The Process of Computational Science

Matthew Knepley

Computation Institute
University of Chicago

Department of Molecular Biology and Physiology
Rush University Medical Center

Applied Physics and Applied Mathematics
School of Engineering and Applied Science, Columbia University
New York, NY February 18, 2013

My approach to Computational Science is

Holistic

My approach to
Computational Science is

Holistic

Computational Science

starting with the numerics of PDEs,

and mathematics of the computation,

through the distillation into

high quality numerical libraries,

to scientific discovery through computing.

Computational Science

starting with the numerics of PDEs,

and mathematics of the computation,

through the distillation into

high quality numerical libraries,

to scientific discovery through computing.

Computational Science

starting with the numerics of PDEs,

and mathematics of the computation,

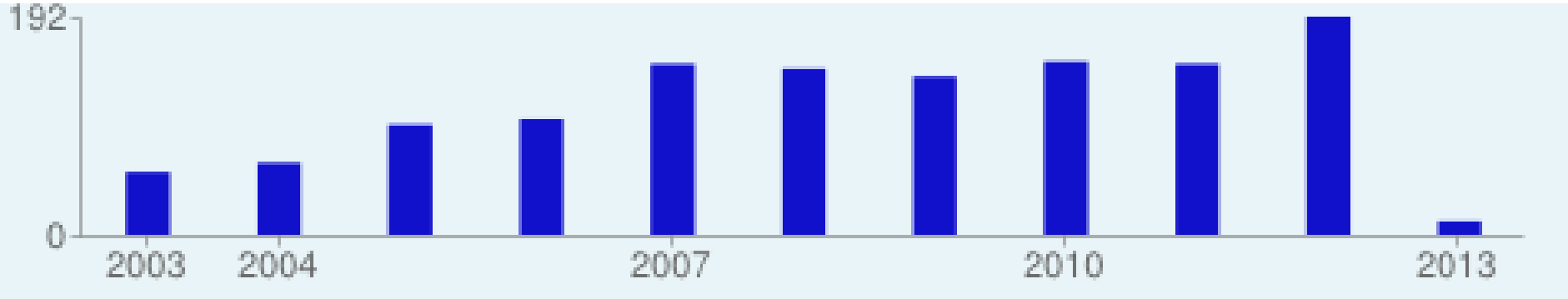
through the distillation into

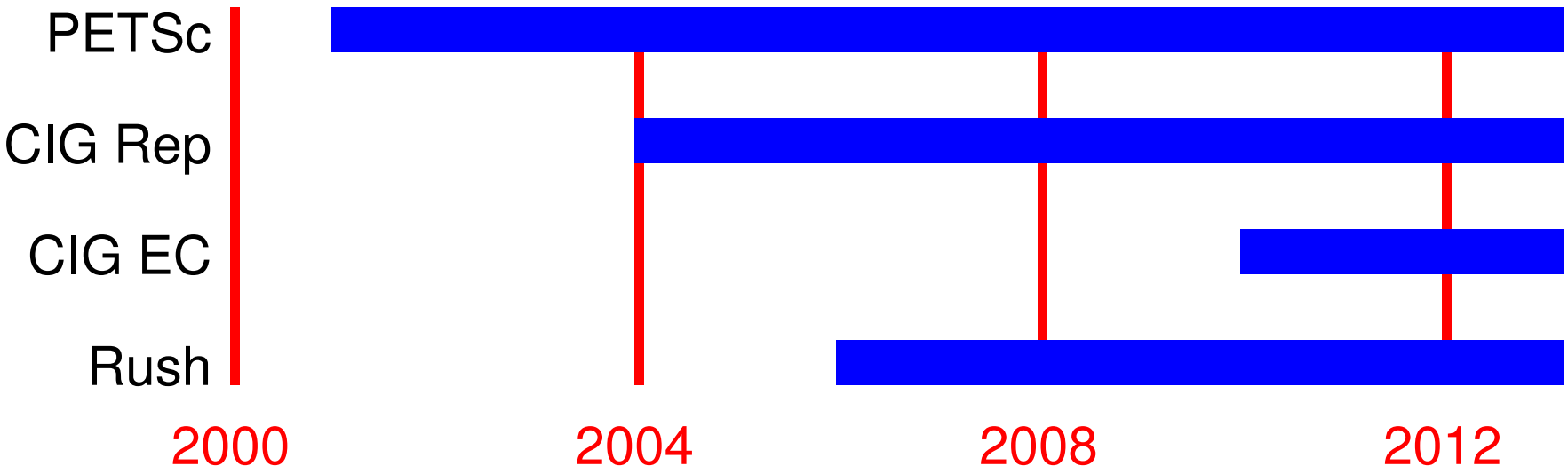
high quality numerical libraries,

to scientific discovery through computing.

Community Involvement

PETSc Citations





Outline

1 Operator Approximation

2 Residual Evaluation

3 Applications

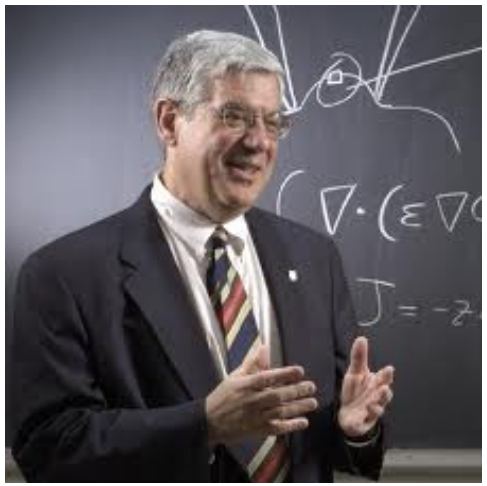
Collaborators

BIBEE Researchers

Jaydeep Bardhan

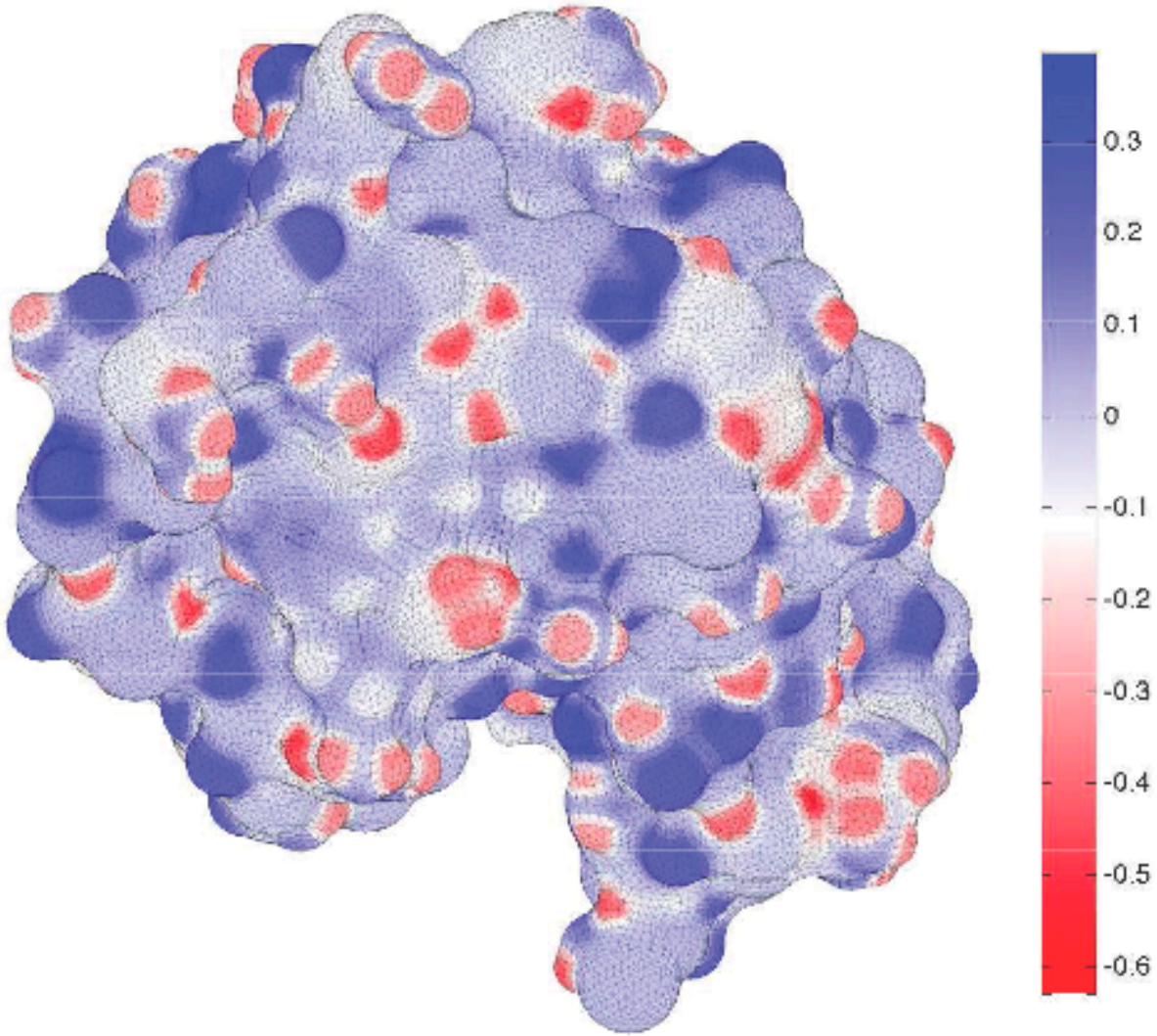
Classical DFT Researchers

Dirk Gillespie



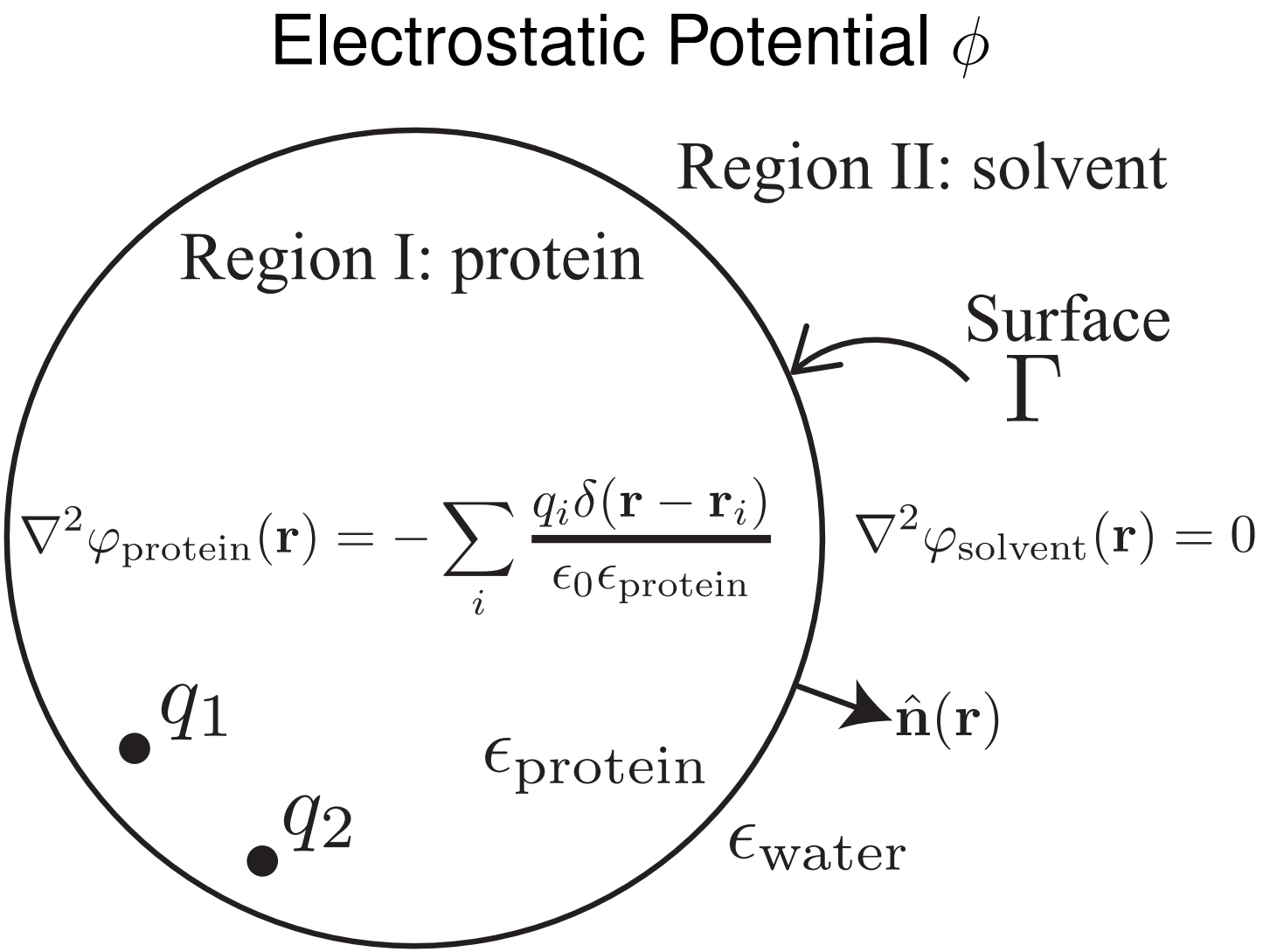
Bob Eisenberg

Bioelectrostatics
The Natural World



Induced Surface Charge on Lysozyme

Bioelectrostatics
Physical Model



Bioelectrostatics
Mathematical Model

We can write a Boundary Integral Equation (BIE) for the induced surface charge σ,

$$\sigma(\vec{r}) + \hat{\epsilon} \int_\Gamma \frac{\partial}{\partial n(\vec{r})} \frac{\sigma(\vec{r}')d^2\vec{r}'}{4\pi||\vec{r} - \vec{r}'||} = -\hat{\epsilon} \sum_{k=1}^{Q} \frac{\partial}{\partial n(\vec{r})} \frac{q_k}{4\pi||\vec{r} - \vec{r}_k||}$$

$$(\mathcal{I} + \hat{\epsilon}\mathcal{D}^*)\,\sigma(\vec{r}) =$$

where we define

$$\hat{\epsilon} = 2\frac{\epsilon_I - \epsilon_{II}}{\epsilon_I + \epsilon_{II}} < 0$$

