including PETSc and HYPRE (Algebraic Multi-grid).

The code for StGermain is all open-source, and can be downloaded, along with documentation and demonstration examples from: <http://rd01.vpac.org/twiki/bin/view/Stgermain>

Presentation

The presentation will discuss the architecture and performance of StGermain for a variety of geophysics applications, and compare it to traditional monolithic applications such as Terra and CitCom.