Computational Considerations for Magma Dynamics

Matthew Knepley and Tobin Isaac

Computational and Applied Mathematics
Rice University

Computation Institute
University of Chicago

Melt in the Mantle Workshop
Isaac Newton Centre
Cambridge University, UK   February 19, 2016

Collaborators

Magma Dynamics Collaboration

Richard Katz

Timestepping Collaboration

Jed Brown

Collaborators

Adaptive Mesh Refinement Collaboration

Toby Isaac

Main Point

What am I going to talk about?

- Nonlinear Solvers for the Mechanical Model
- Discretization of the Mechanical Model
- Meshing for the Mechanical Model

Main Point

What am I going to talk about?

- Nonlinear Solvers for the Mechanical Model
- Discretization of the Mechanical Model
- ~~Meshing for the Mechanical Model~~

Why is this important?

- Comparison is essential for making informed algorithmic choices
- Comparison in a single code seems necessary
- Current codes make it difficult to compare meshes, discretizations, and multilevel solvers

Outline

Main Points

We can construct nonlinear solvers that are more robust than Newton.

We can construct nonlinear solvers that are more rapidly convergent than Newton.

Picard is never really acceptable.

Main Points

We can construct nonlinear solvers that are more robust than Newton.

We can construct nonlinear solvers that are more rapidly convergent than Newton.

Picard is never really acceptable.

Main Points

We can construct nonlinear solvers that are more robust than Newton.

We can construct nonlinear solvers that are more rapidly convergent than Newton.

Picard is never really acceptable.

Outline

Dimensional Formulation

$$\nabla p - \nabla \zeta_\phi (\nabla \cdot \vec{v}^S) - \nabla \cdot \left(2 \eta_\phi \dot{\epsilon}^S \right) = 0$$

$$\nabla \cdot \left(-\frac{K_\phi}{\mu} \nabla p + \vec{v}^S \right) = 0$$

$$\frac{\partial \phi}{\partial t} - \nabla \cdot (1 - \phi) \vec{v}^S = 0$$

Closure Conditions

$$K_\phi = K_0 \left(\frac{\phi}{\phi_0}\right)^n$$

$$\eta_\phi = \eta_0 \exp\left(-\lambda(\phi - \phi_0)\right)$$

$$\zeta_\phi = \zeta_0 \left(\frac{\phi}{\phi_0}\right)^{-m}$$

Nondimensional Formulation

$$\nabla p - \nabla \left(\left(\frac{\phi}{\phi_0} \right)^{-m} \nabla \cdot \vec{v}^S \right) - \nabla \cdot \left(2e^{-\lambda(\phi - \phi_0)} \dot{\epsilon}^S \right) = 0$$

$$\nabla \cdot \left(-\frac{R^2}{r_\zeta + 4/3} \left(\frac{\phi}{\phi_0} \right)^n \nabla p + \vec{v}^S \right) = 0$$

$$\frac{\partial \phi}{\partial t} - \nabla \cdot (1 - \phi) \vec{v}^S = 0$$

Initial and Boundary conditions

Initially

$$\phi = \phi_0 + A\cos(\vec{k}\cdot\vec{x})$$

where

$$A \ll \phi_0$$

and on the top and bottom boundary

$$K_\phi \nabla p \cdot \hat{n} = 0$$

$$\vec{v}^S = \pm\frac{\dot{\gamma}}{2}\hat{x}$$

Mechanical Benchmarks

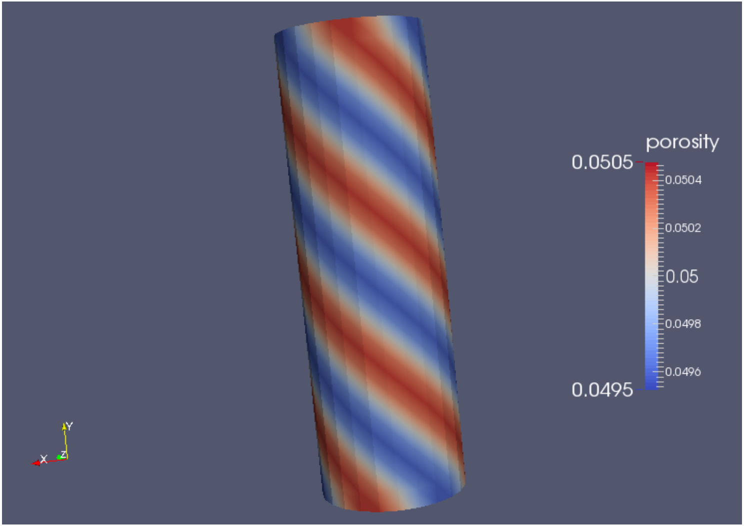
Benchmark 0: $\lambda = 0$
There is no porosity feedback, and the initial pattern is stably advected:

$$\vec{k}(t) = \vec{k}_0 \left(\hat{\vec{x}} \sin\theta_0 + \hat{\vec{y}} \left(\cos\theta_0 - t \sin\theta_0 \right) \right)$$

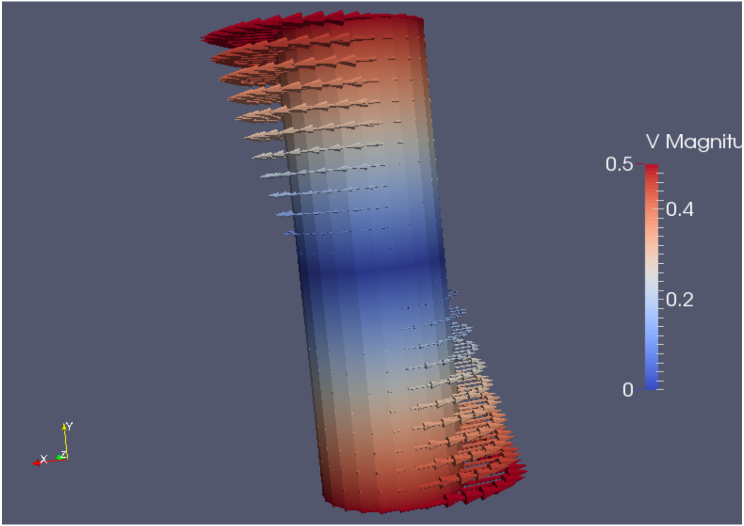
Benchmark 1: $\lambda > 0$
The porosity feedback causes localization, with initial compaction rate:

$$\mathcal{C} = \nabla \cdot \vec{v}_S|_{t=0} = \frac{A \lambda \phi_0 \sin(2\theta_0)}{r_\zeta + 4/3} \cos(\vec{k} \cdot \vec{x})$$

Initial Porosity



Initial Velocity



Outline

Solver Organization

Timestepping

We will use simple Backward Euler:

$$\nabla p^{k+1} - \nabla \left(\left(\frac{\phi^{k+1}}{\phi_0} \right)^{-m} \nabla \cdot \vec{v}^{k+1} \right) - \nabla \cdot \left(2 e^{-\lambda(\phi^{k+1} - \phi_0)} \dot{\epsilon}^{k+1} \right) = 0$$

$$\nabla \cdot \left(-\frac{R^2}{r_\zeta + 4/3} \left(\frac{\phi^{k+1}}{\phi_0} \right)^n \nabla p^{k+1} + \vec{v}^{k+1} \right) = 0$$

$$\frac{\phi^{k+1} - \phi^k}{\Delta t} - \nabla \cdot (1 - \phi^{k+1}) \vec{v}^{k+1} = 0$$

Solver Organization
Newton-Krylov

Begin with a Newton-Krylov solve with line search:

$$\mathcal{N}\backslash \mathrm{K} -_L \mathrm{NRICH}$$

Optimal linear preconditioner in
Rhebergen, Wells, Wathen, and Katz, SISC.

Solver Organization
Newton-Krylov without Porosity

We can separate the Stokes-like solve from the porosity advection:

$$\left(\begin{array}{cc|c} \multicolumn{2}{c|}{A \oplus_{\mathrm{Schur}} L} & \begin{array}{c}0\\0\end{array} \\ \hline F & 0 & I + G \end{array}\right)$$

Solver Organization
Newton-Krylov

```
-pc_type fieldsplit
 -pc_fieldsplit_0_fields 0,1 -pc_fieldsplit_1_fields 2
 -pc_fieldsplit_type multiplicative
   -fieldsplit_0_pc_type fieldsplit
   -fieldsplit_0_pc_fieldsplit_type schur
   -fieldsplit_0_pc_fieldsplit_schur_precondition selfp
   -fieldsplit_0_pc_fieldsplit_schur_factorization_type full
   -fieldsplit_0_fieldsplit_velocity_pc_type lu
   -fieldsplit_0_fieldsplit_pressure_ksp_rtol 1.0e-9
   -fieldsplit_0_fieldsplit_pressure_pc_type gamg
    -fieldsplit_0_fieldsplit_pressure_ksp_monitor
    -fieldsplit_0_fieldsplit_pressure_ksp_gmres_restart 100
    -fieldsplit_fieldsplit_0_pressure_ksp_max_it 200
```

Solver Organization
Newton-Krylov with Porosity

Or we can incorporate the porosity advection into the Stokes-like solve:

$$\left[\begin{array}{c|c} \mathrm{A} & \mathrm{E} \\ \hline \mathrm{F} & \mathrm{I+G} \end{array}\right] \oplus_{\mathrm{Schur}} L$$

Newton options

Newton-Krylov with Porosity

```
-snes_monitor -snes_converged_reason
-snes_type newtonls -snes_linesearch_type bt
-snes_fd_color -snes_fd_color_use_mat -mat_coloring_type greedy
-ksp_rtol 1.0e-10 -ksp_monitor -ksp_gmres_restart 200
-pc_type fieldsplit
 -pc_fieldsplit_0_fields 0,2 -pc_fieldsplit_1_fields 1
 -pc_fieldsplit_type schur -pc_fieldsplit_schur_precondition selfp
  -pc_fieldsplit_schur_factorization_type full
   -fieldsplit_0_pc_type lu
   -fieldsplit_pressure_ksp_rtol 1.0e-9 -fieldsplit_pressure_pc_type gamg
    -fieldsplit_pressure_ksp_monitor
    -fieldsplit_pressure_ksp_gmres_restart 100
    -fieldsplit_pressure_ksp_max_it 200
```