Bioelectrostatics
Mathematical Model

The *reaction* potential is given by

$$\phi^R(\vec{r}) = \int_\Gamma \frac{\sigma(\vec{r}')d^2\vec{r}'}{4\pi\epsilon_1||\vec{r}-\vec{r}'||} = C\sigma$$

which defines G_{es}, the electrostatic part of the solvation free energy

$$\begin{aligned}\Delta G_{es} &= \frac{1}{2}\left\langle q, \phi^R \right\rangle \\ &= \frac{1}{2}\left\langle q, Lq \right\rangle \\ &= \frac{1}{2}\left\langle q, CA^{-1}Bq \right\rangle\end{aligned}$$

where

$$\begin{aligned}Bq &= -\hat{\epsilon}\int_\Omega \frac{\partial}{\partial n(\vec{r})}\frac{q(\vec{r}')d^3\vec{r}'}{4\pi||\vec{r}-\vec{r}'||} \\ A\sigma &= \mathcal{I} + \hat{\epsilon}\mathcal{D}^*\end{aligned}$$

Problem

Boundary element discretizations of the solvation problem (Eq. **??**):

- can be expensive to solve
- are more accurate than required by intermediate design iterations

BIBEE

Approximate $\mathcal{D}^*$ by a diagonal operator

Boundary Integral-Based Electrostatics Estimation

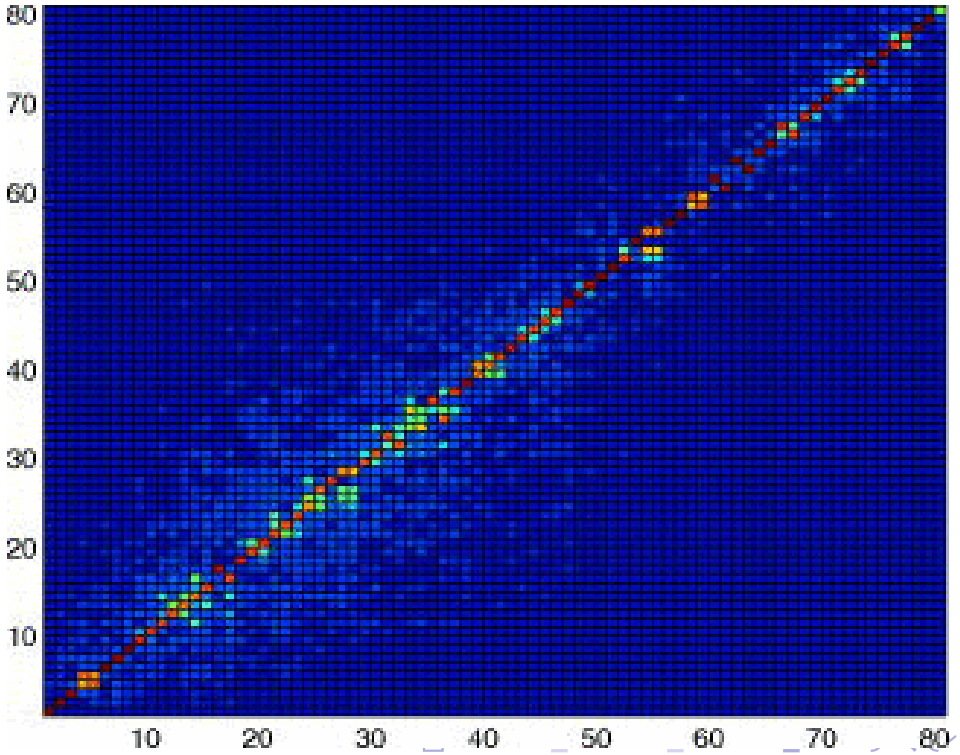
Coulomb Field Approximation:
uniform normal field

$$\left(1 - \frac{\hat{\epsilon}}{2}\right) \sigma_{CFA} = Bq$$

Lower Bound:
no good physical motivation

$$\left(1 + \frac{\hat{\epsilon}}{2}\right) \sigma_{LB} = Bq$$

Eigenvectors: BEM $e_i \cdot e_j$ BIBEE/P



BIBEE

Approximate $\mathcal{D}^*$ by a diagonal operator

Boundary Integral-Based Electrostatics Estimation

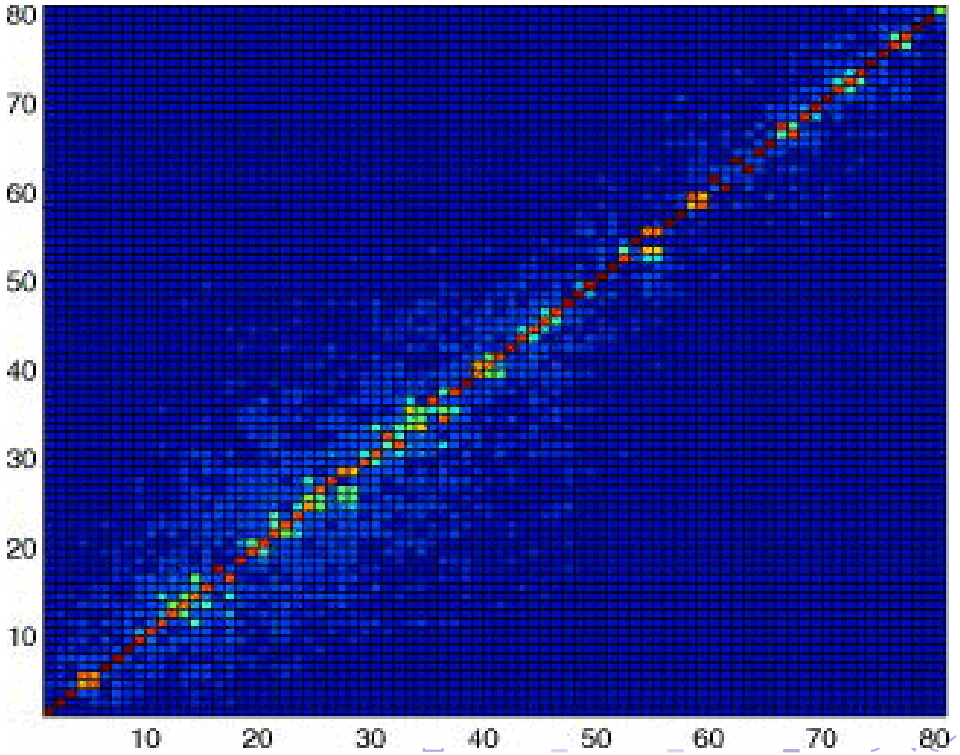
Coulomb Field Approximation:
uniform normal field

$$\left(1 - \frac{\hat{\epsilon}}{2}\right) \sigma_{CFA} = Bq$$

Preconditioning:
consider only local effects

$$\sigma_P = Bq$$

Eigenvectors: BEM $e_i \cdot e_j$ BIBEE/P



BIBEE Bounds on Solvation Energy

Theorem: The electrostatic solvation energy ΔG_{es} has upper and lower bounds given by

$$\frac{1}{2}\left(1+\frac{\hat{\epsilon}}{2}\right)^{-1}\langle q, CBq\rangle \leq \frac{1}{2}\left\langle q, CA^{-1}Bq\right\rangle \leq \frac{1}{2}\left(1-\frac{\hat{\epsilon}}{2}\right)^{-1}\langle q, CBq\rangle,$$

and for spheres and prolate spheroids, we have the improved lower bound,

$$\frac{1}{2}\langle q, CBq\rangle \leq \frac{1}{2}\left\langle q, CA^{-1}Bq\right\rangle,$$

and we note that

$$|\hat{\epsilon}| < \frac{1}{2}.$$

Energy Bounds:

Proof: Bardhan, Knepley, Anitescu, JCP, **130**(10), 2008

I will break the proof into three steps,

- Replace C with B
- Symmetrization
- Eigendecomposition

shown in the following slides.

We will need the single layer operator $\mathcal{S}$ for step 1,

$$\mathcal{S}\tau(\vec{r}) = \int \frac{\tau(\vec{r}')d^2\vec{r}'}{4\pi||\vec{r} - \vec{r}'||}$$

Energy Bounds: First Step

Replace C with B

The potential at the boundary Γ given by

$$\phi^{Coulomb}(\vec{r}) = C^T q$$

can also be obtained by solving an exterior Neumann problem for τ,

$$\begin{aligned}
\phi^{Coulomb}(\vec{r}) &= \mathcal{S}\tau \\
&= \mathcal{S}(\mathcal{I} - 2\mathcal{D}^*)^{-1}(\frac{2}{\hat{\epsilon}}Bq) \\
&= \frac{2}{\hat{\epsilon}}\mathcal{S}(\mathcal{I} - 2\mathcal{D}^*)^{-1}Bq
\end{aligned}$$

so that the solvation energy is given by

$$\frac{1}{2}\left\langle q, CA^{-1}Bq\right\rangle = \frac{1}{\hat{\epsilon}}\left\langle \mathcal{S}(\mathcal{I} - 2\mathcal{D}^*)^{-1}Bq, (\mathcal{I} + \hat{\epsilon}\mathcal{D}^*)^{-1}Bq\right\rangle$$

Energy Bounds: Second Step

Quasi-Hermiticity

Plemelj's symmetrization principle holds that

$$\mathcal{S}\mathcal{D}^* = \mathcal{D}\mathcal{S}$$

and we have

$$\mathcal{S} = \mathcal{S}^{1/2}\mathcal{S}^{1/2}$$

which means that we can define a Hermitian operator H similar to $\mathcal{D}^*$

$$H = \mathcal{S}^{1/2}\mathcal{D}^*\mathcal{S}^{-1/2}$$

leading to an energy

$$\frac{1}{2}\left\langle q, CA^{-1}Bq\right\rangle = \frac{1}{\hat{\epsilon}}\left\langle Bq, \mathcal{S}^{1/2}(\mathcal{I} - 2H)^{-1}(\mathcal{I} + \hat{\epsilon}H)^{-1}\mathcal{S}^{1/2}Bq\right\rangle$$

Energy Bounds: Third Step

Eigendecomposition

The spectrum of $\mathcal{D}^*$ is in $[-\frac{1}{2}, \frac{1}{2})$, and the energy is

$$\frac{1}{2}\left\langle q, CA^{-1}Bq\right\rangle = \sum_i \frac{1}{\hat{\epsilon}}\left(1-2\lambda_i\right)^{-1}\left(1+\hat{\epsilon}\lambda_i\right)^{-1}x_i^2$$

where

$$H = V\Lambda V^T$$

and

$$\vec{x} = V^T\mathcal{S}^{1/2}Bq$$

Energy Bounds: Diagonal Approximations

The BIBEE approximations yield the following bounds

$$\frac{1}{2}\left\langle q, CA_{CFA}^{-1}Bq\right\rangle = \sum_i \frac{1}{\hat{\epsilon}}\left(1-2\lambda_i\right)^{-1}\left(1-\frac{\hat{\epsilon}}{2}\right)^{-1}x_i^2$$

$$\frac{1}{2}\left\langle q, CA_{P}^{-1}Bq\right\rangle = \sum_i \frac{1}{\hat{\epsilon}}\left(1-2\lambda_i\right)^{-1}x_i^2$$

