

Parallel solution of non-linear PDEs with PETSc's Scalable Nonlinear Equations Solvers (SNES)

Richard F. Katz

Myths and Methods in Modeling

November 14, 2004

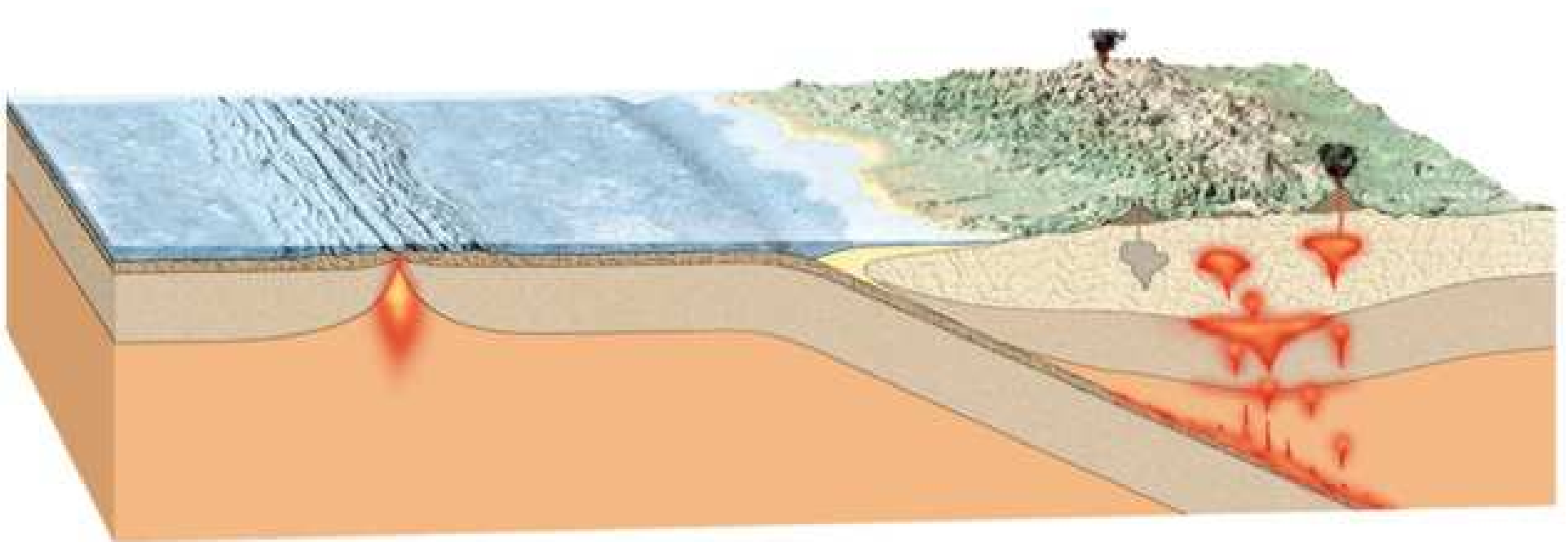
(slides available on the web:

www.ideo.columbia.edu/teaching/snes_talk.pdf)

Basic elements

1. Selecting or deriving a set of partial differential equation to describe the physics relevant to the problem. Start simple!!!
2. Choosing a domain and discretization for the problem.
3. Understanding and using a Newton-Krylov method.
4. Developing code in the PETSc framework.
5. The parameter continuation method.
6. Parallelization.