Early Newton convergence

```
0 TS dt 0.01 time 0
    0 SNES Function norm 5.292194079127e-03
      Linear pressure_ solve converged due to CONVERGED_RTOL its 10
      0 KSP Residual norm 4.618093146920e+00
      Linear pressure_ solve converged due to CONVERGED_RTOL its 10
      1 KSP Residual norm 3.018153330707e-03
      Linear pressure_ solve converged due to CONVERGED_RTOL its 11
      2 KSP Residual norm 4.274869628519e-13
    Linear solve converged due to CONVERGED_RTOL its 2
    1 SNES Function norm 2.766906985362e-06
      Linear pressure_ solve converged due to CONVERGED_RTOL its 8
      0 KSP Residual norm 2.555890235972e-02
      Linear pressure_ solve converged due to CONVERGED_RTOL its 8
      1 KSP Residual norm 1.638293944976e-07
      Linear pressure_ solve converged due to CONVERGED_RTOL its 8
      2 KSP Residual norm 1.771928779400e-14
    Linear solve converged due to CONVERGED_RTOL its 2
    2 SNES Function norm 1.188754322734e-11
  Nonlinear solve converged due to CONVERGED_FNORM_RELATIVE its 2
1 TS dt 0.01 time 0.01
```

Later Newton convergence

```
0 TS dt 0.01 time 0.63
    0 SNES Function norm 9.366565251786e-03
      Linear pressure_ solve converged due to CONVERGED_RTOL its 16
      Linear pressure_ solve converged due to CONVERGED_RTOL its 16
      Linear pressure_ solve converged due to CONVERGED_RTOL its 16
    Linear solve converged due to CONVERGED_RTOL its 2
    1 SNES Function norm 4.492625910272e-03
    Linear solve converged due to CONVERGED_RTOL its 2
    2 SNES Function norm 3.666181450068e-03
    Linear solve converged due to CONVERGED_RTOL its 2
    3 SNES Function norm 2.523116582272e-03
    Linear solve converged due to CONVERGED_RTOL its 2
    4 SNES Function norm 3.022638159491e-04
    Linear solve converged due to CONVERGED_RTOL its 2
    5 SNES Function norm 9.761317324448e-06
    Linear solve converged due to CONVERGED_RTOL its 2
    6 SNES Function norm 1.147944474432e-08
    Linear solve converged due to CONVERGED_RTOL its 2
    7 SNES Function norm 8.729160299009e-14
  Nonlinear solve converged due to CONVERGED_FNORM_RELATIVE its 7
1 TS dt 0.01 time 0.64
```

Newton failure

```
0 TS dt 0.01 time 0.64
Time 0.64 L_2 Error: 0.494811 [0.0413666, 0.491642, 0.0376071]
    0 SNES Function norm 9.682733054059e-03
    Linear solve converged due to CONVERGED_RTOL iterations 2
    1 SNES Function norm 6.841434267123e-03
    Linear solve converged due to CONVERGED_RTOL iterations 3
    2 SNES Function norm 4.412420553822e-03
    Linear solve converged due to CONVERGED_RTOL iterations 5
    3 SNES Function norm 3.309326919835e-03
    Linear solve converged due to CONVERGED_RTOL iterations 6
    4 SNES Function norm 3.022494350289e-03
    Linear solve converged due to CONVERGED_RTOL iterations 7
    5 SNES Function norm 2.941050948582e-03
    Linear solve converged due to CONVERGED_RTOL iterations 7
   ⋮
    9 SNES Function norm 2.631941422878e-03
    Linear solve converged due to CONVERGED_RTOL iterations 7
   10 SNES Function norm 2.631897334054e-03
    Linear solve converged due to CONVERGED_RTOL iterations 10
   11 SNES Function norm 2.631451174722e-03
    Linear solve converged due to CONVERGED_RTOL iterations 15
   ⋮
```

Outline

Abstract System

Out prototypical nonlinear equation is:

$$\mathcal{F}(\vec{x}) = \vec{b} \tag{1}$$

and we define the residual as

$$\vec{r}(\vec{x}) = \mathcal{F}(\vec{x}) - \vec{b} \tag{2}$$

Abstract System

Out prototypical nonlinear equation is:

$$\mathcal{F}(\vec{x}) = \vec{b} \tag{1}$$

and we define the (linear) residual as

$$\vec{r}(\vec{x}) = A\vec{x} - \vec{b} \tag{3}$$

Linear Left Preconditioning

The modified equation becomes

$$P^{-1}\left(A\vec{x} - \vec{b}\right) = 0 \tag{4}$$

Linear Left Preconditioning

The modified defect correction equation becomes

$$P^{-1}\left(A\vec{x}_i - \vec{b}\right) = \vec{x}_{i+1} - \vec{x}_i \tag{5}$$

Additive Combination

The linear iteration

$$\vec{x}_{i+1} = \vec{x}_i - (\alpha P^{-1} + \beta Q^{-1})(A\vec{x}_i - \vec{b}) \tag{6}$$

becomes the nonlinear iteration

Additive Combination

The linear iteration

$$\vec{x}_{i+1} = \vec{x}_i - (\alpha P^{-1} + \beta Q^{-1})\vec{r}_i \tag{7}$$

becomes the nonlinear iteration

Additive Combination

The linear iteration

$$\vec{x}_{i+1} = \vec{x}_i - (\alpha P^{-1} + \beta Q^{-1})\vec{r}_i \tag{7}$$

becomes the nonlinear iteration

$$\vec{x}_{i+1} = \vec{x}_i + \alpha(\mathcal{N}(\mathcal{F}, \vec{x}_i, \vec{b}) - \vec{x}_i) + \beta(\mathcal{M}(\mathcal{F}, \vec{x}_i, \vec{b}) - \vec{x}_i) \tag{8}$$

Nonlinear Left Preconditioning

From the additive combination, we have

$$P^{-1}\vec{r} \Longrightarrow \vec{x}_i - \mathcal{N}(\mathcal{F}, \vec{x}_i, \vec{b}) \tag{9}$$

so we define the preconditioning operation as

$$\vec{r}_L \equiv \vec{x} - \mathcal{N}(\mathcal{F}, \vec{x}, \vec{b}) \tag{10}$$

Multiplicative Combination

The linear iteration

$$\vec{x}_{i+1} = \vec{x}_i - (P^{-1} + Q^{-1} - Q^{-1}AP^{-1})\vec{r}_i \qquad (11)$$

becomes the nonlinear iteration

Multiplicative Combination

The linear iteration

$$\vec{x}_{i+1/2} = \vec{x}_i - P^{-1}\vec{r}_i \tag{12}$$
$$\vec{x}_i = \vec{x}_{i+1/2} - Q^{-1}\vec{r}_{i+1/2} \tag{13}$$

becomes the nonlinear iteration

Multiplicative Combination

The linear iteration

$$\vec{x}_{i+1/2} = \vec{x}_i - P^{-1}\vec{r}_i \tag{12}$$

$$\vec{x}_i = \vec{x}_{i+1/2} - Q^{-1}\vec{r}_{i+1/2} \tag{13}$$

becomes the nonlinear iteration

$$\vec{x}_{i+1} = \mathcal{M}(\mathcal{F}, \mathcal{N}(\mathcal{F}, \vec{x}_i, \vec{b}), \vec{b}) \tag{14}$$

Nonlinear Right Preconditioning

For the linear case, we have

$$AP^{-1}\vec{y} = \vec{b} \tag{15}$$

$$\vec{x} = P^{-1}\vec{y} \tag{16}$$

so we define the preconditioning operation as

$$\vec{y} = \mathcal{M}(\mathcal{F}(\mathcal{N}(\mathcal{F}, \cdot, \vec{b})), \vec{x}_i, \vec{b}) \tag{17}$$

$$\vec{x} = \mathcal{N}(\mathcal{F}, \vec{y}, \vec{b}) \tag{18}$$

Nonlinear Preconditioning

Type	Sym	Statement	Abbreviation
Additive	$+$	$\vec{x} + \alpha(\mathcal{M}(\mathcal{F}, \vec{x}, \vec{b}) - \vec{x})$ $+ \beta(\mathcal{N}(\mathcal{F}, \vec{x}, \vec{b}) - \vec{x})$	$\mathcal{M} + \mathcal{N}$
Multiplicative	$*$	$\mathcal{M}(\mathcal{F}, \mathcal{N}(\mathcal{F}, \vec{x}, \vec{b}), \vec{b})$	$\mathcal{M} * \mathcal{N}$
Left Prec.	$-_L$	$\mathcal{M}(\vec{x} - \mathcal{N}(\mathcal{F}, \vec{x}, \vec{b}), \vec{x}, \vec{b})$	$\mathcal{M} -_L \mathcal{N}$
Right Prec.	$-_R$	$\mathcal{M}(\mathcal{F}(\mathcal{N}(\mathcal{F}, \vec{x}, \vec{b})), \vec{x}, \vec{b})$	$\mathcal{M} -_R \mathcal{N}$
Inner Lin. Inv.	$\backslash$	$\vec{y} = \vec{J}(\vec{x})^{-1}\vec{r}(\vec{x}) = \mathrm{K}(\vec{J}(\vec{x}), \vec{y}_0, \vec{b})$	$\mathcal{N} \backslash \mathrm{K}$

Composing Scalable Nonlinear Algebraic Solvers,
Brune, Knepley, Smith, and Tu, SIAM Review, 2015.

