Correct Finite Difference Solutions to Variable Viscosity Stokes Flow with Sharp Interfaces

Walter Landry
Computational Infrastructure for Geodynamics
walter@geodynamics.org

NSF Award
EAR-0949446

Introduction

Computational Infrastructure for Geodynamics (CIG) is an NSF funded, community governed center that develops, maintains, and distributes high performance parallel software for geophysics. In response to requests from the community, CIG is developing Gamr, a new Adaptive Mesh Refinement (AMR) code for Tectonics and Mantle Convection.

The idea of AMR is to have multiple grids with different resolutions. There is a coarse grid which covers the whole simulation, and then there are finer grids which only cover select regions as in Figure 1.

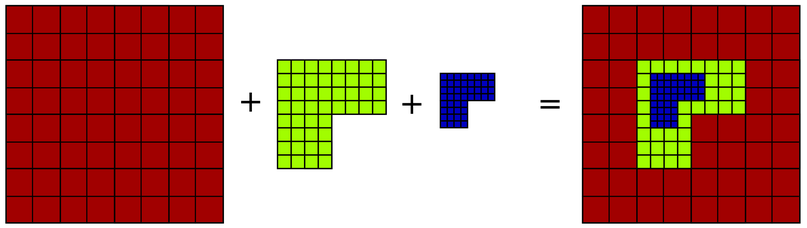

Figure 1: Multiple grids in an AMR simulation.

AMR algorithms generally create finer grids where the error is large. Since the error generally decreases with resolution, this will reduce the global error while only needing to apply effort in small regions.

Solid Earth Modeling and AMR

In the current version, Gamr models solid earth flow by using the finite difference method to solve the Stokes equations.

$$\tau_{ij,j} + p_{,i} = f_i$$

$$\nabla \cdot v = 0$$

Albers (2000) previously found solutions to variable viscosity Stokes flow using finite difference AMR on a single processor. We use the SAMRAI AMR library to enable 2D and 3D finite difference AMR solutions in parallel. Albers noted that interpolating boundary conditions from the coarse level to the fine level requires some care. Unless the boundary conditions are interpolated with quadratic stencils, the error in the finer grids will end up being no better than if we had just used a coarse grid.

Albers applied Dirichlet boundary conditions on the ingoing/outgoing velocity fields on the fine grids. This can be problematic, since compatibilty with the continuity equation is not guaranteed. Instead we apply stress conditions to the ingoing/outgoing velocities and Dirichlet conditions to the transverse velocities.

Solver

To actually solve the Stokes flow, we use multigrid with the smoother developed by Tackley (2008). This gives us good answers to some standard problems such as a circular inclusion under shear.

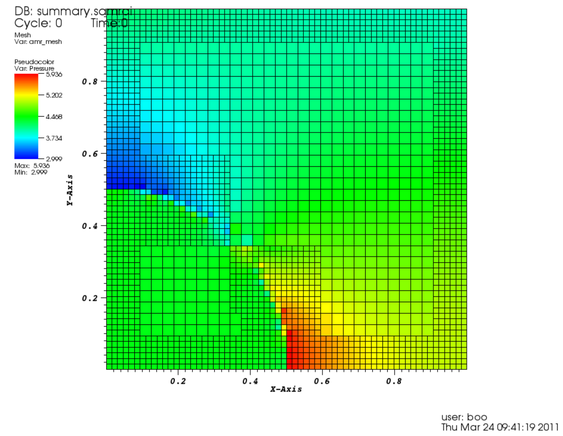

Figure 2: Pressure for circular inclusion under pure shear

Unfortunately, in the presence of sharp viscosity interfaces, such as between a subducting slab and the mantle, traditional finite difference methods result in an error that increases as the grid becomes finer. So the AMR algorithm will try to create an infinitely fine grid around the interface, resulting in an infinitely large error. Figure 3 shows the pressure solution for a sinking block in 2D and 3D.

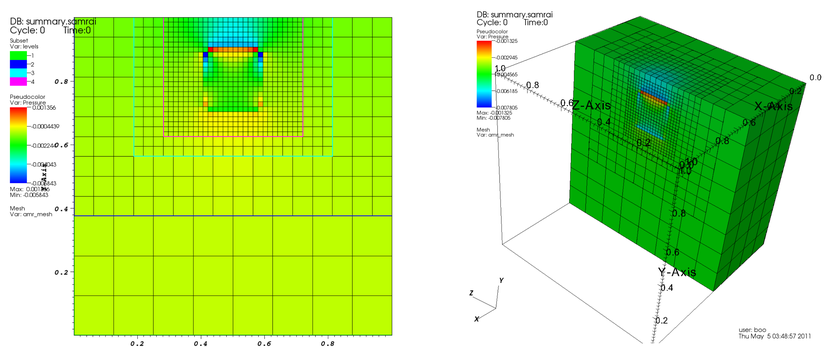

Figure 2: Pressure solutions for high viscosity blocks sinking in a low viscosity medium

geodynamics.org

Immersed Interface Method

We will use the Immersed Interface Method (IIM) to obtain convergent results. Other methods, such as Ghost Fluids and Immersed/Embedded Boundaries ignore errors in some terms in favor of simplicity of implementation. There are many variants of IIM (EJIIM, DIIM, CIM, FIIM, MIIM), but they all work by modifying numerical derivatives by subtracting out spurious terms that come from a sharp interface.

Most applications of IIM to Stokes assume constant viscosity. In order to apply the proper conditions at the interface, we must use, as a fundamental quantity, the velocity scaled by the viscosity

$$u = \eta v$$

Using this to write the Stokes Equations gives

$$-\frac{\partial}{\partial x^j}\left(u_{i,j} + u_{j,i} + u_i \frac{\partial \log \eta}{\partial x^j} + u_j \frac{\partial \log \eta}{\partial x^i}\right) + p_{,i} = f_i$$

$$u_{i,i} - u_i \frac{\partial \log \eta}{\partial x^i} = 0$$

The viscosity only appears explicitly inside a logarithm. Some normally difficult test problems where the viscosity rises exponentially with height thus become relatively trivial. Also, u is better behaved numerically than the velocity, since where the viscosity is large, the velocity is small, and vice versa.

Given this, we can write, for example, the jump conditions for u and its derivative tangential to the interface

$$[u] = [\eta]v \qquad [u_{,\tau}] = [\eta]v_{,\tau}$$

where brackets [] denote the jump across the interface. We can then write a correction to a finite difference derivative as

$$\delta(u_{,x}) = ([u] + [u_{,x}]\delta x + \tfrac{1}{2}[u_{,xx}]\delta x^2)/h$$

where h is the grid spacing and δx is the distance to the interface of point on the "other" side of the interface.

The algorithm then becomes, start with an initial guess for the velocity v. Use this to compute jump conditions and a new solution. Use the new solution to get new jump conditions and iterate.