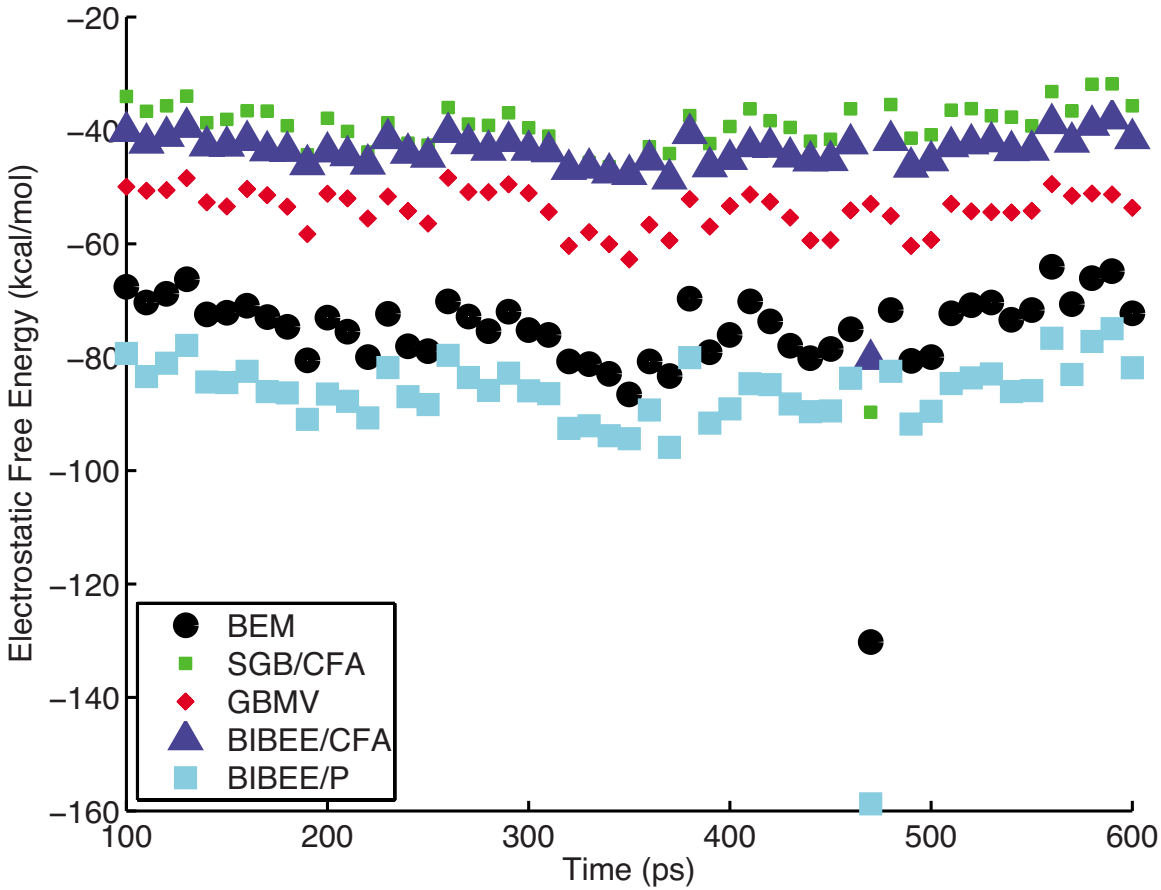
$$\frac{1}{2}\left\langle q, CA_{LB}^{-1}Bq\right\rangle = \sum_i \frac{1}{\hat{\epsilon}}\left(1-2\lambda_i\right)^{-1}\left(1+\frac{\hat{\epsilon}}{2}\right)^{-1}x_i^2$$

where we note that

$$|\hat{\epsilon}| < \frac{1}{2}$$

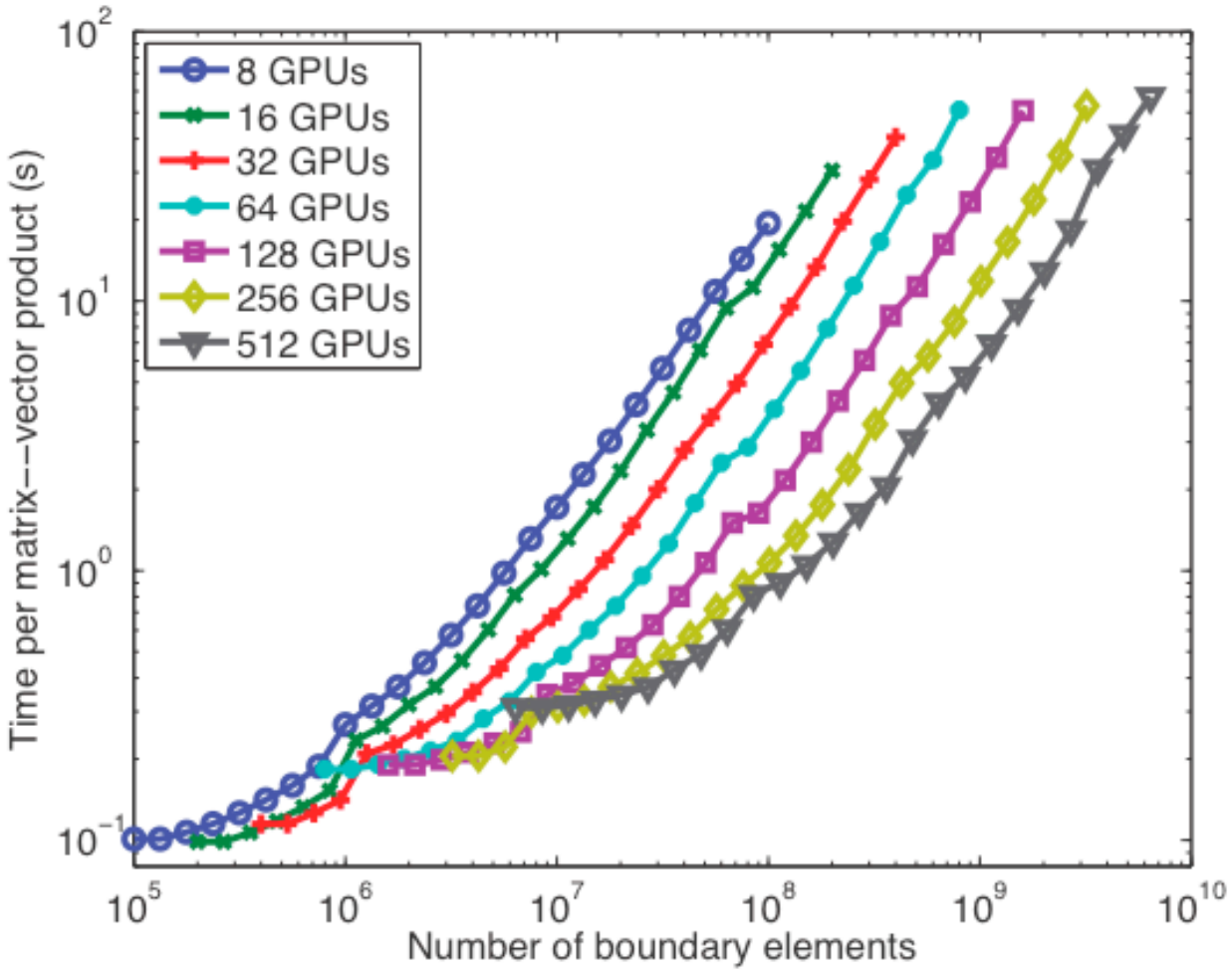
BIBEE Accuracy

Electrostatic solvation free energies of met-enkephalin structures



Snapshots taken from a 500-ps MD simulation at 10-ps intervals.
Bardhan, Knepley, Anitescu, JCP, 2009.

BIBEE Scalability



Yokota, Bardhan, Knepley, Barba, Hamada, CPC, 2011.

Resolution

Boundary element discretizations of the solvation problem:

- can be expensive to solve
  - **Bounding the electrostatic free energies associated with linear continuum models of molecular solvation**, JCP, 2009
- are more accurate than required by intermediate design iterations
  - Accuracy is not tunable

Evolution of BIBEE

- Sharp bounds for solvation energy
- Exploration of behavior in simplified geometries
  - **Mathematical Analysis of the BIBEE Approximation for Molecular Solvation: Exact Results for Spherical Inclusions**, JCP, 2011
  - Represent BIBEE as a deformed boundary condition
  - Fully developed series solution
  - Improve accuracy by combining CFA and P approximations
- Application to protein-ligand binding
  - **Analysis of fast boundary-integral approximations for modeling electrostatic contributions of molecular binding**, Molecular-Based Mathematical Biology, 2013

Future of BIBEE

- Framework for systematic exploration
  - Both analytical and computational foundation
- Reduced-basis Method with analytic solutions
  - Tested in protein binding paper above
  - The spatial high frequency part is handled by BIBEE/P topology is not important
  - The spatial low frequency part is handled by analytic solutions insensitive to bumpiness
  - **Computational science and re-discovery: open-source implementations of ellipsoidal harmonics for problems in potential theory**, CSD, 2012.
- Extend to other kernels, e.g. Yukawa
- Extend to full multilevel method

Outline

1 Operator Approximation

2 Residual Evaluation

3 Applications

Collaborators

PETSc Developers

Barry Smith

Jed Brown

Former UC Students

Andy Terrel

Peter Brune

Problem

Traditional PDE codes cannot:

- Compare different discretizations
  - Different orders, finite elements
  - finite volume vs. finite element
- Compare different mesh types
  - Simplicial, hexahedral, polyhedral
- Run 1D, 2D, and 3D problems
- Enable an optimal solver
  - Fields, auxiliary operators

Problem

Traditional Mesh/Solver Interface is Too General:

- Solver not told about discretization data, e.g. fields
- Cannot take advantage of problem structure
  - blocking
  - saddle point structure
- Cannot use auxiliary data
  - Eigen-estimates
  - null spaces

Problem

Traditional Mesh/Solver Interface is Too Specific:

- Assembly code specialized to each discretization
  - dimension
  - cell shape
  - approximation space
- Explicit references to element type
  - getVertices(faceID), getAdjacency(edgeID, VERTEX), getAdjacency(edgeID, dim = 0)
- No interface for transitive closure
  - Awkward nested loops to handle different dimensions

Mesh Representation

We represent each mesh as a Hasse Diagram:

- Can represent any CW complex
- Can be implemented as a Directed Acyclic Graph
- Reduces mesh information to a single *covering* relation
- Can discover dimension, since meshes are ranked posets

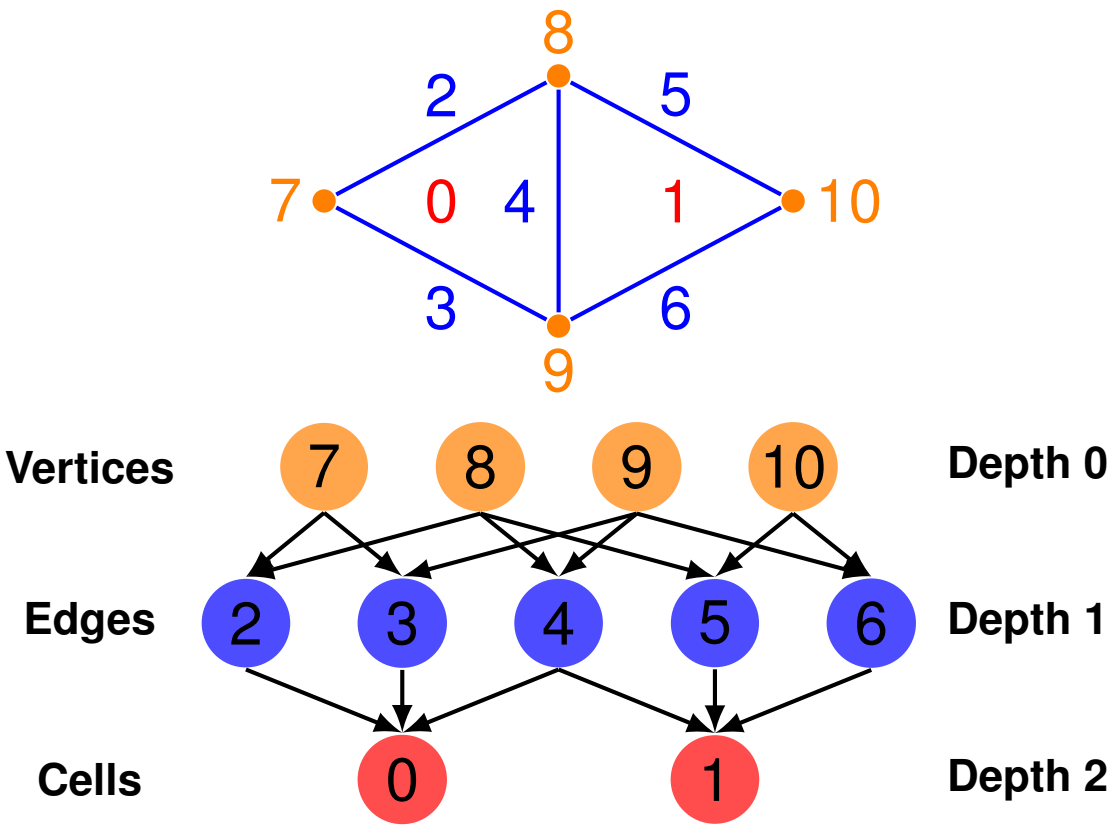
We use an abstract topological interface to organize traversals for:

- discretization integrals
- solver size determination
- computing communication patterns

Mesh geometry is treated as just another mesh function.