Outline

Solver Organization
Preconditioned Newton-Krylov

We can combine Newton-Krylov with Nonlinear CG:

$$(\mathrm{NCG} -_L \mathrm{NRICH}) * (\mathcal{N} \backslash \mathrm{K} -_L \mathrm{NRICH})$$

NCG*Newton options

```
-snes_monitor -snes_converged_reason
-snes_type composite -snes_composite_type multiplicative
 -snes_composite_sneses ncg,newtonls
  -sub_0_snes_monitor -sub_1_snes_monitor
  -sub_0_snes_type ncg -sub_0_snes_linesearch_type cp
   -sub_0_snes_max_it 5
  -sub_1_snes_linesearch_type bt -sub_1_snes_fd_color
   -sub_1_snes_fd_color_use_mat -mat_coloring_type greedy
  -sub_1_ksp_rtol 1.0e-10 -sub_1_ksp_monitor -sub_1_ksp_gmres_restart 200
  -sub_1_pc_type fieldsplit -sub_1_pc_fieldsplit_0_fields 0,2
   -sub_1_pc_fieldsplit_1_fields 1
  -sub_1_pc_fieldsplit_type schur
   -sub_1_pc_fieldsplit_schur_precondition selfp
   -sub_1_pc_fieldsplit_schur_factorization_type full
    -sub_1_fieldsplit_0_pc_type lu
    -sub_1_fieldsplit_pressure_ksp_rtol 1.0e-9
     -sub_1_fieldsplit_pressure_pc_type gamg
     -sub_1_fieldsplit_pressure_ksp_gmres_restart 100
     -sub_1_fieldsplit_pressure_ksp_max_it 200
```

NCG*Newton convergence

```
0 TS dt 0.01 time 0.64
    0 SNES Function norm 9.682733054059e-03
      0 SNES Function norm 9.682733054059e-03
      1 SNES Function norm 3.705698943518e-02
      2 SNES Function norm 4.981898384331e-02
      3 SNES Function norm 5.710183285964e-02
      4 SNES Function norm 5.476973798534e-02
      5 SNES Function norm 6.464724668855e-02
      0 SNES Function norm 6.464724668855e-02
        0 KSP Residual norm 1.021155502263e+00
        1 KSP Residual norm 9.145207488003e-05
        2 KSP Residual norm 3.899752904206e-09
        3 KSP Residual norm 1.001750831581e-12
      1 SNES Function norm 8.940296814443e-03
    1 SNES Function norm 8.940296814443e-03
    2 SNES Function norm 4.290429277269e-02
    3 SNES Function norm 1.154466745956e-02
    4 SNES Function norm 2.938816182982e-03
    5 SNES Function norm 4.148507767082e-04
    6 SNES Function norm 1.892807106900e-05
    7 SNES Function norm 4.912654244547e-08
    8 SNES Function norm 3.851626525260e-13
1 TS dt 0.01 time 0.65
```

Solver Organization
Full Approximation Scheme

We can use Newton-Krylov as a level solver for FAS:

$$\mathrm{FAS}(\mathcal{N}\backslash\mathrm{K}, \mathcal{N}\backslash\mathrm{K})$$

FAS-Newton options

Top level

```
-snes_monitor -snes_converged_reason
-snes_type fas -snes_fas_type full -snes_fas_levels 4
 -fas_levels_3_snes_monitor -fas_levels_3_snes_converged_reason
  -fas_levels_3_snes_atol 1.0e-9 -fas_levels_3_snes_max_it 2
  -fas_levels_3_snes_type newtonls -fas_levels_3_snes_linesearch_type bt
  -fas_levels_3_snes_fd_color -fas_levels_3_snes_fd_color_use_mat
  -fas_levels_3_ksp_rtol 1.0e-10 -mat_coloring_type greedy
  -fas_levels_3_ksp_gmres_restart 50 -fas_levels_3_ksp_max_it 200
   -fas_levels_3_pc_type fieldsplit
    -fas_levels_3_pc_fieldsplit_0_fields 0,2
    -fas_levels_3_pc_fieldsplit_1_fields 1
    -fas_levels_3_pc_fieldsplit_type schur
     -fas_levels_3_pc_fieldsplit_schur_precondition selfp
     -fas_levels_3_pc_fieldsplit_schur_factorization_type full
      -fas_levels_3_fieldsplit_0_pc_type lu
      -fas_levels_3_fieldsplit_pressure_ksp_rtol 1.0e-9
       -fas_levels_3_fieldsplit_pressure_pc_type gamg
       -fas_levels_3_fieldsplit_pressure_ksp_gmres_restart 100
       -fas_levels_3_fieldsplit_pressure_ksp_max_it 200
```

FAS-Newton options

2nd level

```
-fas_levels_2_snes_monitor -fas_levels_2_snes_converged_reason
 -fas_levels_2_snes_atol 1.0e-9 -fas_levels_2_snes_max_it 2
 -fas_levels_2_snes_type newtonls -fas_levels_2_snes_linesearch_type bt
 -fas_levels_2_snes_fd_color -fas_levels_2_snes_fd_color_use_mat
 -fas_levels_2_ksp_rtol 1.0e-10 -fas_levels_2_ksp_gmres_restart 50
 -fas_levels_2_pc_type fieldsplit
  -fas_levels_2_pc_fieldsplit_0_fields 0,2
  -fas_levels_2_pc_fieldsplit_1_fields 1
  -fas_levels_2_pc_fieldsplit_type schur
   -fas_levels_2_pc_fieldsplit_schur_precondition selfp
   -fas_levels_2_pc_fieldsplit_schur_factorization_type full
    -fas_levels_2_fieldsplit_0_pc_type lu
    -fas_levels_2_fieldsplit_pressure_ksp_rtol 1.0e-9
     -fas_levels_2_fieldsplit_pressure_pc_type gamg
     -fas_levels_2_fieldsplit_pressure_ksp_gmres_restart 100
     -fas_levels_2_fieldsplit_pressure_ksp_max_it 200
```

FAS-Newton options

1st level

```
-fas_levels_1_snes_monitor -fas_levels_1_snes_converged_reason
 -fas_levels_1_snes_atol 1.0e-9
 -fas_levels_1_snes_type newtonls -fas_levels_1_snes_linesearch_type bt
 -fas_levels_1_snes_fd_color -fas_levels_1_snes_fd_color_use_mat
 -fas_levels_1_ksp_rtol 1.0e-10 -fas_levels_1_ksp_gmres_restart 50
 -fas_levels_1_pc_type fieldsplit
  -fas_levels_1_pc_fieldsplit_0_fields 0,2
  -fas_levels_1_pc_fieldsplit_1_fields 1
  -fas_levels_1_pc_fieldsplit_type schur
   -fas_levels_1_pc_fieldsplit_schur_precondition selfp
   -fas_levels_1_pc_fieldsplit_schur_factorization_type full
    -fas_levels_1_fieldsplit_0_pc_type lu
    -fas_levels_1_fieldsplit_pressure_ksp_rtol 1.0e-9
    -fas_levels_1_fieldsplit_pressure_pc_type gamg
```

FAS-Newton options

Coarse level

```
-fas_coarse_snes_monitor -fas_coarse_snes_converged_reason
 -fas_coarse_snes_atol 1.0e-9
 -fas_coarse_snes_type newtonls -fas_coarse_snes_linesearch_type bt
 -fas_coarse_snes_fd_color -fas_coarse_snes_fd_color_use_mat
 -fas_coarse_ksp_rtol 1.0e-10 -fas_coarse_ksp_gmres_restart 50
 -fas_coarse_pc_type fieldsplit
  -fas_coarse_pc_fieldsplit_0_fields 0,2
  -fas_coarse_pc_fieldsplit_1_fields 1
  -fas_coarse_pc_fieldsplit_type schur
   -fas_coarse_pc_fieldsplit_schur_precondition selfp
   -fas_coarse_pc_fieldsplit_schur_factorization_type full
    -fas_coarse_fieldsplit_0_pc_type lu
    -fas_coarse_fieldsplit_pressure_ksp_rtol 1.0e-9
    -fas_coarse_fieldsplit_pressure_pc_type gamg
```

FAS-Newton convergence

```
0 TS dt 0.01 time 0.64
    0 SNES Function norm 9.682733054059e-03
      2 SNES Function norm 4.412420553822e-03
            2 SNES Function norm 8.022096211721e-15
          1 SNES Function norm 2.773743832538e-04
            1 SNES Function norm 5.627093528843e-11
          1 SNES Function norm 4.405884464849e-10
        2 SNES Function norm 8.985059910030e-08
          1 SNES Function norm 4.672651281994e-15
            0 SNES Function norm 3.160322858961e-15
          0 SNES Function norm 4.672651281994e-15
        1 SNES Function norm 1.046571008046e-14
      2 SNES Function norm 1.804845173803e-02
        2 SNES Function norm 2.776600115290e-12
          0 SNES Function norm 1.354009326059e-12
            0 SNES Function norm 5.881604627760e-13
          0 SNES Function norm 1.354011456281e-12
        0 SNES Function norm 2.776600115290e-12
      2 SNES Function norm 9.640723411562e-05
    1 SNES Function norm 9.640723411562e-05
    2 SNES Function norm 1.057876040732e-08
    3 SNES Function norm 5.623618219189e-11
1 TS dt 0.01 time 0.65
```

Solver Organization
Full Approximation Scheme

On fine levels, we can replace Newton-Krylov with

Nonlinear Gauss-Siedel:

$$\mathrm{FAS}(\mathrm{NGS}, \mathcal{N} \backslash \mathrm{K})$$

FAS-NGS options

Top level

```
-snes_monitor -snes_converged_reason
-snes_type fas -snes_fas_type full -snes_fas_levels 4
 -fas_levels_3_snes_monitor -fas_levels_3_snes_converged_reason
  -fas_levels_3_snes_atol 1.0e-9 -fas_levels_3_snes_max_it 10
  -fas_levels_3_snes_type ngs -fas_levels_3_snes_linesearch_type nleqerr
```