An example: thermal structure of a subduction zone



1. Select PDEs

- Continuity equation for an incompressible fluid:

$$\nabla \cdot \mathbf{v} = 0$$

- Navier-Stokes, $\mathbf{v}_t + \mathbf{v} \cdot \nabla \mathbf{v} = \nabla \cdot (\eta \dot{\epsilon}) - \nabla P$ in the limit of infinite Reynolds number:

$$\nabla P - \nabla \cdot [\eta (\nabla \mathbf{v} + \nabla \mathbf{v}^T)] = 0$$

where viscosity is a **non-linear** function $\eta = \eta(z, T, \mathbf{v})$

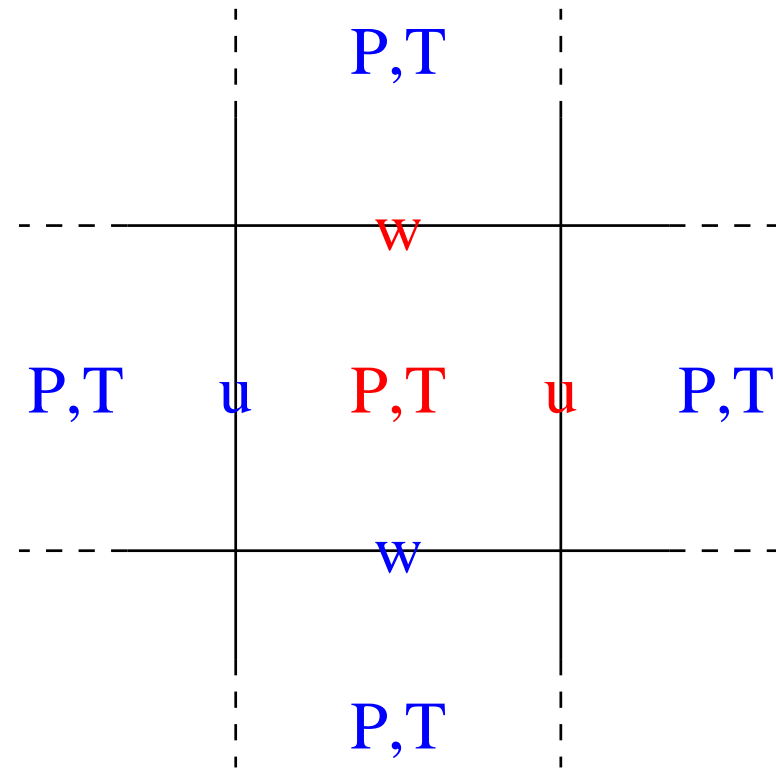
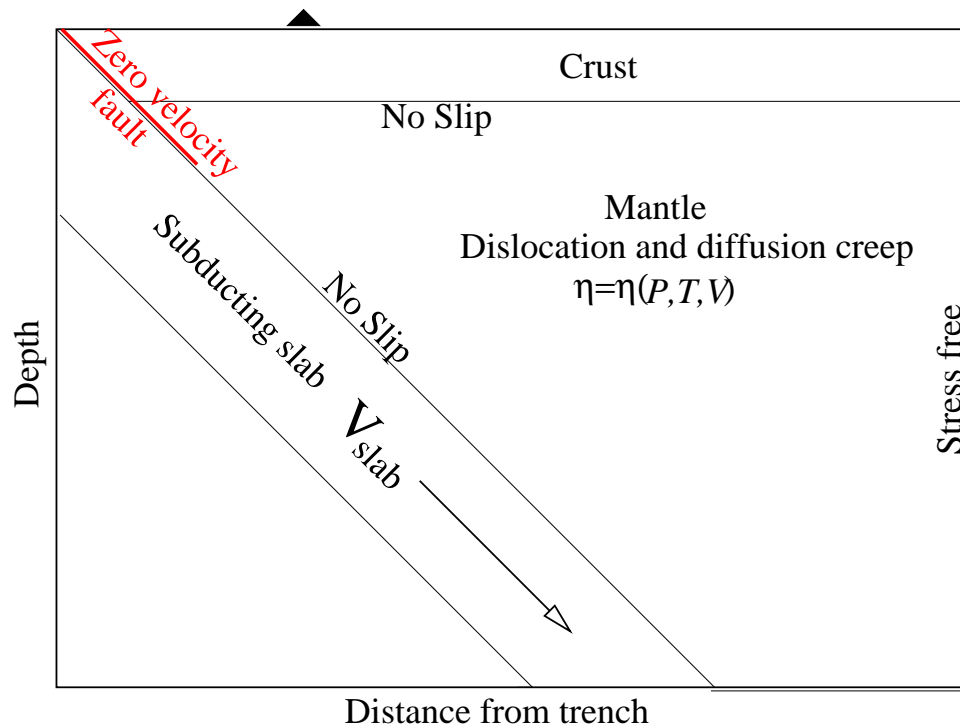
- Advection-diffusion of temperature:

$$\mathbf{v} \cdot \nabla T - \frac{1}{Pe} \nabla^2 T = 0$$

- Keep it simple: no buoyancy forces, no frictional heating, etc.