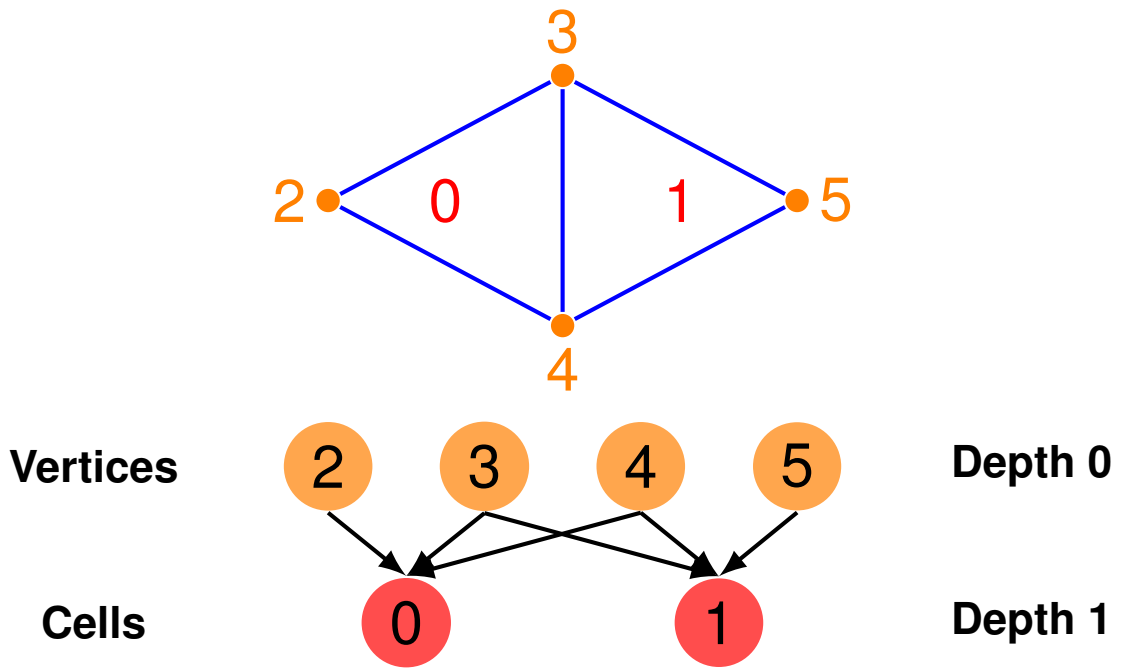
Sample Meshes

Interpolated triangular mesh



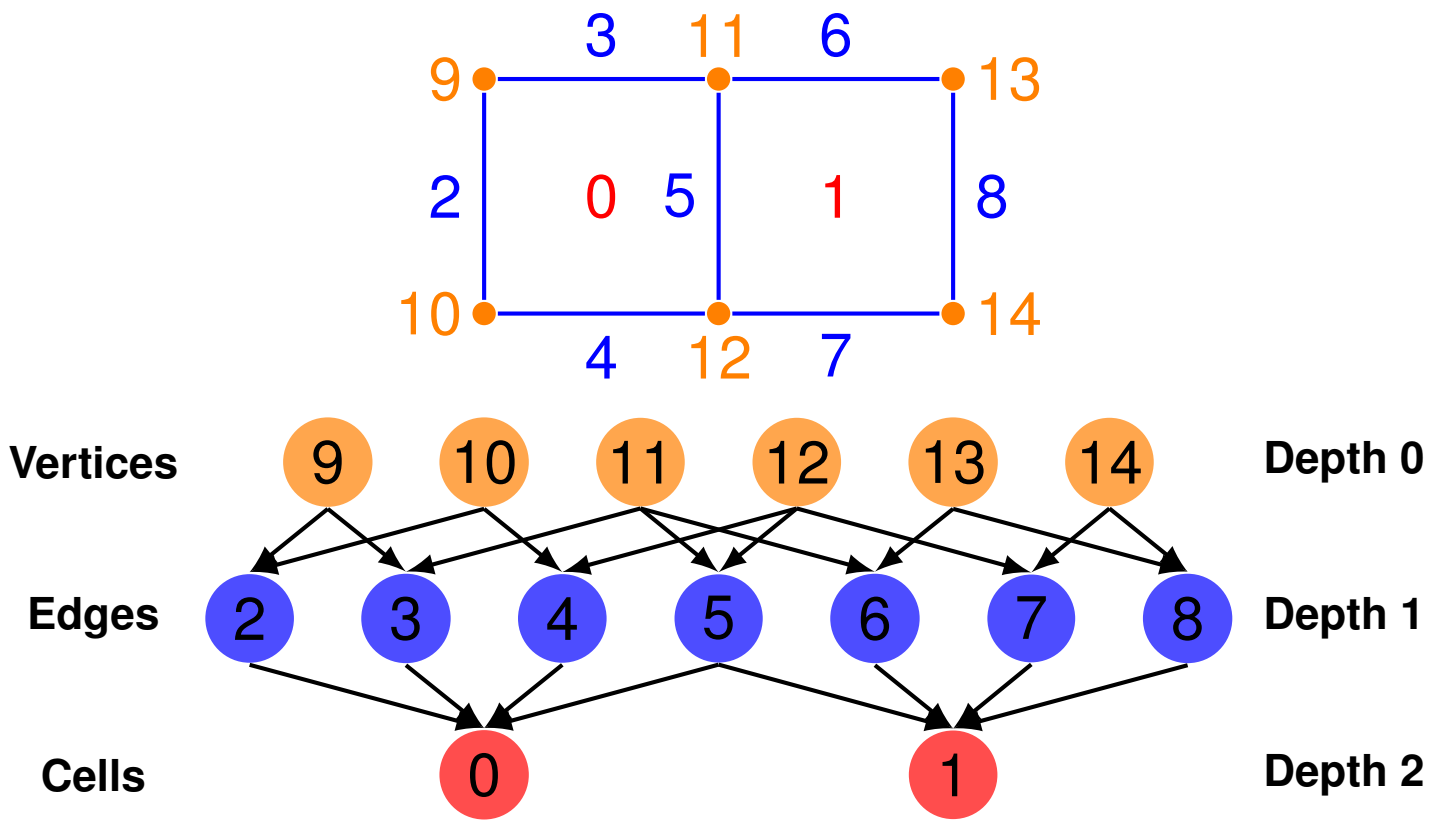
Sample Meshes

Optimized triangular mesh



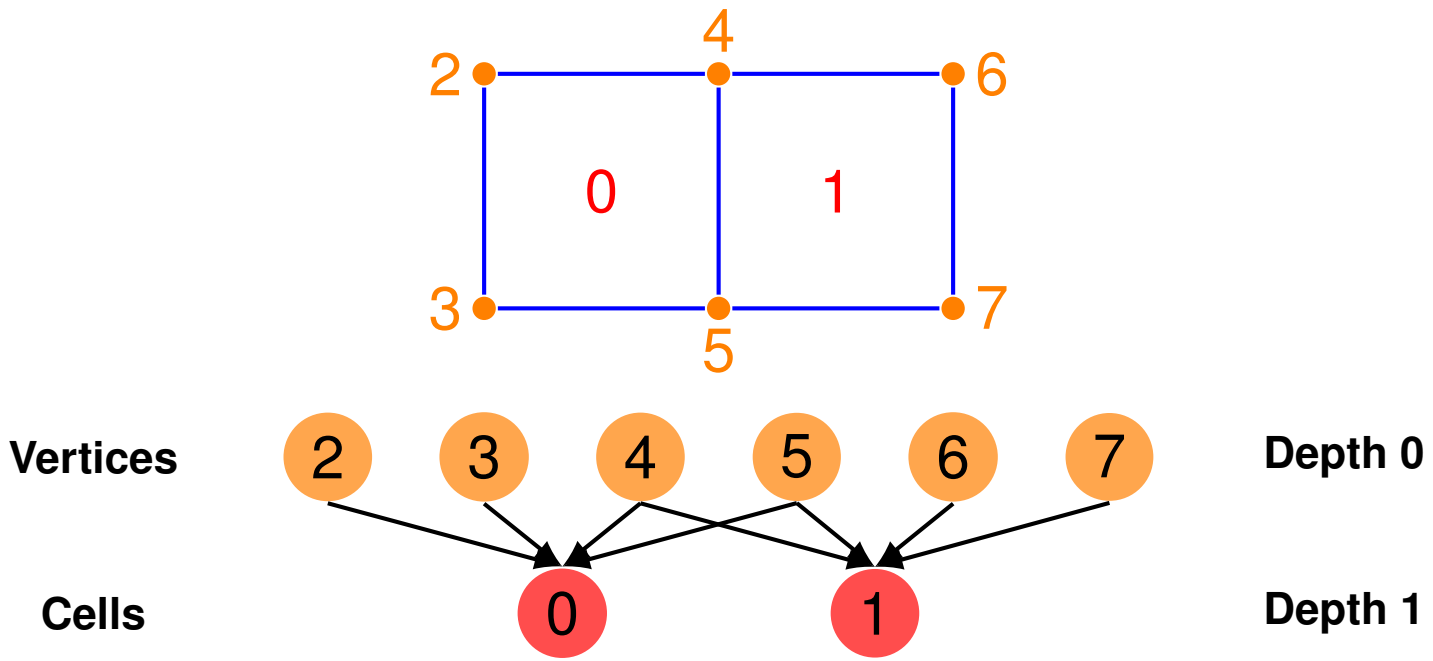
Sample Meshes

Interpolated quadrilateral mesh



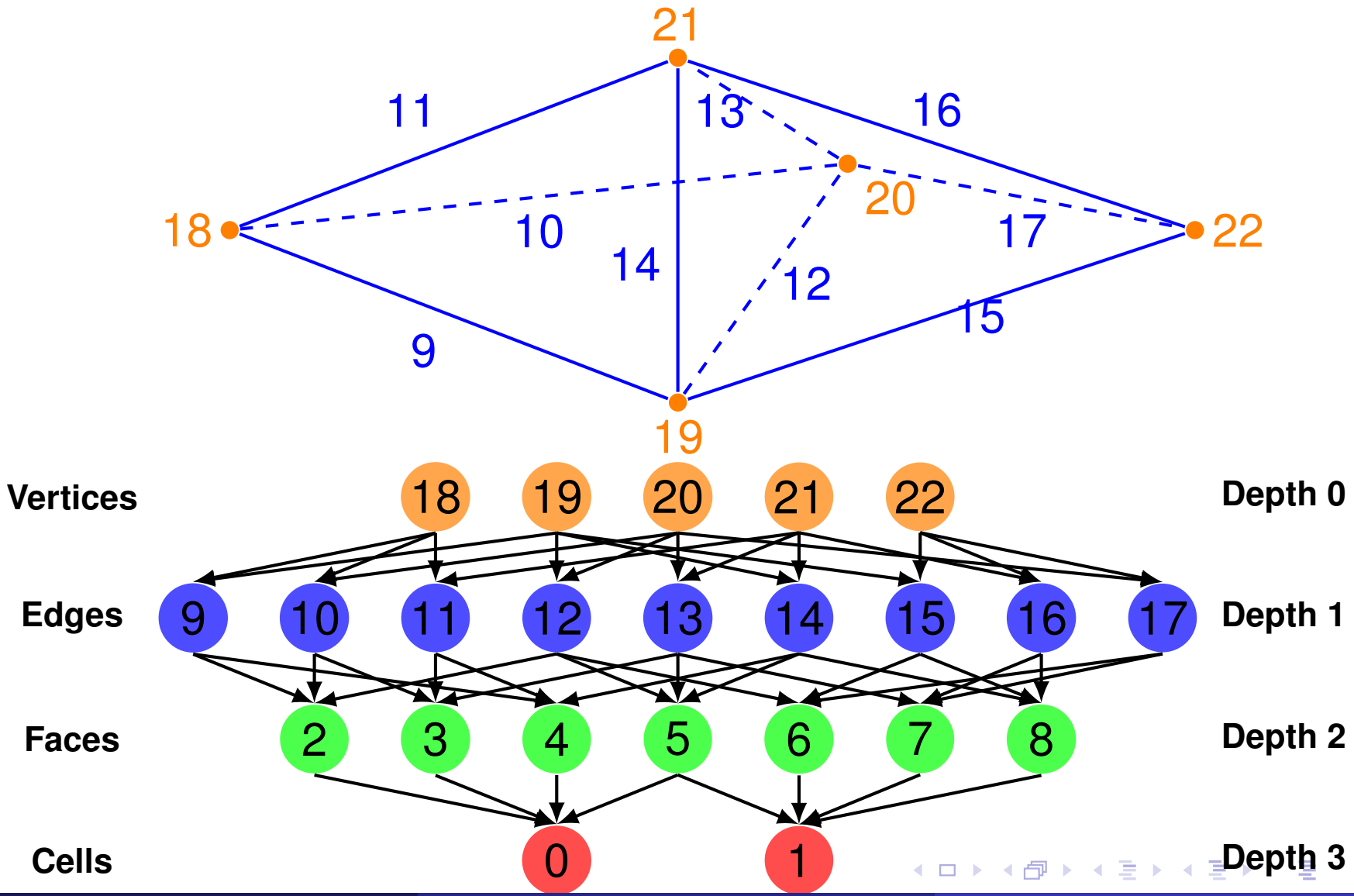
Sample Meshes

Optimized quadrilateral mesh



Sample Meshes

Interpolated tetrahedral mesh



Mesh Interface

By focusing on the key topological relations,
the interface can be both concise and quite general

- Single relation
- Dual is obtained by reversing arrows
- Can associate functions with DAG points
  - Dual operation gives the support of the function

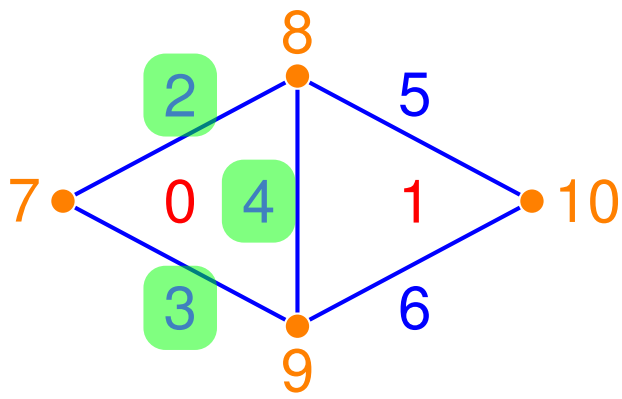
Mesh Algorithms for PDE with Sieve I: Mesh Distribution, Knepley, Karpeev, Sci. Prog., 2009.