FAS-NGS convergence

```
0 TS dt 0.01 time 0.64
Time 0.64 L_2 Error: 0.494811 [0.0413666, 0.491642, 0.0376071]
    0 SNES Function norm 9.68e-03 [1.96e-03, 1.71e-14, 9.65e-03]
      0 SNES Function norm 9.682733054059e-03
      3 SNES Function norm 9.069944580453e-01
          3 SNES Function norm 3.790367845975e-11
        0 SNES Function norm 1.884126634610e+00
        1 SNES Function norm 6.752057466899e-02
          2 SNES Function norm 3.799909413083e-11
        0 SNES Function norm 1.450032375835e-01
        1 SNES Function norm 2.567674743706e-04
      0 SNES Function norm 1.027806561203e+00
      3 SNES Function norm 1.582489644172e+00
        1 SNES Function norm 4.847533456932e-01
          3 SNES Function norm 7.366666076108e-15
        1 SNES Function norm 1.744390611632e-02
      3 SNES Function norm 1.473321454964e+00
    1 SNES Function norm 1.47e+00 [1.44e+00, 2.92e-01, 8.82e-04]
      0 SNES Function norm 9.962396109825e+03
      1 SNES Function norm 3.189537494940e+86
    Nonlinear fas_levels_2_ solve did not converge, DIVERGED_FNORM_NAN
  Nonlinear solve did not converge due to DIVERGED_INNER
```

Outline

Solver Organization
Timestepping

We can use a simple split scheme:

$$\nabla p^{k+1} - \nabla\left(\left(\frac{\phi^{k}}{\phi_0}\right)^{-m}\nabla\cdot\vec{v}^{k+1}\right) - \nabla\cdot\left(2e^{-\lambda(\phi^{k}-\phi_0)}\dot{\epsilon}^{k+1}\right) = 0$$

$$\nabla\cdot\left(-\frac{R^2}{r_\zeta + 4/3}\left(\frac{\phi^{k}}{\phi_0}\right)^{n}\nabla p^{k+1} + \vec{v}^{k+1}\right) = 0$$

$$\frac{\phi^{k+1} - \phi^{k}}{\Delta t} - \nabla\cdot(1-\phi^{k})\vec{v}^{k+1} = 0$$

Solver Organization
Timestepping

Or one that couples the algebraic and evolution equations:

$$\nabla p^{k+1} - \nabla \left(\left(\frac{\phi^{k+1}}{\phi_0} \right)^{-m} \nabla \cdot \vec{v}^{k+1} \right) - \nabla \cdot \left(2 e^{-\lambda(\phi^{k+1} - \phi_0)} \dot{\epsilon}^{k+1} \right) = 0$$

$$\nabla \cdot \left(- \frac{R^2}{r_\zeta + 4/3} \left(\frac{\phi^{k+1}}{\phi_0} \right)^{n} \nabla p^{k+1} + \vec{v}^{k+1} \right) = 0$$

$$\frac{\phi^{k+1} - \phi^k}{\Delta t} - \nabla \cdot (1 - \phi^k) \vec{v}^{k+1} = 0$$

Newton options

```
-snes_atol 1.0e-10 -snes_monitor_field -snes_converged_reason
-snes_linesearch_type basic -snes_fd_color -snes_fd_color_use_mat
-mat_coloring_type greedy -mat_coloring_greedy_symmetric 0
 -ksp_rtol 1.0e-10 -ksp_monitor -ksp_gmres_restart 200
 -pc_type fieldsplit
 -pc_fieldsplit_0_fields 0,2 -pc_fieldsplit_1_fields 1
 -pc_fieldsplit_type schur -pc_fieldsplit_schur_precondition selfp
 -pc_fieldsplit_schur_factorization_type full
  -fieldsplit_0_ksp_rtol 1.0e-8  -fieldsplit_0_pc_type lu
  -fieldsplit_pressure_ksp_rtol 1.0e-9 -fieldsplit_pressure_pc_type svd
```

Early Newton convergence

```
5 TS dt 0.005 time 0.025
    0 SNES Function norm 6.52e-02 [1.46e-14, 4.91e-16, 6.52e-02]
      0 KSP Residual norm 4.26e-04
      1 KSP Residual norm 1.78e-17
    1 SNES Function norm 2.19e-03 [2.96e-08, 3.91e-09, 2.19e-03]
    2 SNES Function norm 7.51e-05 [3.40e-11, 4.55e-12, 7.51e-05]
    3 SNES Function norm 2.58e-06 [2.46e-13, 1.28e-14, 2.58e-06]
    4 SNES Function norm 8.86e-08 [1.39e-14, 6.64e-16, 8.86e-08]
6 TS dt 0.005 time 0.03
```

Late Newton convergence

```
0 TS dt 0.005 time 0.825
    0 SNES Function norm 2.14e+00 [  1.40e-14,   3.67e-16,   2.14e+00]
      0 KSP Residual norm 3.53e-01
      1 KSP Residual norm 1.03e-10
      2 KSP Residual norm 2.82e-16
    1 SNES Function norm 5.13e-02 [  2.01e-04,   1.47e-04,   5.13e-02]
    2 SNES Function norm 2.47e-02 [  9.20e-06,   7.73e-06,   2.47e-02]
    3 SNES Function norm 7.81e-03 [  2.13e-06,   1.67e-06,   7.81e-03]
    4 SNES Function norm 2.12e-03 [  1.81e-07,   1.41e-07,   2.12e-03]
    5 SNES Function norm 4.72e-04 [  1.08e-08,   8.28e-09,   4.72e-04]
    6 SNES Function norm 1.12e-04 [  5.76e-10,   4.41e-10,   1.12e-04]
    7 SNES Function norm 2.63e-05 [  3.21e-11,   2.50e-11,   2.63e-05]
    8 SNES Function norm 6.17e-06 [  1.77e-12,   1.26e-12,   6.17e-06]
    9 SNES Function norm 1.45e-06 [  1.07e-13,   9.84e-14,   1.45e-06]
   10 SNES Function norm 3.40e-07 [  1.78e-14,   4.74e-15,   3.40e-07]
   11 SNES Function norm 7.99e-08 [  1.36e-14,   1.88e-15,   7.99e-08]
   12 SNES Function norm 1.88e-08 [  1.34e-14,   5.72e-16,   1.88e-08]
1 TS dt 0.005 time 0.83
```

FAS-Newton options

Top level

```
-snes_atol 1.0e-9 -snes_monitor_field -snes_converged_reason
-snes_type fas -snes_fas_type full -snes_fas_levels 3
 -fas_levels_2_snes_monitor -fas_levels_2_snes_converged_reason
  -fas_levels_2_snes_atol 1.0e-9 -fas_levels_2_snes_max_it 2
  -fas_levels_2_snes_type newtonls
  -fas_levels_2_snes_linesearch_type basic
  -fas_levels_2_snes_fd_color -fas_levels_2_snes_fd_color_use_mat
   -fas_levels_2_ksp_rtol 1.0e-10 -fas_levels_2_ksp_gmres_restart 50
   -fas_levels_2_pc_type fieldsplit
    -fas_levels_2_pc_fieldsplit_0_fields 0,2
    -fas_levels_2_pc_fieldsplit_1_fields 1
    -fas_levels_2_pc_fieldsplit_type schur
    -fas_levels_2_pc_fieldsplit_schur_precondition selfp
    -fas_levels_2_pc_fieldsplit_schur_factorization_type full
     -fas_levels_2_fieldsplit_0_pc_type lu
     -fas_levels_2_fieldsplit_pressure_ksp_rtol 1.0e-9
     -fas_levels_2_fieldsplit_pressure_pc_type svd
     -fas_levels_2_fieldsplit_pressure_ksp_gmres_restart 100
     -fas_levels_2_fieldsplit_pressure_ksp_max_it 200
```

FAS-Newton options

Coarse level

```
-fas_coarse_snes_max_it 10 -fas_coarse_snes_max_linear_solve_fail 10
-fas_coarse_snes_atol 1.0e-9
-fas_coarse_snes_monitor -fas_coarse_snes_converged_reason
-fas_coarse_snes_type newtonls -fas_coarse_snes_linesearch_type bt
-fas_coarse_snes_fd_color -fas_coarse_snes_fd_color_use_mat
 -fas_coarse_ksp_rtol 1.0e-10 -fas_coarse_ksp_gmres_restart 50
 -fas_coarse_pc_type fieldsplit
 -fas_coarse_pc_fieldsplit_0_fields 0,2
 -fas_coarse_pc_fieldsplit_1_fields 1
 -fas_coarse_pc_fieldsplit_type schur
 -fas_coarse_pc_fieldsplit_schur_precondition selfp
 -fas_coarse_pc_fieldsplit_schur_factorization_type full
  -fas_coarse_fieldsplit_0_pc_type lu
  -fas_coarse_fieldsplit_pressure_ksp_rtol 1.0e-9
  -fas_coarse_fieldsplit_pressure_pc_type svd
```

FAS-Newton convergence

```
0 TS dt 0.005 time 0.825
    0 SNES Function norm 2.14e+00 [1.40e-14, 3.67e-16, 2.14e+00]
      0 SNES Function norm 2.136811983007e+00
      2 SNES Function norm 2.467490038458e-02
          0 SNES Function norm 2.892788645925e-02
          5 SNES Function norm 6.686368379854e-11
        0 SNES Function norm 5.034219273717e-02
        1 SNES Function norm 1.054842559307e-03
          0 SNES Function norm 1.663080254945e-03
          4 SNES Function norm 2.126370356882e-10
        0 SNES Function norm 2.599480303180e-03
        1 SNES Function norm 9.990047497418e-05
      0 SNES Function norm 4.798584798600e-02
      2 SNES Function norm 1.288870672992e-03
        1 SNES Function norm 3.770621359658e-05
          2 SNES Function norm 1.127970439777e-08
        1 SNES Function norm 1.008431552413e-06
      0 SNES Function norm 2.502531975042e-03
      2 SNES Function norm 4.730240156687e-05
    1 SNES Function norm 4.73e-05 [1.04e-10, 7.85e-11, 4.73e-05]
    2 SNES Function norm 3.98e-09 [1.38e-14, 4.18e-16, 3.98e-09]
1 TS dt 0.005 time 0.83
```