2. Domain and Discretization

- Define sub-regions of domain & boundary conditions.
- Finite-volume on a staggered grid.



2 cont'd. Discretization

- For each grid cell, formulate the finite-difference/volume approximations of the PDEs. The continuity equation for example:

$$\nabla \cdot \mathbf{v} = 0 \rightarrow \frac{(u_{ij} - u_{i-1j})}{dx} + \frac{(w_{ij} - w_{ij-1})}{dz} = 0$$

- A set of **linear** equations for an aggregate vector of variables x would take the form $Ax = b$. We could use any linear solver to solve for x .
- Viscosity is **non-linear** and thus we cannot express the system as a set of linear equations. Instead we can write $A(x) = 0$.

2 cont'd. Discretization

- We cannot solve $A(x) = 0$ directly by inverting A .
- Must use an iterative method:
for $k = 1, 2, \dots$

$$r_k = A(x_k)$$

$$x_{k+1} = x_k + f(r_k)$$

- Choose f such that $r_{k+1} < r_k$.

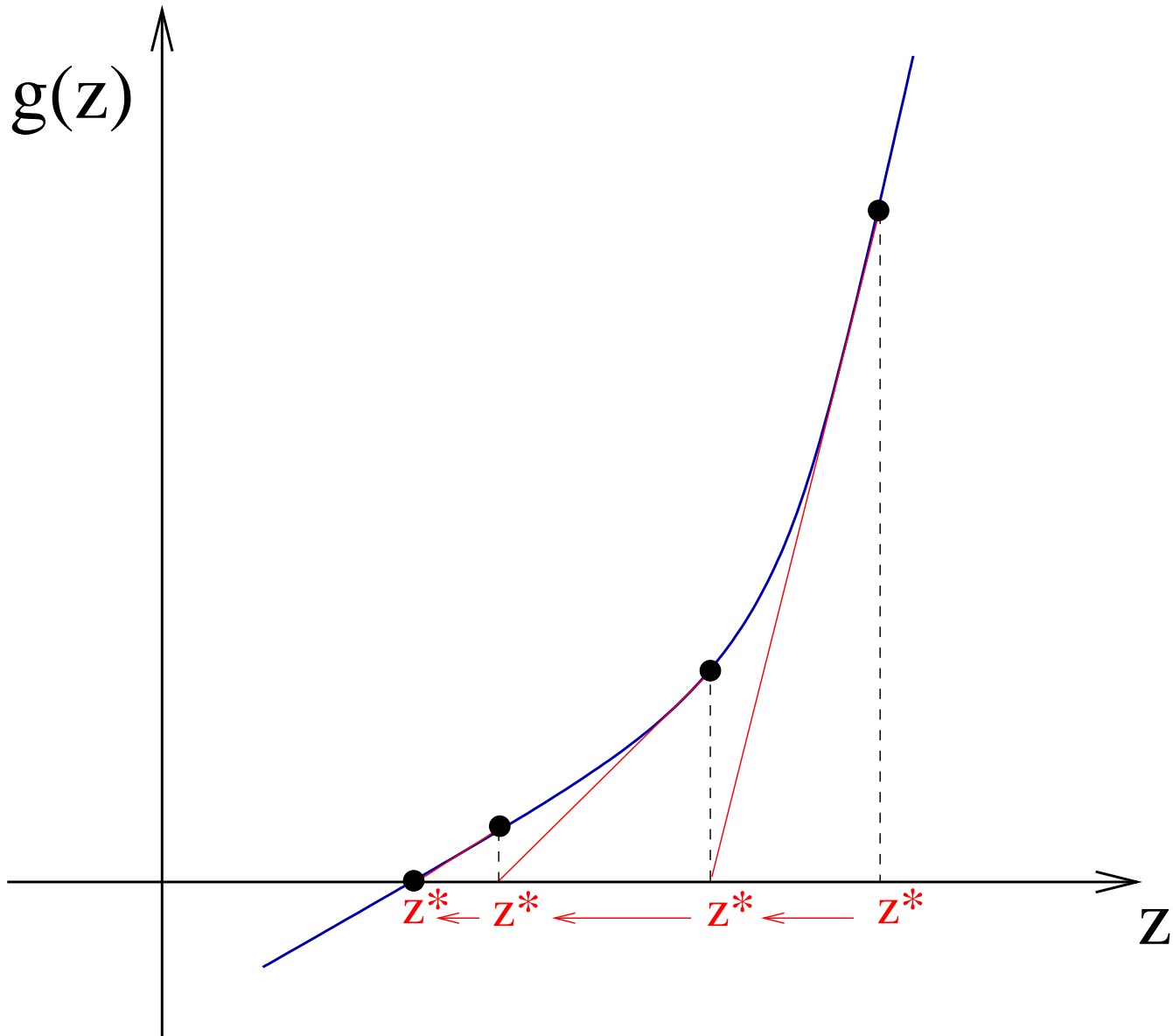
3. Newton's method (aka Newton-Raphson) in 1D