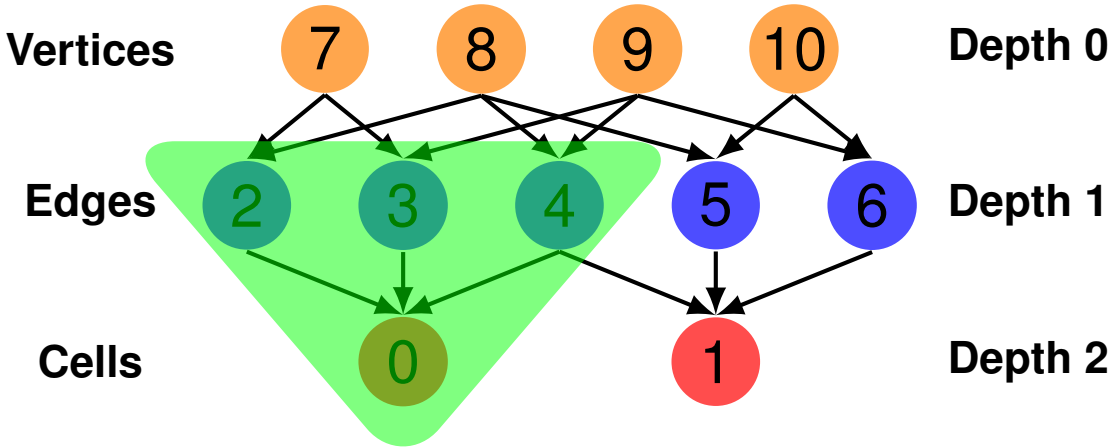
Basic Operations

Cone

We begin with the basic covering relation,

cone(0) = {2, 3, 4}

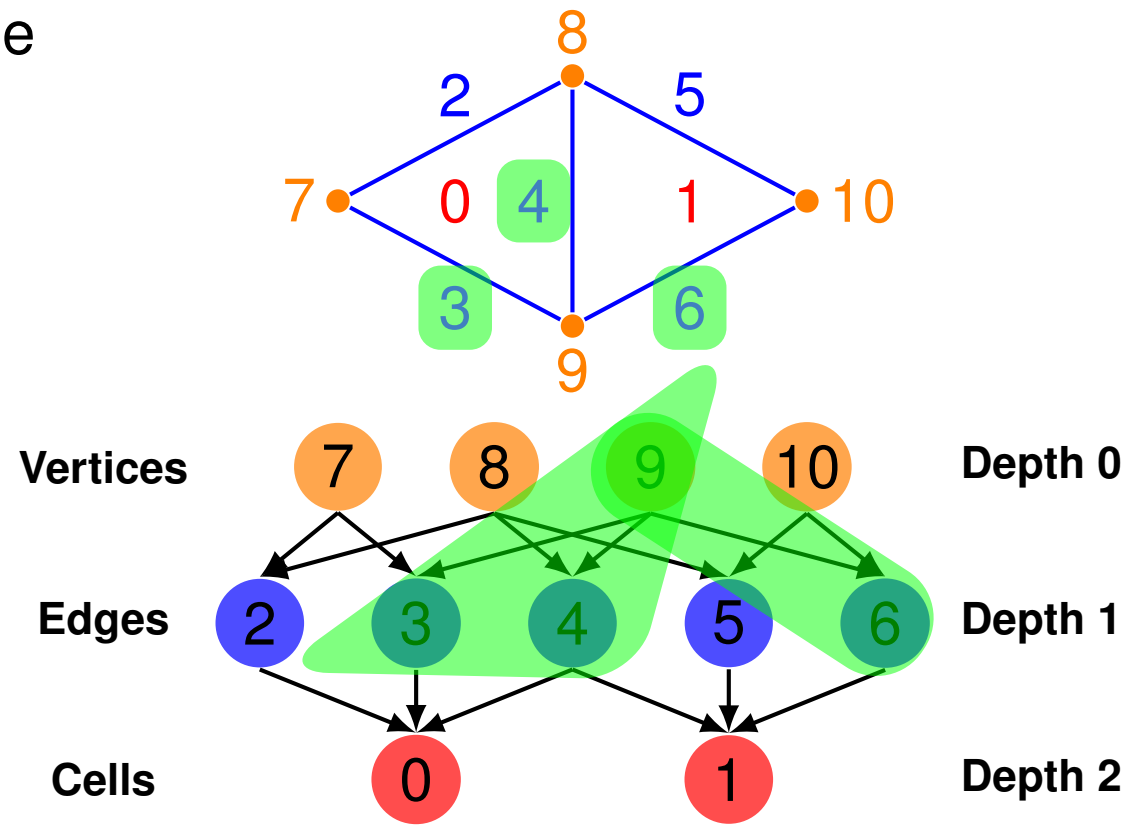




Basic Operations
Support

reverse arrows to get the dual operation,

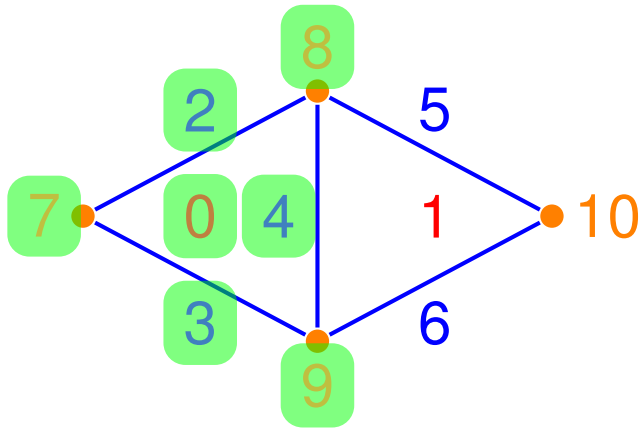
support(9) = {3, 4, 6}

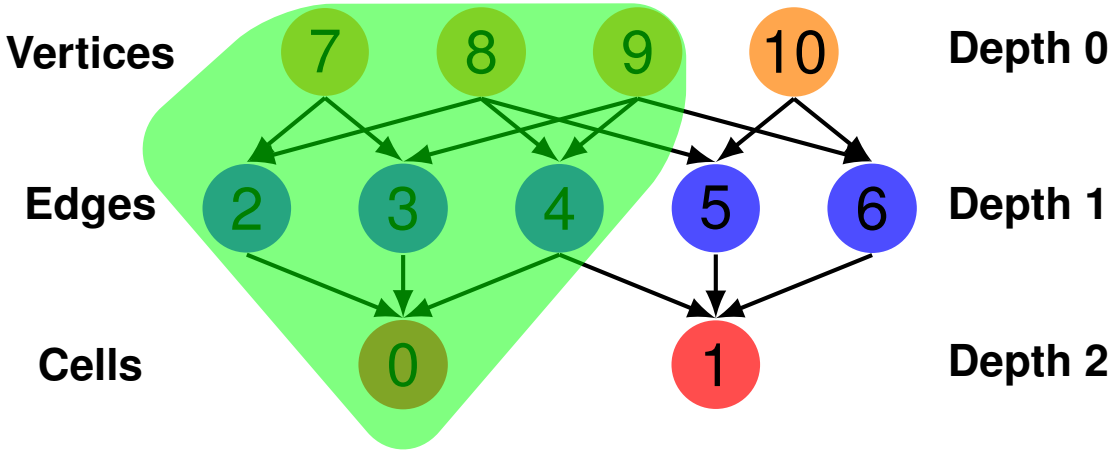


Basic Operations
Closure

add the transitive closure of the relation,

closure(0) = {0, 2, 3, 4, 7, 8, 9}



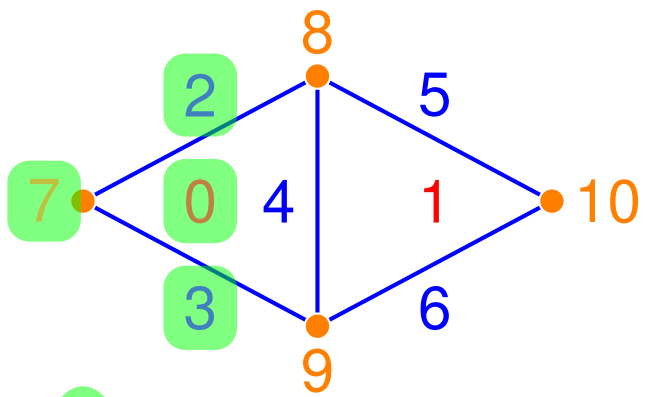


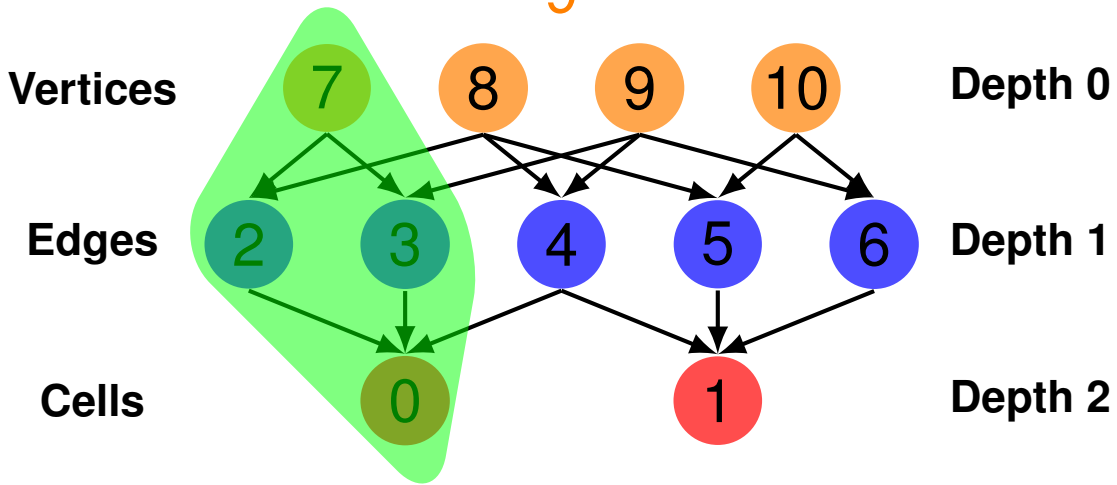
Basic Operations

Star

and the transitive closure of the dual,

star(7) = {7, 2, 3, 0}

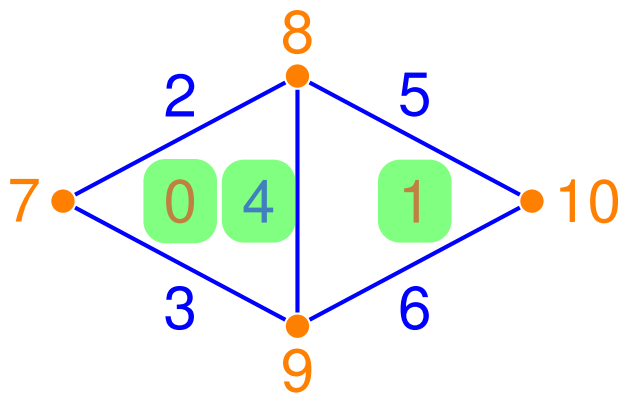


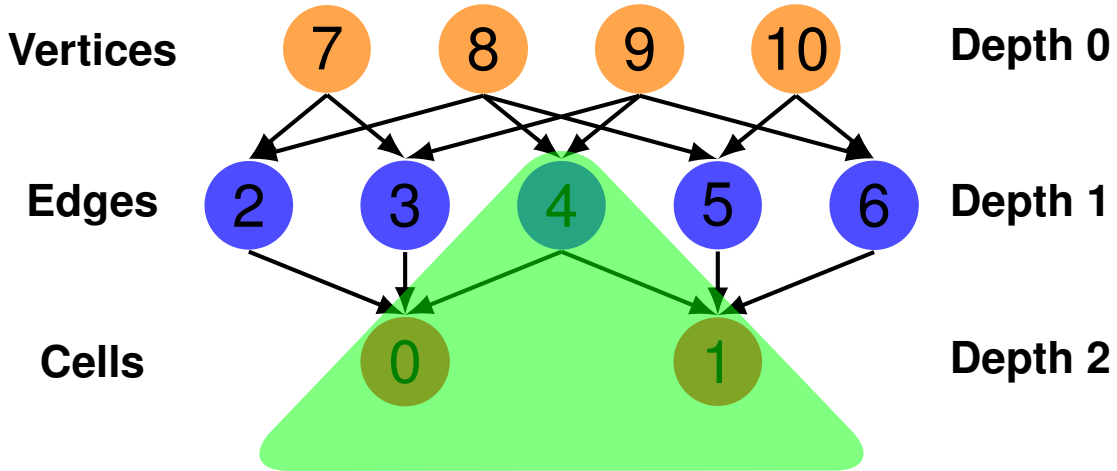


Basic Operations
Meet

and augment with lattice operations.

meet(0, 1) = {4}

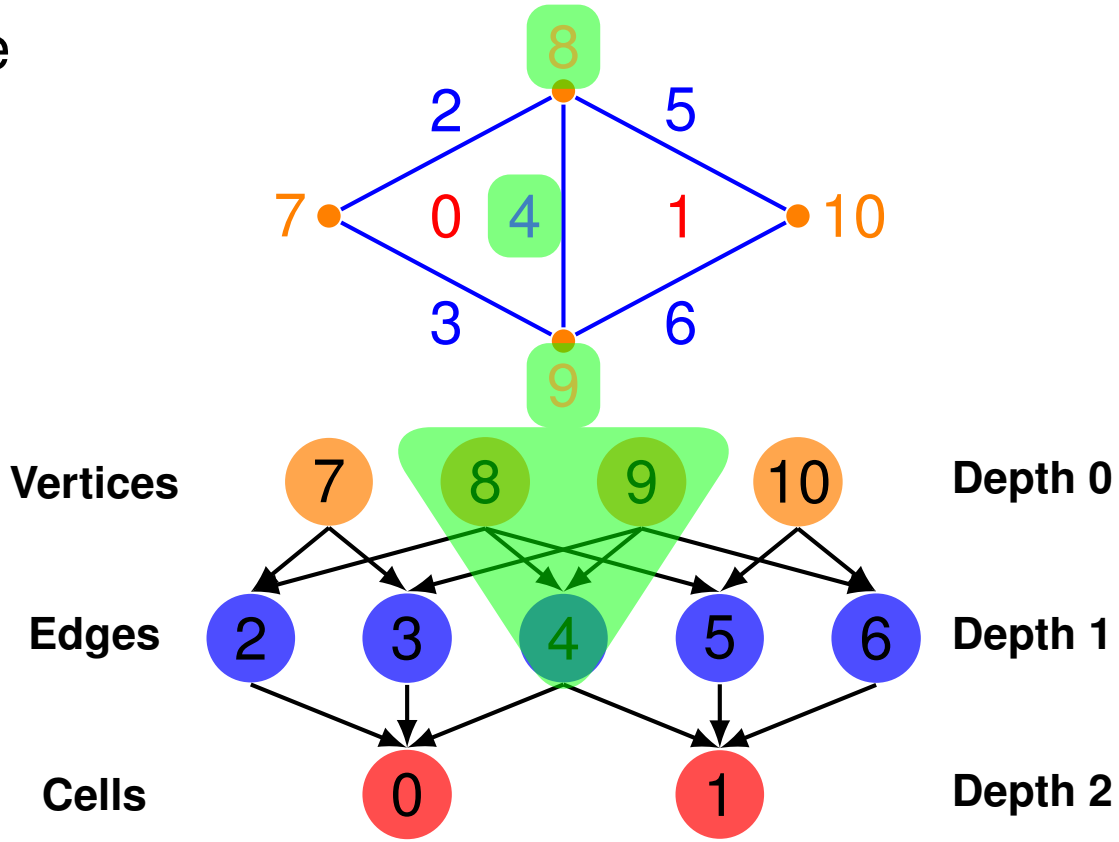




Basic Operations
Join

and augment with lattice operations.

join(8, 9) = {4}



Residual Evaluation

I developed a single residual evaluation routine independent of spatial dimension, cell geometry, and finite element:

$$F(\vec{u}) = 0$$

Dim	Cell Types	Discretizations
1	Simplex	Lagrange FEM
2	Tensor Product	H(div) FEM*
3	Polyhedral	H(curl) FEM*
6†	Prism	DG FEM *‡

† Peter Brune, ANL

* FEniCS Project

‡ Blaise Bourdin, LSU

We have also implemented a polyhedral FVM.