FAS-NGS options

Top level

```
-snes_atol 1.0e-9 -snes_monitor_field -snes_converged_reason
-snes_type fas -snes_fas_type full -snes_fas_levels 3
  -fas_levels_2_snes_monitor -fas_levels_2_snes_converged_reason
  -fas_levels_2_snes_atol 1.0e-9 -fas_levels_2_snes_max_it 10
  -fas_levels_2_snes_type ngs -fas_levels_2_snes_linesearch_type bt
```

FAS-NGS convergence

```
0 TS dt 0.005 time 0.825
    0 SNES Function norm 2.14e+00 [1.39e-14, 3.66e-16, 2.13e+00]
    1 SNES Function norm 2.87e-04 [2.08e-04, 4.80e-06, 1.98e-04]
    2 SNES Function norm 3.15e-05 [2.30e-05, 9.56e-07, 2.19e-05]
    3 SNES Function norm 1.65e-05 [1.14e-05, 5.44e-07, 1.21e-05]
    4 SNES Function norm 1.07e-05 [7.38e-06, 3.48e-07, 7.89e-06]
    5 SNES Function norm 7.06e-06 [4.85e-06, 2.26e-07, 5.20e-06]
    6 SNES Function norm 4.67e-06 [3.20e-06, 1.48e-07, 3.44e-06]
    7 SNES Function norm 3.09e-06 [2.12e-06, 9.82e-08, 2.27e-06]
    8 SNES Function norm 2.05e-06 [1.40e-06, 6.52e-08, 1.50e-06]
    9 SNES Function norm 1.36e-06 [9.35e-07, 4.34e-08, 1.00e-06]
   10 SNES Function norm 9.03e-07 [6.21e-07, 2.89e-08, 6.64e-07]
   11 SNES Function norm 6.00e-07 [4.13e-07, 1.94e-08, 4.41e-07]
   12 SNES Function norm 3.99e-07 [2.75e-07, 1.30e-08, 2.94e-07]
   13 SNES Function norm 2.67e-07 [1.84e-07, 8.84e-09, 1.96e-07]
   14 SNES Function norm 1.78e-07 [1.23e-07, 6.01e-09, 1.31e-07]
   15 SNES Function norm 1.20e-07 [8.31e-08, 4.12e-09, 8.80e-08]
   16 SNES Function norm 8.12e-08 [5.64e-08, 2.85e-09, 5.94e-08]
   17 SNES Function norm 5.55e-08 [3.87e-08, 1.99e-09, 4.05e-08]
   18 SNES Function norm 3.86e-08 [2.70e-08, 1.41e-09, 2.80e-08]
   19 SNES Function norm 2.74e-08 [1.93e-08, 1.01e-09, 1.97e-08]
   20 SNES Function norm 2.00e-08 [1.43e-08, 7.48e-10, 1.44e-08]
1 TS dt 0.005 time 0.83
```

Outline

Can't you just pick

the *right* discretization?

Continuous Galerkin

Can't you just use CG for everything?

- Good idea for elliptic equations
  Mass and momentum conservation
- Bad for porosity advection (very diffusive)
  Easy to see in uniform advection of a porous block
- Solitary wave benchmark does not test diffusivity
  because you can make the wave stand almost still
- Shear band benchmark does not test diffusivity
  but localization will be artificially arrested

Continuous Galerkin

Can't you just use CG for everything?

- Good idea for elliptic equations
  Mass and momentum conservation
- Bad for porosity advection (very diffusive)
  Easy to see in uniform advection of a porous block
- Solitary wave benchmark does not test diffusivity
  because you can make the wave stand almost still
- Shear band benchmark does not test diffusivity
  but localization will be artificially arrested

Continuous Galerkin

Can't you just use CG for everything?

- Good idea for elliptic equations
  Mass and momentum conservation
- Bad for porosity advection (very diffusive)
  Easy to see in uniform advection of a porous block
- Solitary wave benchmark does not test diffusivity
  because you can make the wave stand almost still
- Shear band benchmark does not test diffusivity
  but localization will be artificially arrested

Continuous Galerkin

Can't you just use CG for everything?

- Good idea for elliptic equations
  Mass and momentum conservation
- Bad for porosity advection (very diffusive)
  Easy to see in uniform advection of a porous block
- Solitary wave benchmark does not test diffusivity
  because you can make the wave stand almost still
- Shear band benchmark does not test diffusivity
  but localization will be artificially arrested

Discontinuous Galerkin

Can't you just use DG for everything?

- Good idea for advection
- Bad idea for conservation of mass and momentum
  Forcing continuity results (morally) in penalization
  creating hard-to-solve systems of equations, resulting in . . .

Discontinuous Galerkin

Can't you just use DG for everything?

- Good idea for advection
- Bad idea for conservation of mass and momentum
  Forcing continuity results (morally) in penalization
  creating hard-to-solve systems of equations, resulting in ...

Discontinuous Galerkin

Can't you just use DG for everything?

- Good idea for advection
- Bad idea for conservation of mass and momentum
  Forcing continuity results (morally) in penalization
  creating hard-to-solve systems of equations, resulting in ...

Discontinuous Galerkin

Can't you just use DG for everything?

- Good idea for advection
- Bad idea for conservation of mass and momentum
  Forcing continuity results (morally) in penalization
- Bad idea for efficiency
  Lehmann, Lukáčová-Medvid'ová, Kaus and Popov, ZAMM, 2015
  DG has higher cost for equivalent accuracy, resulting in . . .

Discontinuous Galerkin

Can't you just use DG for everything?

- Good idea for advection
- Bad idea for conservation of mass and momentum
  Forcing continuity results (morally) in penalization
- Bad idea for efficiency
  DG has higher cost for equivalent accuracy, resulting in ...

Discontinuous Galerkin

DG makes a lot of sense for the advection equation

- Not diffusive
- Handles sharp fronts
- Can be high order accurate
- Implicit formulation makes sense
- Needs a limiter to prevent oscillation

Finite Volume

FV also makes sense for the advection equation

- Not diffusive
- Handles sharp fronts
- Can be low order accurate
- Implicit formulation does not make sense
- Needs a limiter to prevent oscillation

Composability

Can we mix discretizations?

- FEniCS/Firedrake can do CG+DG
- Deal.II can do CG+DG
- PETSc can do CG+FV

Discretization

Using continuous FE spaces,

which satisfy an inf-sup stability condition.

Discretization

Using continuous FE spaces,

$$\begin{array}{ll} Q2 & \text{velocity} \\ Q1 & \text{pressure} \\ Q1 & \text{porosity} \end{array}$$

which satisfy an inf-sup stability condition.

Discretization

Using continuous FE spaces,

```
-velocity_petscspace_order 2
  -velocity_petscspace_poly_tensor
-pressure_petscspace_order 1
  -pressure_petscspace_poly_tensor
-porosity_petscspace_order 1
  -porosity_petscspace_poly_tensor
```

which satisfy an inf-sup stability condition.

Discretization

Using continuous/discontinuous FE spaces,

$$\begin{array}{ll} Q2 & \text{velocity} \\ P1_{\mathrm{disc}} & \text{pressure} \\ Q1 & \text{porosity} \end{array}$$

which satisfy an inf-sup stability condition.

Discretization

Using continuous/discontinuous FE spaces,

```
-velocity_petscspace_order 2
  -velocity_petscspace_poly_tensor
-pressure_petscspace_order 1
  -pressure_petscdualspace_lagrange_continuity 0
-porosity_petscspace_order 1
  -porosity_petscspace_poly_tensor
```

which satisfy an inf-sup stability condition.

Discretization

Using continuous FE spaces and simple FV,

$Q2$	velocity
$Q1$	pressure
FV	porosity

which we connect by cell/face interpolants.

```
/* Set discretization object */
if (user->useFV) {
  PetscDSSetDiscretization(prob, 2, fv);
} else {
  PetscDSSetDiscretization(prob, 2, fe[2]);
}
/* Set pointwise residual functions */
PetscDSSetResidual(prob, 2, f0_advection, f1_scalar_zero);
PetscDSSetRiemannSolver(prob, 2, riemann_coupled_advection);
```

Discretization

Using continuous FE spaces and simple FV,

```
-velocity_petscspace_order 2
  -velocity_petscspace_poly_tensor
-pressure_petscspace_order 1
  -pressure_petscspace_poly_tensor
-use_fv
```

which we connect by cell/face interpolants.

```
/* Set discretization object */
if (user->useFV) {
  PetscDSSetDiscretization(prob, 2, fv);
} else {
  PetscDSSetDiscretization(prob, 2, fe[2]);
}
/* Set pointwise residual functions */
PetscDSSetResidual(prob, 2, f0_advection, f1_scalar_zero);
PetscDSSetRiemannSolver(prob, 2, riemann_coupled_advection);
```

Discretization

Using continuous FE spaces and simple FV,

$Q2$	velocity
$Q1$	pressure
FV	porosity

which we connect by cell/face interpolants.

```
/* Set discretization object */
if (user->useFV) {
  PetscDSSetDiscretization(prob, 2, fv);
} else {
  PetscDSSetDiscretization(prob, 2, fe[2]);
}
/* Set pointwise residual functions  */
PetscDSSetResidual(prob, 2, f0_advection, f1_scalar_zero);
PetscDSSetRiemannSolver(prob, 2, riemann_coupled_advection);
```

Adaptive Mesh Refinement

Should we use AMR?