- Suppose that we wish to solve $g(z) = 0$ but g has no *analytic* roots.
- Assume that g has a solution at z_0 . That is $g(z_0) = 0$.
- Suppose that we have a good guess z^* for z_0 but we are off by some amount δz . In other words, $z^* = z_0 + \delta z$.
- Expand $g(z)$ in a Taylor series around the solution z_0 :

$$g(z^*) = g(z_0 + \delta z) = g(z_0) + \frac{dg}{dz}(z_0)\delta z + \dots$$

- $g(z_0) = 0$ and so $\delta z \approx g(z^*) / \frac{dg}{dz}$ and we can improve our guess
 $z^* \leftarrow z^* - \delta z$

Newton's method in 1D



Newton's method in ND

- Formally the same as Newton's method in 1D.
- Define the residual as $r = A(x)$.
- Find solution \tilde{x} such that $\|A(\tilde{x})\| < tol$.
- for $k = 0, 1, 2, \dots$

Evaluate residual $r^k = A(x^k),$

Check for convergence $\|r^k\| < tol,$

Form Jacobian $J_{ij}^k = \frac{\partial r_i^k}{\partial x_j^k},$

Linear solve for correction $J^k \delta x = r^k,$

Correct guess $x^{(k+1)} = x^k - \delta x.$

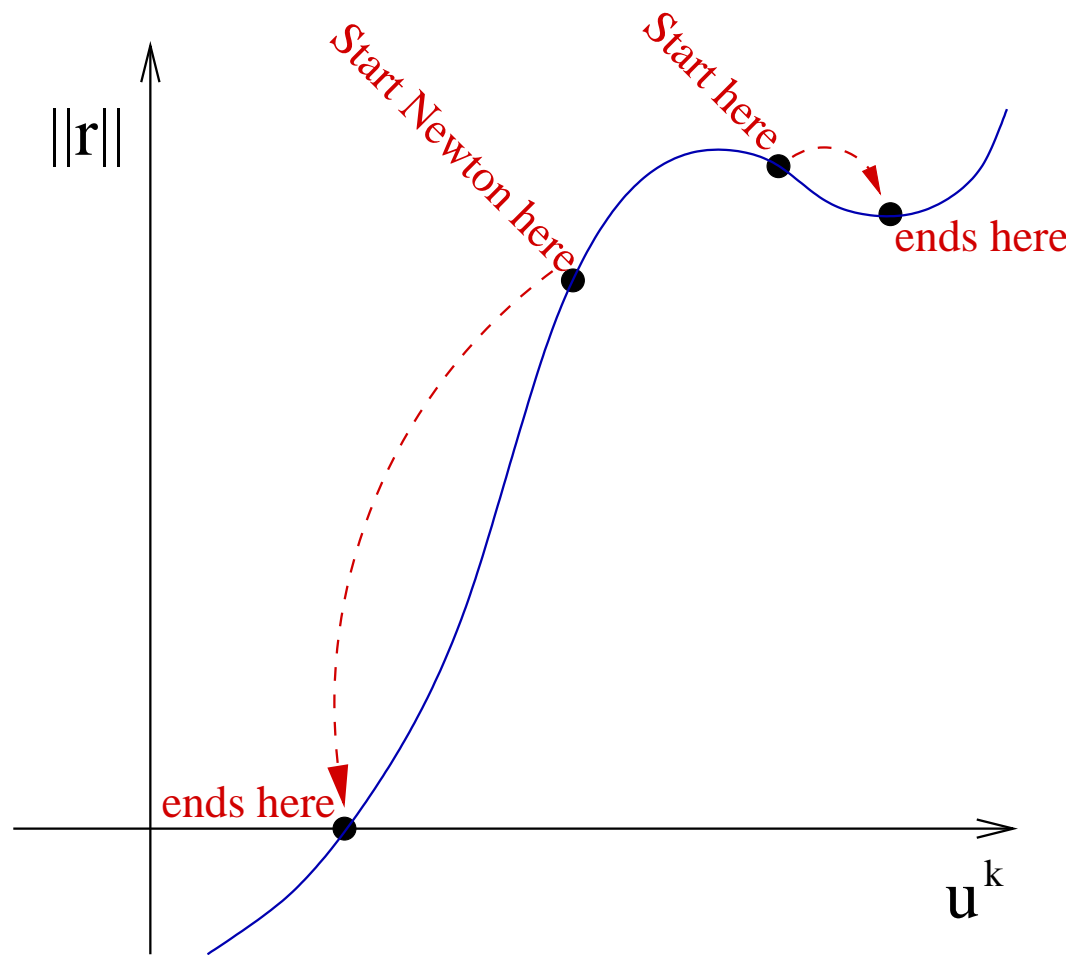
- Use a Krylov method (such as GMRES) to find δx .

Newton-Krylov method

- Most often used approach to solving sets of non-linear equations.
- Basic structure:

```
Initialize solution vector with a guess;  
While (norm of solution > tolerance) {  
    Compute jacobian of residual;  
    Set up linear system for correction  
        to solution;  
    Precondition linear system;  
    Solve linear system with Krylov method;  
    Apply correction to solution;  
    Compute norm of solution;  
}
```

Importance of a good guess to Newton



Newton can only find a root if your initial guess, u^0 is sufficiently close.

4. Developing code in the PETSc framework

- Each node on the grid contains set of field variables:

```
typedef struct {  
    PetscScalar u,w,p,T;  
} Field;
```

- For regular rectangular meshes, allocate a **distributed array**.

```
DACreate2d(comm,grid.periodic,grid.stencil,grid.ni,grid.nj,  
           PETSC_DECIDE,PETSC_DECIDE,grid.dof,grid.stencil_width,  
           0,0,&da);
```

- Provide call-back functions to initialize the solution vector and evaluate the residual $A(u^k)$.

```
DMMGSetSNESLocal(dmmg,FormFunctionLocal,0,0,0);  
DMMGSetInitialGuess(dmmg,FormInitialGuess);
```

- You can select a preconditioner and Krylov method on the command line (or use the defaults).