Residual Evaluation

I developed a single residual evaluation routine independent of spatial dimension, cell geometry, and finite element:

$$F(\vec{u}) = 0$$

Dim	Cell Types	Discretizations
1	Simplex	Lagrange FEM
2	Tensor Product	H(div) FEM*
3	Polyhedral	H(curl) FEM*
6†	Prism	DG FEM *‡

† Peter Brune, ANL

* FEniCS Project

‡ Blaise Bourdin, LSU

We have also implemented a polyhedral FVM.

Residual Evaluation

I developed a single residual evaluation routine independent of spatial dimension, cell geometry, and finite element:

$$F(\vec{u}) = 0$$

Dim
1
2
3
6†

Cell Types
Simplex
Tensor Product
Polyhedral
Prism

Discretizations
Lagrange FEM
H(div) FEM*
H(curl) FEM*
DG FEM *‡

† Peter Brune, ANL

* FEniCS Project

‡ Blaise Bourdin, LSU

We have also implemented a polyhedral FVM.

Residual Evaluation

I developed a single residual evaluation routine independent of spatial dimension, cell geometry, and finite element:

$$F(\vec{u}) = 0$$

Dim
1
2
3
6†

Cell Types
Simplex
Tensor Product
Polyhedral
Prism

Discretizations
Lagrange FEM
H(div) FEM*
H(curl) FEM*
DG FEM *‡

† Peter Brune, ANL

* FEniCS Project

‡ Blaise Bourdin, LSU

We have also implemented a polyhedral FVM.

Residual Evaluation

I developed a single residual evaluation routine independent of spatial dimension, cell geometry, and finite element:

$$F(\vec{u}) = 0$$

Dim
1
2
3
6†

Cell Types
Simplex
Tensor Product
Polyhedral
Prism

Discretizations
Lagrange FEM
H(div) FEM*
H(curl) FEM*
DG FEM *‡

† Peter Brune, ANL

* FEniCS Project

‡ Blaise Bourdin, LSU

We have also implemented a polyhedral FVM.

FEM Integration Model

Proposed by Jed Brown

We consider weak forms dependent only on fields and gradients,

$$\int_\Omega \phi \cdot f_0(u, \nabla u) + \nabla\phi : \vec{f}_1(u, \nabla u) = 0. \tag{1}$$

Discretizing we have

$$\sum_e \mathcal{E}_e^T \left[B^T W^q f_0(u^q, \nabla u^q) + \sum_k D_k^T W^q \vec{f}_1^k(u^q, \nabla u^q) \right] = 0 \tag{2}$$

f_n	pointwise physics functions
u^q	field at a quad point
W^q	diagonal matrix of quad weights
B,D	basis function matrices which reduce over quad points
$\mathcal{E}$	assembly operator

Batch Integration

```
DMPlexComputeResidualFEM(dm, X, F, user)
{
  VecSet(F, 0.0);
  <Put boundary conditions into local input vector>
  <Extract coefficients and geometry for batch>
  <Integrate batch of elements>
  <Insert batch of element vectors into global vector>
}
```

Batch Integration
Set boundary conditions

```
DMPlexComputeResidualFEM(dm, X, F, user)
{
  VecSet(F, 0.0);
  DMPlexProjectFunctionLocal(dm, numComponents,
    bcFuncs, INSERT_BC_VALUES, X);
  <Extract coefficients and geometry for batch>
  <Integrate batch of elements>
  <Insert batch of element vectors into global vector>
}
```

Batch Integration

Extract coefficients and geometry

```
DMPlexComputeResidualFEM(dm, X, F, user)
{
  VecSet(F, 0.0);
  <Put boundary conditions into local input vector>
  DMPlexGetHeightStratum(dm, 0, &cStart, &cEnd);
  for (c = cStart; c < cEnd; ++c) {
    DMPlexComputeCellGeometry(dm, c, &v0[c*dim],
      &J[c*dim*dim], &invJ[c*dim*dim], &detJ[c]);
    DMPlexVecGetClosure(dm, NULL, X, c, NULL, &x);
    for (i = 0; i < cellDof; ++i) u[c*cellDof+i] = x[i];
    DMPlexVecRestoreClosure(dm, NULL, X, c, NULL, &x);
  }
  <Integrate batch of elements>
  <Insert batch of element vectors into global vector>
}
```

Batch Integration

Integrate element batch

```
DMPlexComputeResidualFEM(dm, X, F, user)
{
  VecSet(F, 0.0);
  <Put boundary conditions into local input vector>
  <Extract coefficients and geometry for batch>
  for (field = 0; field < numFields; ++field) {
    (*mesh->integrateResidualFEM)(Ne, numFields, field,
      quad, u,
      v0, J, invJ, detJ,
      f0, f1, elemVec);
    (*mesh->integrateResidualFEM)(Nr, ...);
  }
  <Insert batch of element vectors into global vector>
}
```

Batch Integration

Insert element vectors

```
DMPlexComputeResidualFEM(dm, X, F, user)
{
  VecSet(F, 0.0);
  <Put boundary conditions into local input vector>
  <Extract coefficients and geometry for batch>
  <Integrate batch of elements>
   for (c = cStart; c < cEnd; ++c) {
     DMPlexVecSetClosure(dm, NULL, F, c,
       &elemVec[c*cellDof], ADD_VALUES);
  }
}
```

Element Integration

```
FEMIntegrateResidualBatch(Ne, numFields, field,
  quad[], coefficients[],
  v0s[], jacobians[], jacobianInv[], jacobianDet[],
  f0_func, f1_func)
{
  <Loop over batch of elements (e)>
    <Loop over quadrature points (q)>
      <Make x_q>
      <Make u_q and gradU_q>
      <Call f_0 and f_1>
    <Loop over element vector entries (f, fc)>
      <Add contributions from f_0 and f_1>
}
```

Element Integration

Calculate x_q

```
FEMIntegrateResidualBatch(...)
{
  <Loop over batch of elements (e)>
    <Loop over quadrature points (q)>
      for (d = 0; d < dim; ++d) {
        x[d] = v0[d];
        for (d2 = 0; d2 < dim; ++d2) {
          x[d] += J[d*dim+d2]*(quadPoints[q*dim+d2]+1);
        }
      }
      <Make x_q>
      <Make u_q and gradU_q>
      <Call f_0 and f_1>
    <Loop over element vector entries (f, fc)>
      <Add contributions from f_0 and f_1>
}
```

Element Integration

Calculate u_q and ∇u_q

```
FEMIntegrateResidualBatch(...)
{
  <Loop over batch of elements (e)>
    <Loop over quadrature points (q)>
      <Make x_q>
      for (f = 0; f < numFields; ++f) {
        for (b = 0; b < Nb; ++b) {
          for (comp = 0; comp < Ncomp; ++comp) {
            u[comp] += coefficients[cidx]*basis[q+cidx];
            for (d = 0; d < dim; ++d) {
              <Transform derivative to real space>
              gradU[comp*dim+d] +=
                coefficients[cidx]*realSpaceDer[d];
            }
          }
        }
      }
      <Call f_0 and f_1>
    <Loop over element vector entries (f, fc)>
```

Element Integration

Calculate u_q and ∇u_q

```
FEMIntegrateResidualBatch(...)
{
  <Loop over batch of elements (e)>
    <Loop over quadrature points (q)>
      <Make x_q>
      for (f = 0; f < numFields; ++f) {
        for (b = 0; b < Nb; ++b) {
          for (comp = 0; comp < Ncomp; ++comp) {
            u[comp] += coefficients[cidx]*basis[q+cidx];
            for (d = 0; d < dim; ++d) {
              realSpaceDer[d] = 0.0;
              for (g = 0; g < dim; ++g) {
                realSpaceDer[d] +=
                  invJ[g*dim+d]*basisDer[(q+cidx)*dim+g];
              }
              gradU[comp*dim+d] +=
                coefficients[cidx]*realSpaceDer[d];
            }
          }
```

Element Integration

Call f_0 and f_1

```
FEMIntegrateResidualBatch(...)
{
  <Loop over batch of elements (e)>
    <Loop over quadrature points (q)>
      <Make x_q>
      <Make u_q and gradU_q>
      f0_func(u, gradU, x, &f0[q*Ncomp]);
      for (i = 0; i < Ncomp; ++i) {
        f0[q*Ncomp+i] *= detJ*quadWeights[q];
      }
      f1_func(u, gradU, x, &f1[q*Ncomp*dim]);
      for (i = 0; i < Ncomp*dim; ++i) {
        f1[q*Ncomp*dim+i] *= detJ*quadWeights[q];
      }
    <Loop over element vector entries (f, fc)>
      <Add contributions from f_0 and f_1>
}
```

Element Integration
Update element vector

```
FEMIntegrateResidualBatch(...)
{
  <Loop over batch of elements (e)>
    <Loop over quadrature points (q)>
      <Make x_q>
      <Make u_q and gradU_q>
      <Call f_0 and f_1>
    <Loop over element vector entries (f, fc)>
      for (q = 0; q < Nq; ++q) {
        elemVec[cidx] += basis[q+cidx]*f0[q+comp];
        for (d = 0; d < dim; ++d) {
          <Transform derivative to real space>
          elemVec[cidx] +=
            realSpaceDer[d]*f1[(q+comp)*dim+d];
        }
      }
}
```

GPU Integration

Porting to the GPU meant changing only the element integration functior

- Has the same flexibility as CPU version
- Multiple threads execute each cell integral
- Achieves 100 GF/s for 2D P_1 Laplacian
- Code is available here
- Finite Element Integration on GPUs, TOMS, 2013
- **Finite Element Integration with Quadrature on the GPU**, PLC, 2013

Solver Integration: No New Code

ex62: P_2/P_1 Stokes Problem on Unstructured Mesh

Full Schur Complement

```
-ksp_type fgmres -pc_type fieldsplit -pc_fieldsplit_type schur
-pc_fieldsplit_schur_factorization_type full
  -fieldsplit_velocity_ksp_type gmres -fieldsplit_velocity_pc_type lu
  -fieldsplit_pressure_ksp_rtol 1e-10 -fieldsplit_pressure_pc_type jacobi
```

$$\begin{pmatrix} I & 0 \\ B^T A^{-1} & I \end{pmatrix} \begin{pmatrix} \hat{A} & 0 \\ 0 & \hat{S} \end{pmatrix} \begin{pmatrix} I & A^{-1}B \\ 0 & I \end{pmatrix}$$

Solver Integration: No New Code

ex62: P_2/P_1 Stokes Problem on Unstructured Mesh

SIMPLE

```
-ksp_type fgmres -pc_type fieldsplit -pc_fieldsplit_type schur
-pc_fieldsplit_schur_factorization_type full
  -fieldsplit_velocity_ksp_type gmres -fieldsplit_velocity_pc_type lu
  -fieldsplit_pressure_ksp_rtol 1e-10 -fieldsplit_pressure_pc_type jacobi
  -fieldsplit_pressure_inner_ksp_type preonly
    -fieldsplit_pressure_inner_pc_type jacobi
  -fieldsplit_pressure_upper_ksp_type preonly
    -fieldsplit_pressure_upper_pc_type jacobi
```

$$\begin{pmatrix} I & 0 \\ B^T D_A^{-1} & I \end{pmatrix} \begin{pmatrix} \hat{A} & 0 \\ 0 & \hat{S} \end{pmatrix} \begin{pmatrix} I & D_A^{-1} B \\ 0 & I \end{pmatrix}$$