- Uniform refinement is scalable, but
  - AMR achieves higher accuracy on fixed resources, or
  - AMR uses less memory for fixes accuracy
- AMR could affect the nonlinear conditioning
- AMR complicates multilevel solves

Adaptive Mesh Refinement

Should we use AMR?

- Uniform refinement is scalable, but
  - AMR achieves higher accuracy on fixed resources, or
  - AMR uses less memory for fixes accuracy
- AMR could affect the nonlinear conditioning
- AMR complicates multilevel solves

Adaptive Mesh Refinement

Should we use AMR?

- Uniform refinement is scalable, but
  - AMR achieves higher accuracy on fixed resources, or
  - AMR uses less memory for fixes accuracy
- AMR could affect the nonlinear conditioning
- AMR complicates multilevel solves

Adaptive Mesh Refinement

Should we use AMR?

- Uniform refinement is scalable, but
  - AMR achieves higher accuracy on fixed resources, or
  - AMR uses less memory for fixes accuracy
- AMR could affect the nonlinear conditioning
- AMR complicates multilevel solves

Adaptive Mesh Refinement

Should we use AMR?

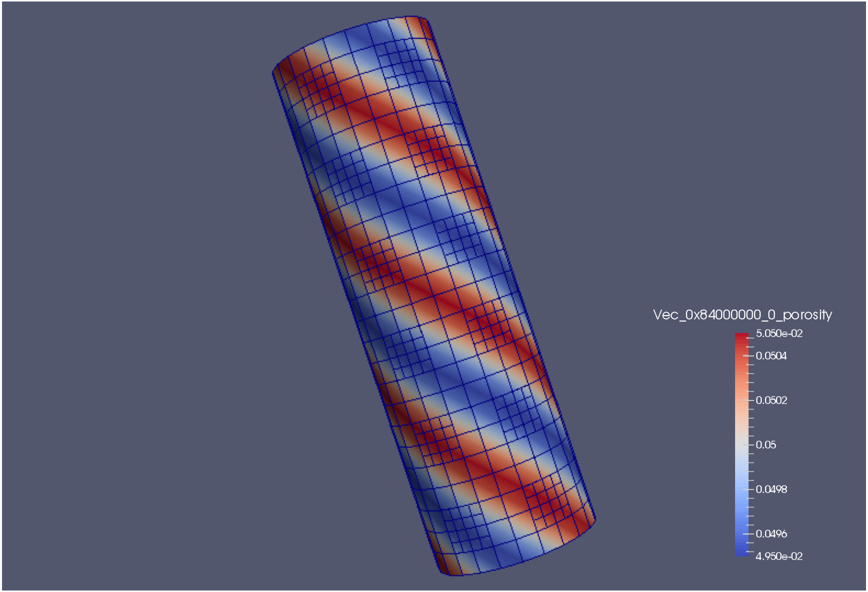
- Uniform refinement is scalable, but
  - AMR achieves higher accuracy on fixed resources, or
  - AMR uses less memory for fixes accuracy
- AMR could affect the nonlinear conditioning
- AMR complicates multilevel solves

AMR Implementation

Can we use AMR?

- LibMesh can do AMR
- Deal.II can do AMR
- PETSc can do AMR

AMR Shear Bands



Outline

Conclusions

We need composable,

extensible systems

to match numerics

to the physics.

Conclusions

We need composable,
extensible systems
to match numerics
to the physics.

Conclusions

We need composable,
extensible systems
to match numerics
to the physics.

Conclusions

We need composable,
extensible systems
to match numerics
to the physics.

Outline

Outline

Rate of Convergence

What should be a Rate of Convergence? [Ptak, 1977]:

1. It should relate quantities which may be measured or estimated during the actual process
2. It should describe accurately in particular the initial stage of the process, not only its asymptotic behavior …

$$\|x_{n+1} - x^*\| \leq c\|x_n - x^*\|^q$$

Rate of Convergence

What should be a Rate of Convergence? [Ptak, 1977]:

1. It should relate quantities which may be measured or estimated during the actual process
2. It should describe accurately in particular the initial stage of the process, not only its asymptotic behavior ...

$$\|x_{n+1} - x_n\| \leq c\|x_n - x_{n-1}\|^q$$

Rate of Convergence

What should be a Rate of Convergence? [Ptak, 1977]:

1. It should relate quantities which may be measured or estimated during the actual process
2. It should describe accurately in particular the initial stage of the process, not only its asymptotic behavior ...

$$\|x_{n+1} - x_n\| \leq \omega(\|x_n - x_{n-1}\|)$$

where we have for all $r \in (0, R]$

$$\sigma(r) = \sum_{n=0}^{\infty} \omega^{(n)}(r) < \infty$$

Nondiscrete Induction

Define an approximate set $Z(r)$, where $x^* \in Z(0)$ implies $f(x^*) = 0$.

Nondiscrete Induction

Define an approximate set $Z(r)$, where $x^* \in Z(0)$ implies $f(x^*) = 0$.

For Newton's method, we use

$$Z(r) = \left\{ x \middle| \|f'(x)^{-1} f(x)\| \le r, d(f'(x)) \ge h(r), \|x - x_0\| \le g(r) \right\},$$

where

$$d(A) = \inf_{\|x\| \ge 1} \|Ax\|,$$

and $h(r)$ and $g(r)$ are positive functions.

Nondiscrete Induction

Define an approximate set $Z(r)$, where $x^* \in Z(0)$ implies $f(x^*) = 0$.

For $r \in (0, R]$,

$$Z(r) \subset U(Z(\omega(r)), r)$$

implies

$$Z(r) \subset U(Z(0), \sigma(r)).$$

Nondiscrete Induction

For the fixed point iteration

$$x_{n+1} = Gx_n,$$

if I have

$$x_0 \in Z(r_0)$$

and for $x \in Z(r)$,

$$\begin{aligned} \|Gx - x\| &\leq r \\ Gx &\in Z(\omega(r)) \end{aligned}$$

then

Nondiscrete Induction

For the fixed point iteration

$$x_{n+1} = Gx_n,$$

if I have

$$x_0 \in Z(r_0)$$

and for $x \in Z(r)$,

$$\begin{aligned} \|Gx - x\| &\leq r \\ Gx &\in Z(\omega(r)) \end{aligned}$$

then

$$\begin{aligned} x^* &\in Z(0) \\ x_n &\in Z(\omega^{(n)}(r_0)) \end{aligned}$$

Nondiscrete Induction

For the fixed point iteration

$$x_{n+1} = Gx_n,$$

if I have

$$x_0 \in Z(r_0)$$

and for $x \in Z(r)$,

$$\begin{aligned} \|Gx - x\| &\leq r \\ Gx &\in Z(\omega(r)) \end{aligned}$$

then

$$\begin{aligned} \|x_{n+1} - x_n\| &\leq \omega^{(n)}(r_0) \\ \|x_n - x^*\| &\leq \sigma(\omega^{(n)}(r_0)) \end{aligned}$$

Nondiscrete Induction

For the fixed point iteration

$$x_{n+1} = Gx_n,$$

if I have

$$x_0 \in Z(r_0)$$

and for $x \in Z(r)$,

$$\begin{aligned} \|Gx - x\| &\leq r \\ Gx &\in Z(\omega(r)) \end{aligned}$$

then

$$\begin{aligned} \|x_n - x^*\| &\leq \sigma(\omega(\|x_n - x_{n-1}\|)) \\ &= \sigma(\|x_n - x_{n-1}\|) - \|x_n - x_{n-1}\| \end{aligned}$$

Newton's Method

$$\omega_{\mathcal{N}}(r) = cr^2$$

Newton's Method

$$\omega_{\mathcal{N}}(r) = \frac{r^2}{2\sqrt{r^2 + a^2}}$$
$$\sigma_{\mathcal{N}}(r) = r + \sqrt{r^2 + a^2} - a$$

where

$$a = \frac{1}{k_0}\sqrt{1 - 2k_0 r_0},$$

k_0 is the (scaled) Lipschitz constant for f', and
r_0 is the (scaled) initial residual.

Newton's Method

$$\omega_{\mathcal{N}}(r) = \frac{r^2}{2\sqrt{r^2 + a^2}}$$
$$\sigma_{\mathcal{N}}(r) = r + \sqrt{r^2 + a^2} - a$$

This estimate is *tight* in that the bounds hold with equality for some function f,

$$f(x) = x^2 - a^2$$

using initial guess

$$x_0 = \frac{1}{k_0}.$$

Also, if equality is attained for some n_0, this holds for all $n \geq n_0$.