Basics of the FormFunction

```
FormFunctionLocal(DALocalInfo *info,Field **x,  
                 Field **r,void *ptr)  
{  
    ...  
    for (j=js; j<je; j++) {  
        for (i=is; i<ie; i++) {  
            r[j][i].u = XMomentumResidual(x,i,j,user);  
            r[j][i].w = ZMomentumResidual(x,i,j,user);  
            r[j][i].p = ContinuityResidual(x,i,j,user);  
            r[j][i].T = EnergyResidual(x,i,j,user);  
        }  
    }  
    ... return;  
}
```

An example residual calculation

```
PetscScalar ContinuityResidual(Field **x, int i,  
                                int j, float dx, float dz)  
{  
    dudx = (x[j][i].u - x[j][i-1].u)/dx;  
    dwdz = (x[j][i].w - x[j-1][i].w)/dz;  
    return dudx + dwdz;  
}
```

5. The continuation method

- Degree of non-linearity may control the convergence behavior of Newton or the conditioning of the Jacobian.
- Possible to “creep up” on the solution by adjusting a parameter that controls the non-linearity.

- For example:

$$\nabla P - \nabla \cdot \left[\alpha \sqrt{\eta} (\nabla \mathbf{v} + \nabla \mathbf{v}^T) \right] = 0; \quad \eta, \alpha \in (0, 1]$$

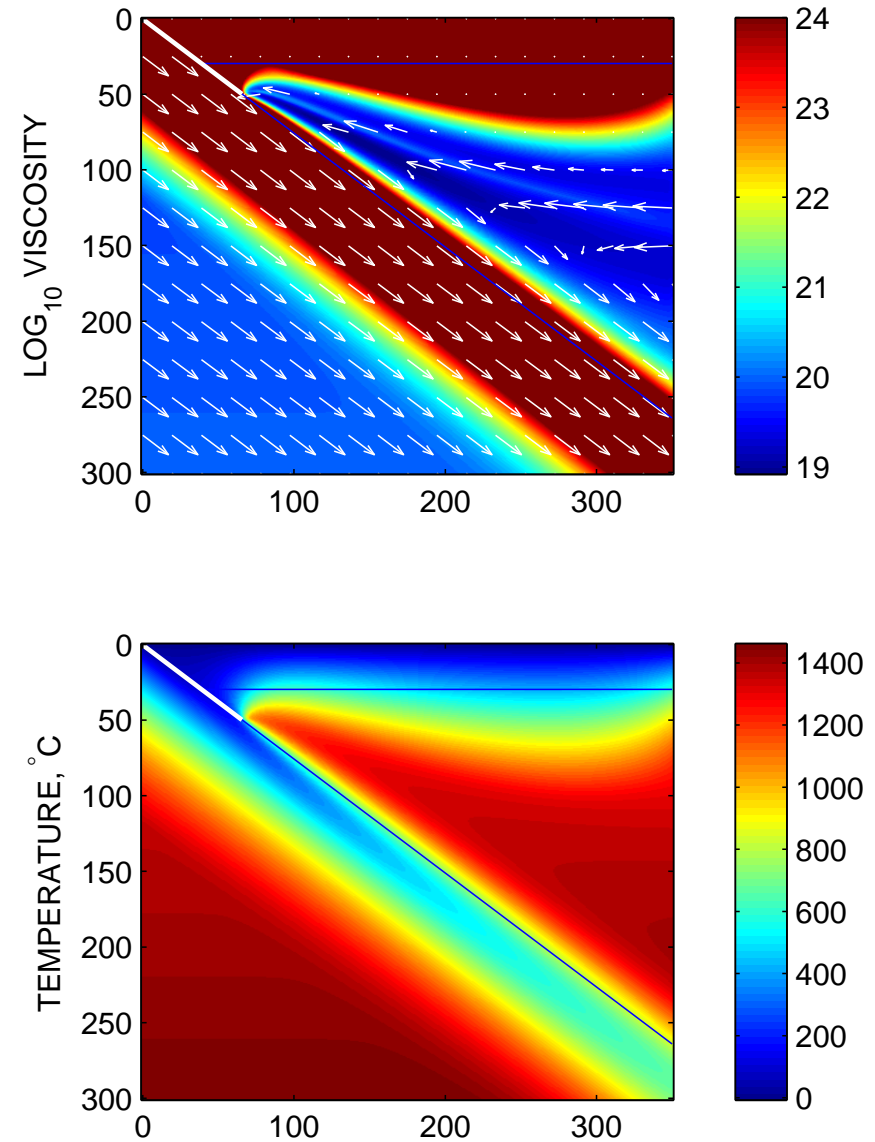
- $\alpha = 1$ gives back the correct equation but is hard to solve because η is a strongly non-linear function.
- Instead solve for $\alpha = 0.3, 0.4, 0.5, \dots, 1.0$ using old solution as an initial guess for new solution.