Solver Integration: No New Code

ex31: P_2/P_1 Stokes Problem with Temperature on Unstructured Mesh

Additive Schwarz + Full Schur Complement

```
-ksp_type fgmres -pc_type fieldsplit -pc_fieldsplit_0_fields 0,1
-pc_fieldsplit_1_fields 2 -pc_fieldsplit_type additive
  -fieldsplit_0_ksp_type fgmres -fieldsplit_0_pc_type fieldsplit
  -fieldsplit_0_pc_fieldsplit_type schur
  -fieldsplit_0_pc_fieldsplit_schur_factorization_type full
    -fieldsplit_0_fieldsplit_velocity_ksp_type preonly
    -fieldsplit_0_fieldsplit_velocity_pc_type lu
    -fieldsplit_0_fieldsplit_pressure_ksp_rtol 1e-10
    -fieldsplit_0_fieldsplit_pressure_pc_type jacobi
  -fieldsplit_temperature_ksp_type preonly
  -fieldsplit_temperature_pc_type lu
```

$$\begin{pmatrix} \begin{pmatrix} I & 0 \\ B^T A^{-1} & I \end{pmatrix} \begin{pmatrix} \hat{A} & 0 \\ 0 & \hat{S} \end{pmatrix} \begin{pmatrix} I & A^{-1}B \\ 0 & I \end{pmatrix} & 0 \\ 0 & L_T \end{pmatrix}$$

Solver Integration: No New Code

ex31: P_2/P_1 Stokes Problem with Temperature on Unstructured Mesh

Least-Squares Commutator + Upper Schur Comp. + Full Schur Comp.

```
-ksp_type fgmres -pc_type fieldsplit -pc_fieldsplit_0_fields 0,1
-pc_fieldsplit_1_fields 2 -pc_fieldsplit_type schur
-pc_fieldsplit_schur_factorization_type upper
  -fieldsplit_0_ksp_type fgmres -fieldsplit_0_pc_type fieldsplit
  -fieldsplit_0_pc_fieldsplit_type schur
  -fieldsplit_0_pc_fieldsplit_schur_factorization_type full
    -fieldsplit_0_fieldsplit_velocity_ksp_type preonly
    -fieldsplit_0_fieldsplit_velocity_pc_type lu
    -fieldsplit_0_fieldsplit_pressure_ksp_rtol 1e-10
    -fieldsplit_0_fieldsplit_pressure_pc_type jacobi
  -fieldsplit_temperature_ksp_type gmres
  -fieldsplit_temperature_pc_type lsc
```

$$\begin{pmatrix} \begin{pmatrix} I & 0 \\ B^T A^{-1} & I \end{pmatrix} \begin{pmatrix} \hat{A} & 0 \\ 0 & \hat{S} \end{pmatrix} \begin{pmatrix} I & A^{-1}B \\ 0 & I \end{pmatrix} & G \\ 0 & \hat{S}_{\mathrm{LSC}} \end{pmatrix}$$

Resolution

Traditional PDE codes:

- Cannot compare different discretizations
  - **Automated FEM Discretizations for the Stokes Equation**, Terrel, et.al., BIT, 2008
  - Efficient Assembly of H(div) and H(curl) Conforming Finite Elements, Rognes, et.al., SISC, 2009
- Compare different mesh types
  - **A Domain Decomposition Approach to Implementing Fault Slip in Finite-Element Models of Quasi-static and Dynamic Crustal Deformation**, Aagaard, Knepley, Williams, JGR, 2013
- Run 1D, 2D, and 3D problems
  - **Ibid.**
- Enabling an optimal solver without programming
  - **Ibid.**
  - **Composable linear solvers for multiphysics**, Brown, et.al., IPDPS, 2012
  - On the rise of strongly tilted mantle plume tails, Mériaux, PEPI, 2011

Resolution

Traditional PDE codes:

- Cannot compare different discretizations
  - **Automated FEM Discretizations for the Stokes Equation**, Terrel, et.al., BIT, 2008
  - Efficient Assembly of H(div) and H(curl) Conforming Finite Elements, Rognes, et.al., SISC, 2009
- Compare different mesh types
  - **A Domain Decomposition Approach to Implementing Fault Slip in Finite-Element Models of Quasi-static and Dynamic Crustal Deformation**, Aagaard, Knepley, Williams, JGR, 2013
- Run 1D, 2D, and 3D problems
  - **Ibid.**
- Enabling an optimal solver without programming
  - **Ibid.**
  - **Composable linear solvers for multiphysics**, Brown, et.al., IPDPS, 2012
  - On the rise of strongly tilted mantle plume tails, Mériaux, PEPI, 2011

Resolution

Traditional PDE codes:

- Cannot compare different discretizations
  - **Automated FEM Discretizations for the Stokes Equation**, Terrel, et.al., BIT, 2008
  - Efficient Assembly of H(div) and H(curl) Conforming Finite Elements, Rognes, et.al., SISC, 2009
- Compare different mesh types
  - **A Domain Decomposition Approach to Implementing Fault Slip in Finite-Element Models of Quasi-static and Dynamic Crustal Deformation**, Aagaard, Knepley, Williams, JGR, 2013
- Run 1D, 2D, and 3D problems
  - **Ibid.**
- Enabling an optimal solver without programming
  - **Ibid.**
  - **Composable linear solvers for multiphysics**, Brown, et.al., IPDPS, 2012
  - On the rise of strongly tilted mantle plume tails, Mériaux, PEPI, 2011

Resolution

Traditional PDE codes:

- Cannot compare different discretizations
  - **Automated FEM Discretizations for the Stokes Equation**, Terrel, et.al., BIT, 2008
  - Efficient Assembly of H(div) and H(curl) Conforming Finite Elements, Rognes, et.al., SISC, 2009
- Compare different mesh types
  - **A Domain Decomposition Approach to Implementing Fault Slip in Finite-Element Models of Quasi-static and Dynamic Crustal Deformation**, Aagaard, Knepley, Williams, JGR, 2013
- Run 1D, 2D, and 3D problems
  - **Ibid.**
- Enabling an optimal solver without programming
  - **Ibid.**
  - **Composable linear solvers for multiphysics**, Brown, et.al., IPDPS, 2012
  - On the rise of strongly tilted mantle plume tails, Mériaux, PEPI, 2011

Future Work

- Unify FEM and FVM residual evaulation
- Batched integration on accelerators
- Integrate auxiliary fields
- Incorporate cell problems for coefficients

Outline

1 Operator Approximation

2 Residual Evaluation

3 Applications

PyLith

PyLith is an open source, parallel simulator for crustal deformation problems developed by myself, Brad Aagaard, and Charles Williams.

Brad Aagaard

Charles Williams

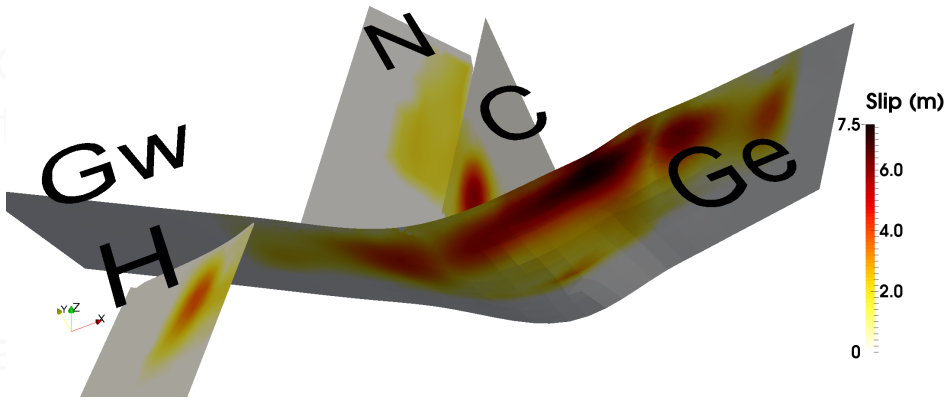
PyLith employs a finite element discretization on unstructured meshes and is built on the PETSc libraries from ANL.

PyLith

- Multiple problems
  - Dynamic rupture
  - Quasi-static relaxation
- Multiple models
  - Fault constitutive models
  - Nonlinear visco-elastic-plastic
  - Finite deformation
- Multiple Meshes
  - 1D, 2D, 3D
  - Hex and tet meshes

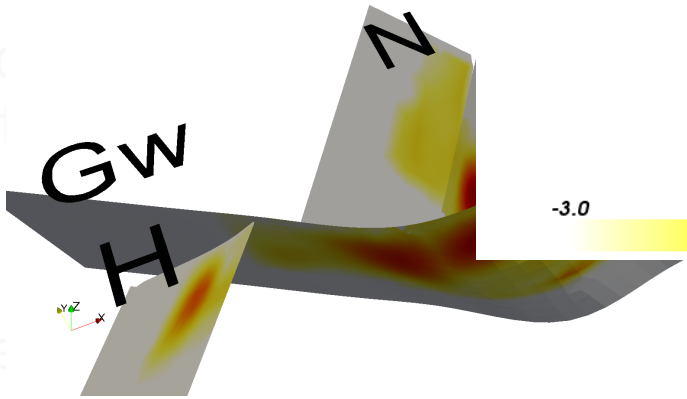
PyLith

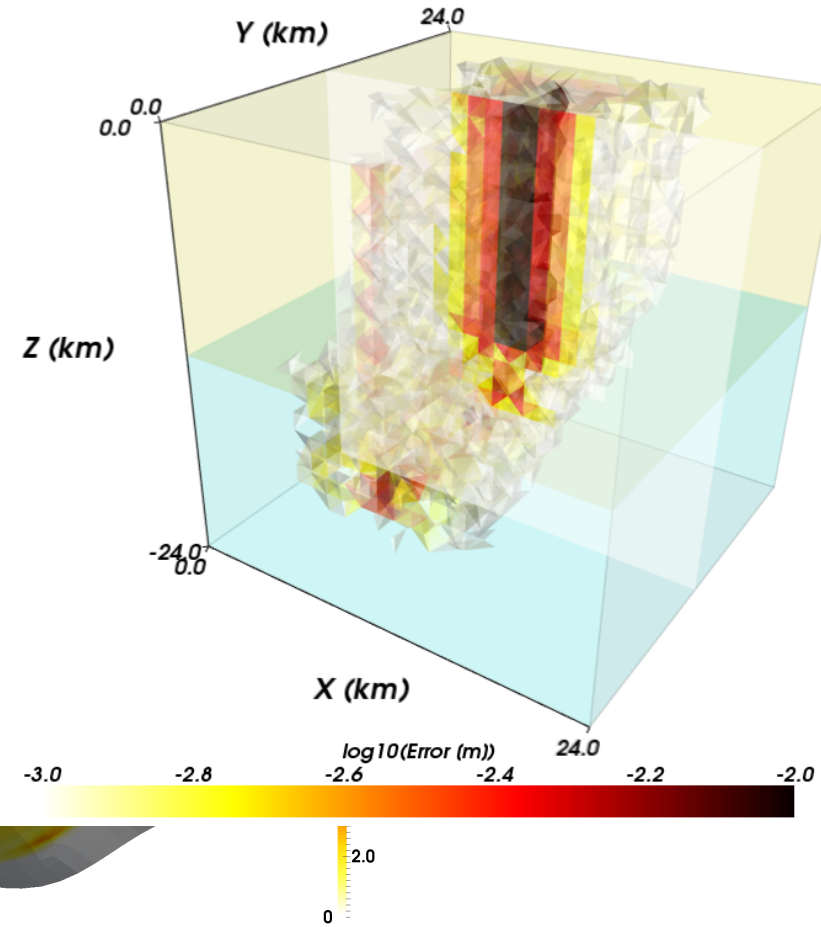
- Multiple problems
  - Dynamic rupture
  - Quasi-static relaxation
- Multiple models
  - Fault constituti
  - Nonlinear visco
  - Finite deformat
- Multiple Meshes
  - 1D, 2D, 3D
  - Hex and tet me



PyLith

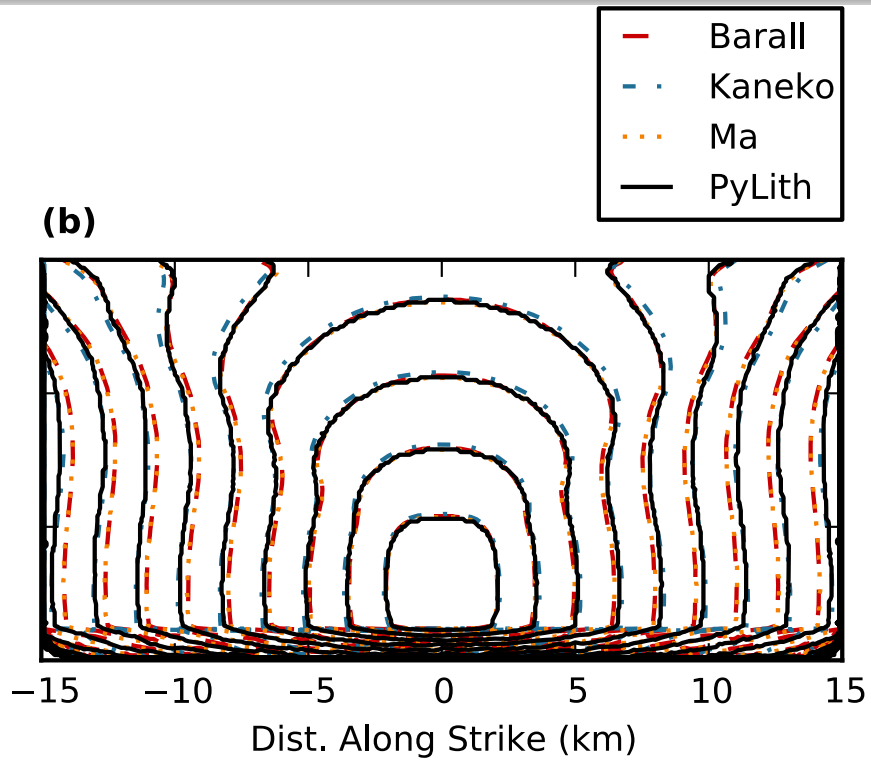
- Multiple problems
  - Dynamic rupture
  - Quasi-static relaxation
- Multiple models
  - Fault constituti
  - Nonlinear visco
  - Finite deformat
- Multiple Meshes
  - 1D, 2D, 3D
  - Hex and tet me