Newton's Method

$$\omega_{\mathcal{N}}(r) = \frac{r^2}{2\sqrt{r^2 + a^2}}$$
$$\sigma_{\mathcal{N}}(r) = r + \sqrt{r^2 + a^2} - a$$

If $r \gg a$, meaning we have an inaccurate guess,

$$\omega_{\mathcal{N}}(r) \approx \frac{1}{2} r,$$

whereas if $r \ll a$, meaning we are close to the solution,

$$\omega_{\mathcal{N}}(r) \approx \frac{1}{2a} r^2.$$

Left vs. Right

Left:

$$\mathcal{F}(x) \Longrightarrow x - \mathcal{N}(\mathcal{F}, x, b)$$

Right:

$$x \Longrightarrow y = \mathcal{N}(\mathcal{F}, x, b)$$

Heisenberg vs. Schrödinger Picture

Left vs. Right

Left:

$$\mathcal{F}(x) \Longrightarrow x - \mathcal{N}(\mathcal{F}, x, b)$$

Right:

$$x \Longrightarrow y = \mathcal{N}(\mathcal{F}, x, b)$$

Heisenberg vs. Schrödinger Picture

$\mathcal{M} -_R \mathcal{N}$

We start with $x \in Z(r)$, apply $\mathcal{N}$ so that

$$y \in Z(\omega_{\mathcal{N}}(r)),$$

and then apply $\mathcal{M}$ so that

$$x' \in Z(\omega_{\mathcal{M}}(\omega_{\mathcal{N}}(r))).$$

Thus we have

$$\omega_{\mathcal{M} -_R \mathcal{N}} = \omega_{\mathcal{M}} \circ \omega_{\mathcal{N}}$$

Non-Abelian

$\mathcal{N} -_R$ NRICH

$$\begin{aligned}
\omega_{\mathcal{N}} \circ \omega_{\text{NRICH}} &= \frac{1}{2}\frac{r^2}{\sqrt{r^2+a^2}} \circ cr, \\
&= \frac{1}{2}\frac{c^2 r^2}{\sqrt{c^2 r^2+a^2}}, \\
&= \frac{1}{2}\frac{cr^2}{\sqrt{r^2+(a/c)^2}}, \\
&= \frac{1}{2}c\frac{r^2}{\sqrt{r^2+\tilde{a}^2}},
\end{aligned}$$

Non-Abelian

$$\mathcal{N} -_R \text{NRICH: } \frac{1}{2} c \frac{r^2}{\sqrt{r^2 + \tilde{a}^2}}$$

NRICH $-_R \mathcal{N}$

$$\begin{aligned}
\omega_{\text{NRICH}} \circ \omega_{\mathcal{N}} &= cr \circ \frac{1}{2} \frac{r^2}{\sqrt{r^2 + a^2}}, \\
&= \frac{1}{2} c \frac{r^2}{\sqrt{r^2 + a^2}}, \\
&= \frac{1}{2} c \frac{r^2}{\sqrt{r^2 + a^2}}.
\end{aligned}$$

Non-Abelian

$$\mathcal{N} -_R \text{NRICH: } \frac{1}{2} c \frac{r^2}{\sqrt{r^2 + \tilde{a}^2}}$$

$$\text{NRICH} -_R \mathcal{N}\text{: } \frac{1}{2} c \frac{r^2}{\sqrt{r^2 + a^2}}$$

The first method also changes the onset of second order convergence.

Outline

Composed Rates of Convergence

Theorem

If ω_1 *and* ω_2 *are convex rates of convergence, then* $\omega = \omega_1 \circ \omega_2$ *is a rate of convergence.*

Composed Rates of Convergence

Theorem

If ω_1 and ω_2 are convex rates of convergence, then $\omega = \omega_1 \circ \omega_2$ is a rate of convergence.

First we show that

$$\omega(s) \leq \frac{s}{r}\omega(r),$$

which means that convex rates of convergence are non-decreasing.

This implies that compositions of convex rates of convergence are also convex and non-decreasing.

Composed Rates of Convergence

Theorem

If ω_1 and ω_2 are convex rates of convergence, then $\omega = \omega_1 \circ \omega_2$ is a rate of convergence.

Then we show that

$$\omega(r) < r \qquad \forall r \in (0, R)$$

by contradiction.

Composed Rates of Convergence

Theorem

If ω_1 and ω_2 are convex rates of convergence, then $\omega = \omega_1 \circ \omega_2$ is a rate of convergence.

This is enough to show that

$$\omega_1(\omega_2(r)) < \omega_1(r),$$

and in fact

$$(\omega_1 \circ \omega_2)^{(n)}(r) < \omega_1^{(n)}(r).$$

Multidimensional Induction Theorem

Preconditions

Theorem

Let

- p *(1 for our case) and* m *(2 for our case) be two positive integers,*
- X *be a complete metric space and* $D \subset X^p$*,*
- $G : D \to X^p$ *and* $F : D \to X^{p+1}$ *be defined by* $Fu = (u, Gu)$*,*
- $F_k = P_k F$*,* $-p + 1 \leq k \leq m$*, the components of* F*,*
- $P = P_m$*,*
- $Z(r) \subset D$ *for each* $r \in T^p$*,*
- ω *be a rate of convergence of type* (p, m) *on* T*,*
- $u_0 \in D$ *and* $r_0 \in T^p$*.*

Multidimensional Induction Theorem

Theorem

If the following conditions hold

$$\begin{aligned} u_0 &\in Z(r_0), \\ PFZ(r) &\subset Z(\tilde{\omega}(r)), \\ \|F_k u - F_{k+1} u\| &\leq \omega_k(r), \end{aligned}$$

for all $r \in T^p$, $u \in Z(r)$, *and* $k = 0, \ldots, m-1$, *then*

1. u_0 *is admissible, and* $\exists x^* \in X$ *such that* $(P_k u_n)_{n \geq 0} \to x^*$,
2. *and the following relations hold for* $n > 1$,

$$\begin{aligned} Pu_n &\in Z(\tilde{\omega}(r_0)), \\ \|P_k u_n - P_{k+1} u_n\| &\leq \omega_k^{(n)}(r_0), \qquad 0 \leq k \leq m-1, \\ \|P_k u_n - x^*\| &\leq \sigma_k(\tilde{\omega}(r_0)), \qquad 0 \leq k \leq m; \end{aligned}$$

Multidimensional Induction Theorem

Theorem

If the following conditions hold

$$\begin{aligned} u_0 &\in Z(r_0), \\ PFZ(r) &\subset Z(\tilde{\omega}(r)), \\ \|F_k u - F_{k+1} u\| &\leq \omega_k(r), \end{aligned}$$

for all $r \in T^p$, $u \in Z(r)$, *and* $k = 0, \ldots, m-1$, *then*

1. u_0 *is admissible, and* $\exists x^* \in X$ *such that* $(P_k u_n)_{n \geq 0} \to x^*$,
2. *and the following relations hold for* $n > 1$,

$$\|P_k u_n - x^*\| \leq \sigma_k(r_n), \qquad 0 \leq k \leq m.$$

where $r_n \in T^p$ *and* $P u_{n-1} \in Z(r_n)$.

Multidimensional Induction Theorem

Theorem

If the following conditions hold

$$
\begin{aligned}
u_0 &\in Z(r_0), \\
PFZ(r) &\subset Z(\tilde{\omega}(r)), \\
\|F_k u - F_{k+1} u\| &\leq \omega_k(r),
\end{aligned}
$$

for all $r \in T^p$, $u \in Z(r)$, and $k = 0, \ldots, m-1$, then

1. *u_0 is admissible, and $\exists x^* \in X$ such that $(P_k u_n)_{n \geq 0} \to x^*$,*
2. *and the following relations hold for $n > 1$,*

$$
\begin{aligned}
Pu_n &\in Z(\tilde{\omega}(r_0)), \\
\|P_k u_n - P_{k+1} u_n\| &\leq \omega_k^{(n)}(r_0), \qquad 0 \leq k \leq m-1, \\
\|P_k u_n - x^*\| &\leq \sigma_k(\tilde{\omega}(r_0)), \qquad 0 \leq k \leq m;
\end{aligned}
$$

Multidimensional Induction Theorem

Theorem

If the following conditions hold

$$
\begin{aligned}
u_0 &\in Z(r_0), \\
PFZ(r) &\subset Z(\omega \circ \psi(r)), \\
\|F_0 u - F_1 u\| &\le r, \\
\|F_1 u - F_2 u\| &\le \psi(r),
\end{aligned}
$$

for all $r \in T^p$, $u \in Z(r)$, *and* $k = 0, \dots, m-1$, *then*

1. u_0 *is admissible, and* $\exists x^* \in X$ *such that* $(P_k u_n)_{n \ge 0} \to x^*$,
2. *and the following relations hold for* $n > 1$,

$$
\begin{aligned}
Pu_n &\in Z(\tilde{\omega}(r_0)), \\
\|P_k u_n - P_{k+1} u_n\| &\le \omega_k^{(n)}(r_0), \qquad 0 \le k \le m-1, \\
\|P_k u_n - x^*\| &\le \sigma_k(\tilde{\omega}(r_0)), \qquad 0 \le k \le m;
\end{aligned}
$$

Composed Newton Methods

Theorem

Suppose that we have two nonlinear solvers

- $\mathcal{M}$, Z_1, ω,
- $\mathcal{N}$, Z_0, ψ,

and consider $\mathcal{M} -_R \mathcal{N}$*, meaning a single step of* $\mathcal{N}$ *for each step of* $\mathcal{M}$.

Concretely, take $\mathcal{M}$ *to be the Newton iteration, and* $\mathcal{N}$ *the Chord method. Then the assumptions of the theorem above are satisfied using* $Z = Z_1$ *and*

$$\omega(r) = \{\psi(r), \omega \circ \psi(r)\},$$

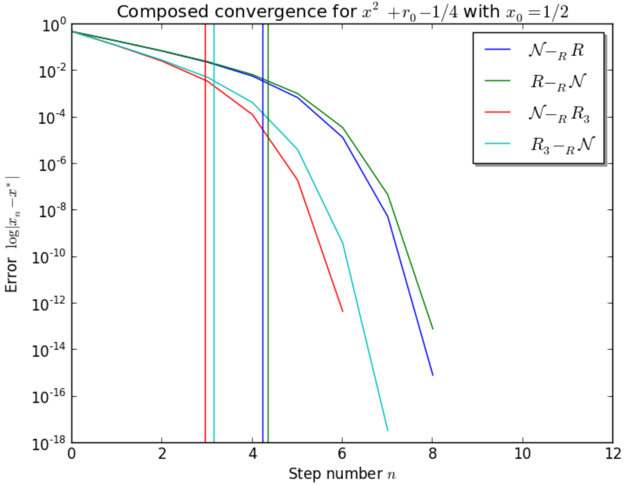
giving us the existence of a solution, and both a priori and a posteriori bounds on the error.