3 cont'd. The continuation method

```
for (param->continuation=0.20;
     param->continuation<=1.0; ) {

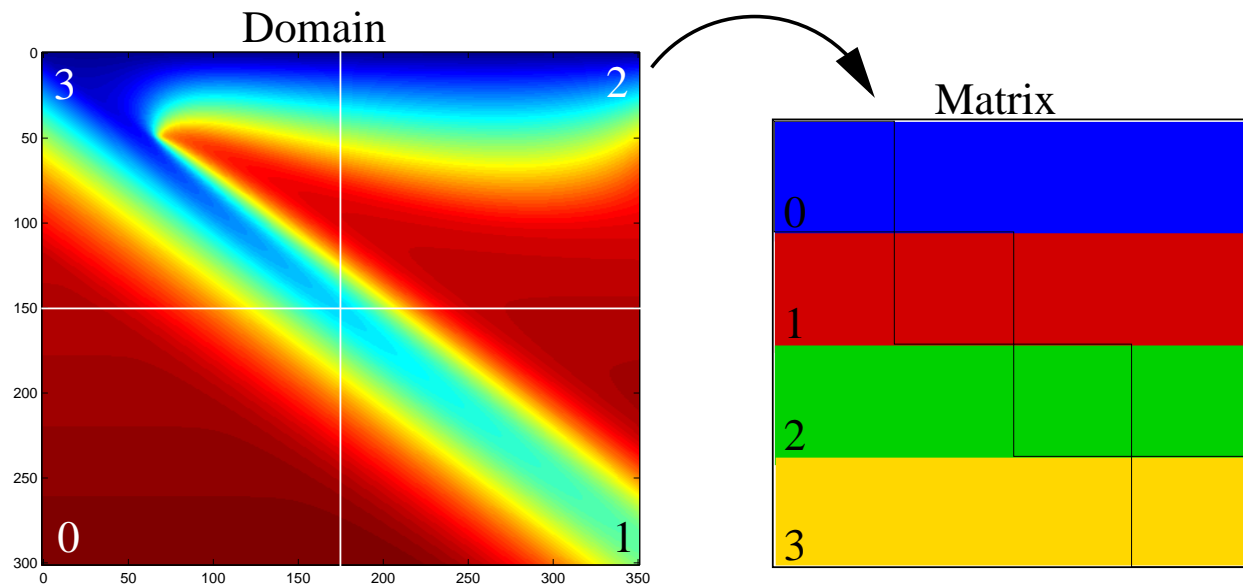
  /* solve the non-linear system */
  DMMGSolve(dmmg);
  VecCopy(DMMGGetx(dmmg),user->Xguess);
  SNESGetIterationNumber(snes,&its);
  if (param->continuation==1.0) goto done;

  /* update continuation parameter */
  if (its<=3) {
    param->continuation += 0.125;
  } else if (its<=8) {
    param->continuation += 0.10;
  } else if (its<=20) {
    param->continuation += 0.075;
  } else {
    param->continuation += 0.05;
  }
}
```



Parallelization

- In PETSc, Krylov methods and Newton's method are already parallel—you don't have to do anything!*
- Parallel preconditioner must be used.
- (Common choice) Schwartz method of domain decomposition: each subdomain (approximately) inverts its piece of A .



- A^{-1} is constructed by putting together the A_i^{-1} .