Example

$$f(x) = x^2 + (0.0894427)^2$$

n	$\|x_{n+1} - x_n\|$	$\|x_{n+1} - x_n\| - w^{(n)}(r_0)$	$\|x_n - x^*\| - s(w^{(n)}(r_0))$
0	1.9990e+00	$< 10^{-16}$	$< 10^{-16}$
1	9.9850e-01	$< 10^{-16}$	$< 10^{-16}$
2	4.9726e-01	$< 10^{-16}$	$< 10^{-16}$
3	2.4470e-01	$< 10^{-16}$	$< 10^{-16}$
4	1.1492e-01	$< 10^{-16}$	$< 10^{-16}$
5	4.5342e-02	$< 10^{-16}$	$< 10^{-16}$
6	1.0251e-02	$< 10^{-16}$	$< 10^{-16}$
7	5.8360e-04	$< 10^{-16}$	$< 10^{-16}$
8	1.9039e-06	$< 10^{-16}$	$< 10^{-16}$
9	2.0264e-11	$< 10^{-16}$	$< 10^{-16}$
10	0.0000e+00	$< 10^{-16}$	$< 10^{-16}$

Example

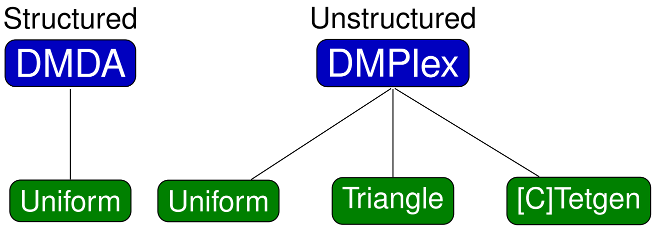


Matrix iterations also 1D scalar once you diagonalize

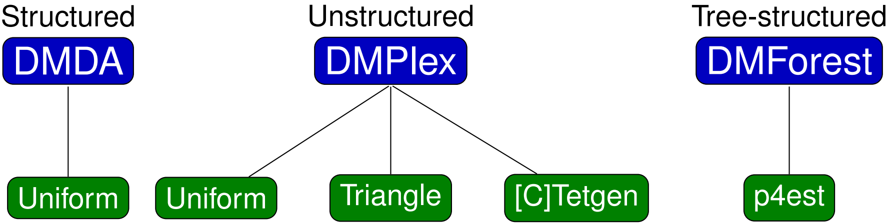
Pták's nondiscrete induction and its application to matrix iterations, Liesen, IMA J. Num. Anal.,

Outline

Mesh Refinement in PETSc



Mesh Refinement in PETSc

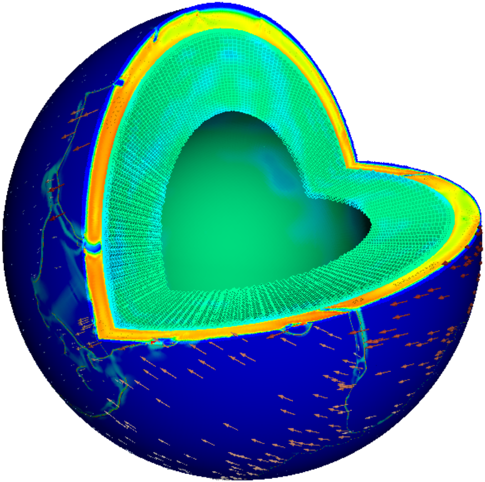


p4est overview

The p4est library (lead developer Carsten Burstedde) provides scalable AMR routines via a forest-of-octrees/quadtrees:

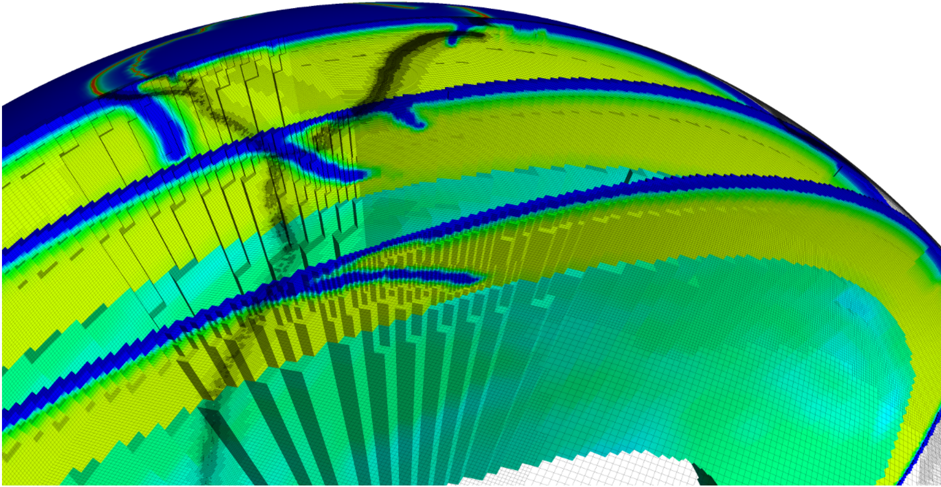
- a unstructured hexahedral mesh (“the forest”);
- where each hexahedron contains an arbitrarily refined octree;
- space-filling curve (SFC) orders elements;
- philosophy: as-simple-as-possible coarse mesh describes geometry, refinement captures all detail.
- not a framework: does not have numerical methods
  - Used for parallelism by Deal.II
  - Tight integration with solvers (e.g., multilevel) is still the domain of experts (next slide)

p4est in geophysics



(Rudi et al., 2015), "An extreme-scale implicit solver for complex PDEs: highly heterogeneous flow in earth's mantle," doi:10.1145/2807591.2807675.

p4est in geophysics



(Rudi et al., 2015), “An extreme-scale implicit solver for complex PDEs: highly heterogeneous flow in earth’s mantle,” doi:10.1145/2807591.2807675.

Integrating solvers with p4est (or any format)

- Comparison of numerical methods (discretizations (FE/FV), solvers (FAS/NASM/etc.)) requires implementation of numerical methods
- Implementations for exotic formats can be expensive (man hours) and error prone
- Numerical methods we are interested in are naturally described with operations on unstructured meshes (DMPlex)

```
DoSomething_p4est (DM dmp4est)
{
  if (dmp4est->ops->dosomething) { /* optimized implementation */
    (dmp4est->ops->dosomething) (dmp4est);
  } else {
    DM dmplex;

    DMConvert (dmp4est, &dmplex);
    DoSomething (dmplex);
    DMDestroy (&dmplex);
  }
}
```

Integrating solvers with p4est (or any format)

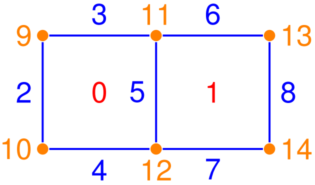
- Comparison of numerical methods (discretizations (FE/FV), solvers (FAS/NASM/etc.)) requires implementation of numerical methods
- Implementations for exotic formats can be expensive (man hours) and error prone
- Numerical methods we are interested in are naturally described with operations on unstructured meshes (DMPlex)

```
DoSomething_p4est (DM dmp4est)
{
  if (dmp4est->ops->dosomething) { /* optimized implementation */
    (dmp4est->ops->dosomething) (dmp4est);
  } else {
    DM dmplex;

    DMConvert (dmp4est, &dmplex);
    DoSomething (dmplex);
    DMDestroy (&dmplex);
  }
}
```

Nonconformal, Unstructured Meshes

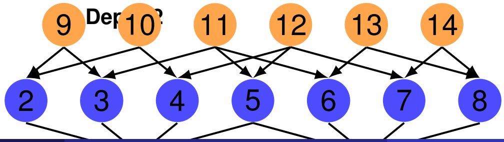
DMPlex is designed for conformal meshes. . .



Vertices **Depth 0**

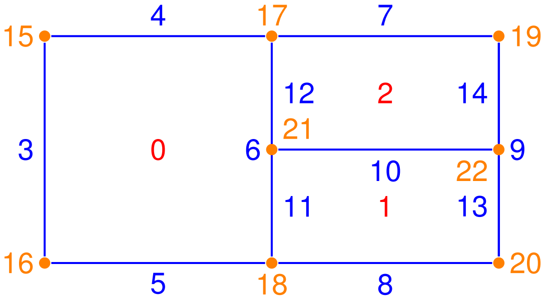
Edges **Depth 1**

Cells **Depth 2**



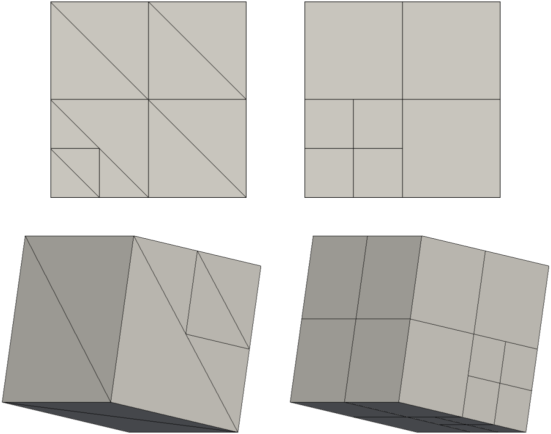
Nonconformal, Unstructured Meshes

. . . but now can handle nonconformal with parent/child relationships.



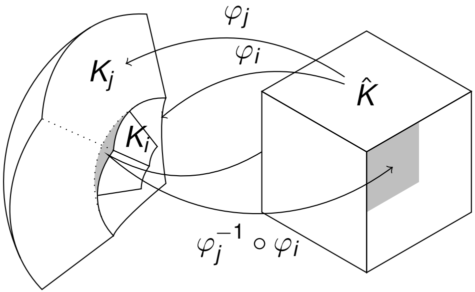
Nonconformal, Unstructured Meshes

. . . for arbitrary elements.



Hanging Nodes

Hanging-node continuity constraints are computed for arbitrary finite elements via dual-space functional mapping.



To enforce continuity at the interface (gray), each reference-cell functional σ_r for K_i is pushed-forward via $\varphi_j^{-1} \circ \varphi_i$ and evaluated at each shape function ψ_s for cell K_j, creating the constraint matrix C_{rs}.