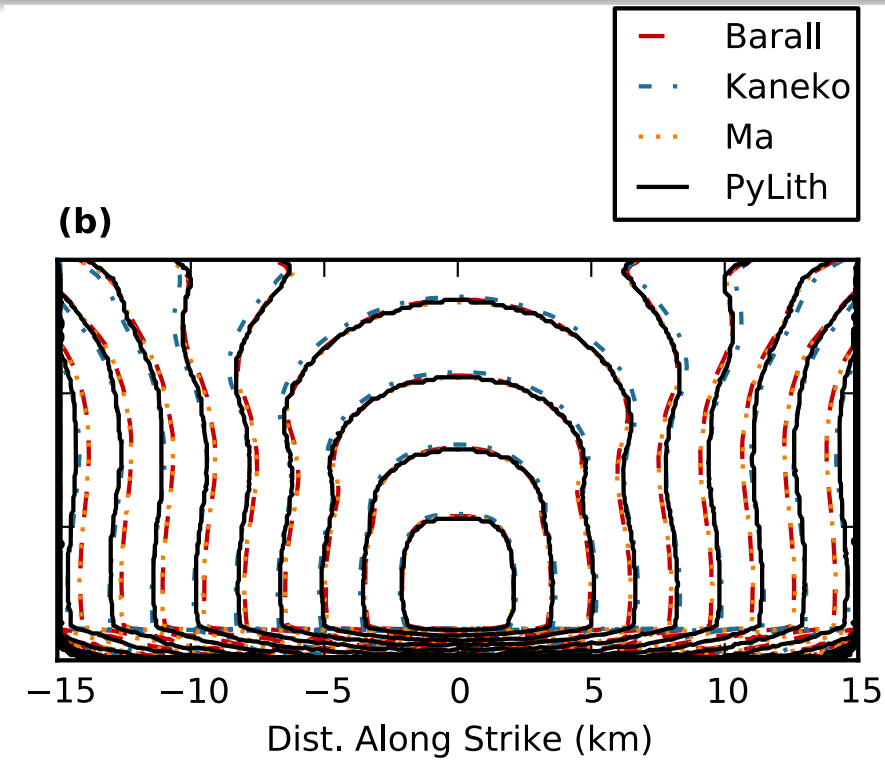
PyLith

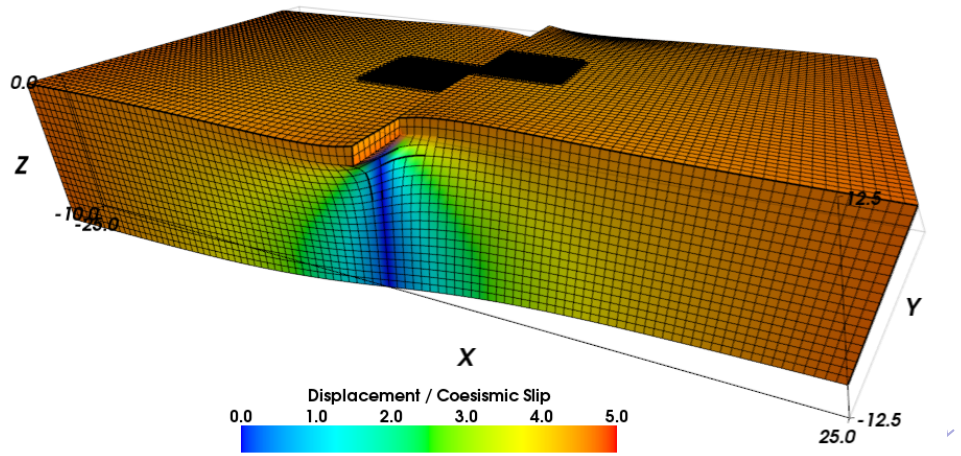
- Multiple problems
  - Dynamic rupture
  - Quasi-static relaxation
- Multiple models
  - Fault constitutive models
  - Nonlinear visco-elastic-plastic
  - Finite deformation
- Multiple Meshes
  - 1D, 2D, 3D
  - Hex and tet meshes



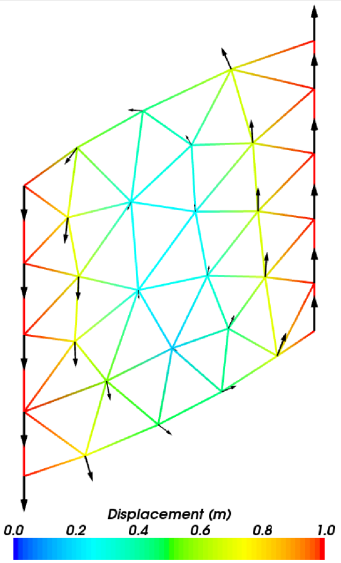
PyLith

- Multiple problems
  - Dynamic rupture
  - Quasi-static relaxation
- Multiple models
  - Fault constitutive models
  - Nonlinear visco-elastic-plastic
  - Finite deformation
- Multiple Meshes
  - 1D, 2D, 3D
  - Hex and tet meshes





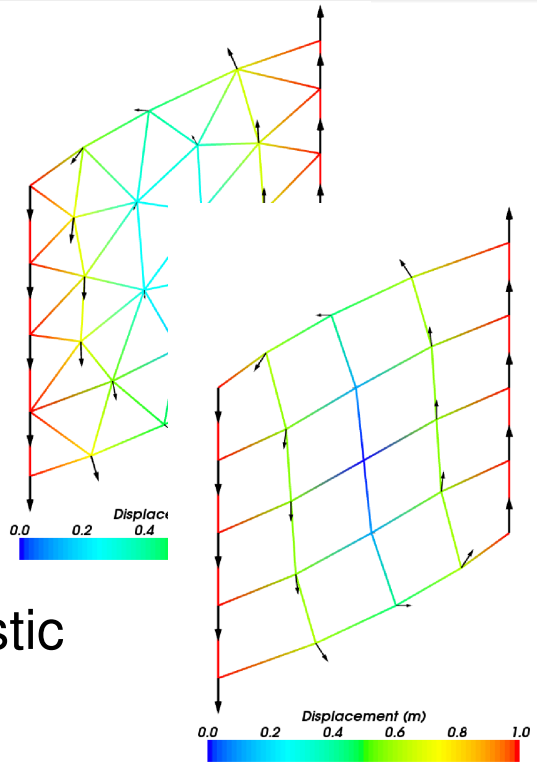
PyLith



- Multiple problems
  - Dynamic rupture
  - Quasi-static relaxation
- Multiple models
  - Fault constitutive models
  - Nonlinear visco-elastic-plastic
  - Finite deformation
- Multiple Meshes
  - 1D, 2D, 3D
  - Hex and tet meshes

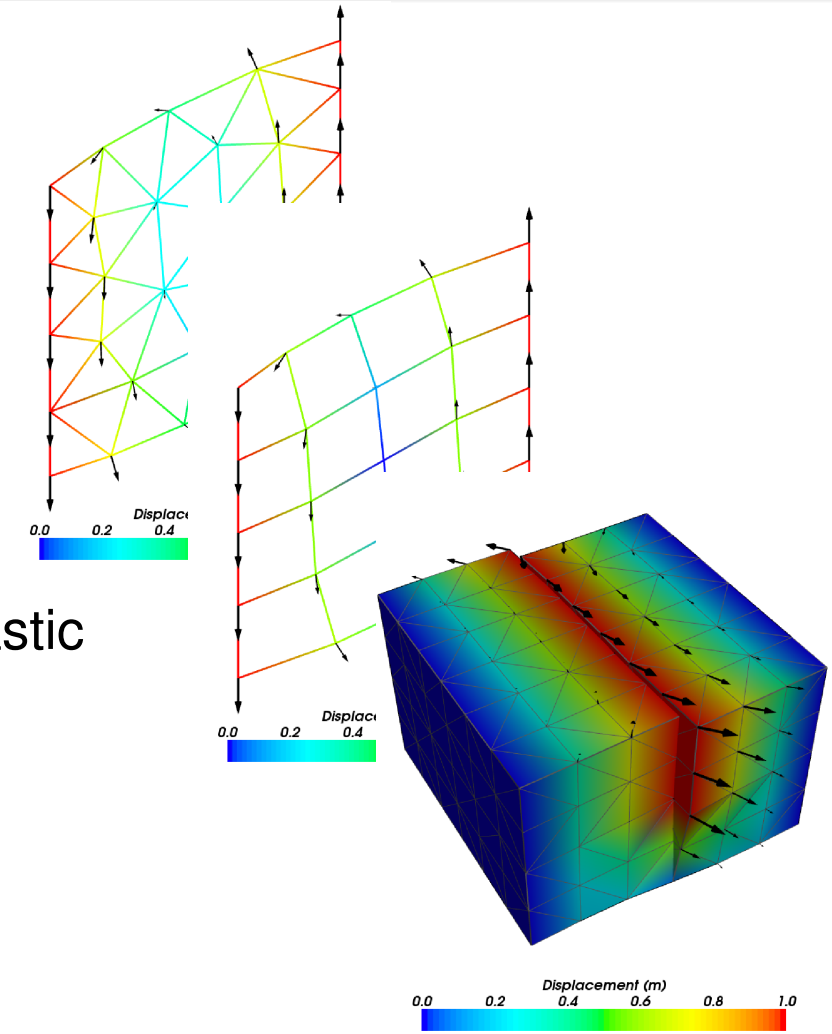
PyLith

- Multiple problems
  - Dynamic rupture
  - Quasi-static relaxation
- Multiple models
  - Fault constitutive models
  - Nonlinear visco-elastic-plastic
  - Finite deformation
- Multiple Meshes
  - 1D, 2D, 3D
  - Hex and tet meshes



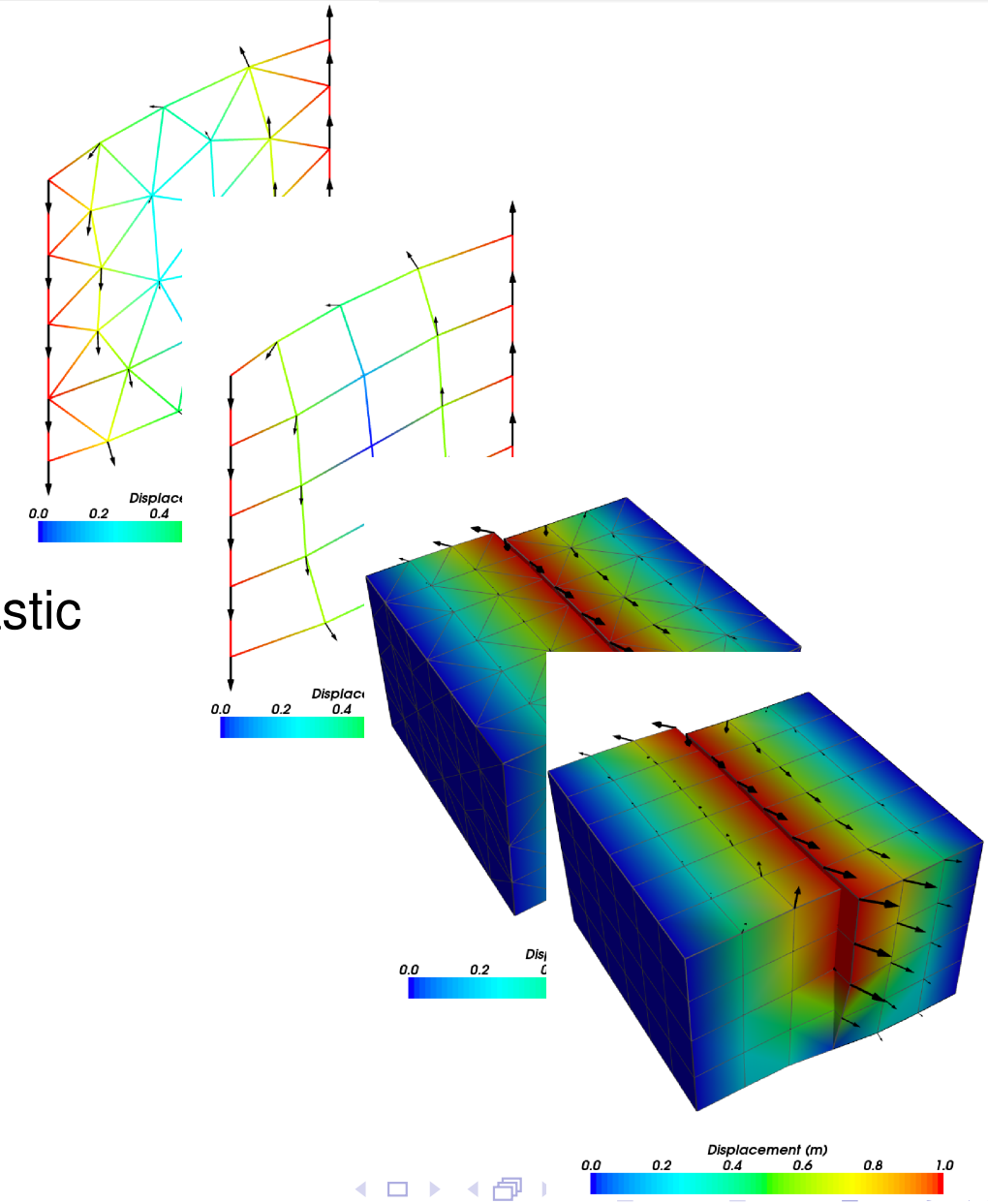
PyLith

- Multiple problems
  - Dynamic rupture
  - Quasi-static relaxation
- Multiple models
  - Fault constitutive models
  - Nonlinear visco-elastic-plastic
  - Finite deformation
- Multiple Meshes
  - 1D, 2D, 3D
  - Hex and tet meshes



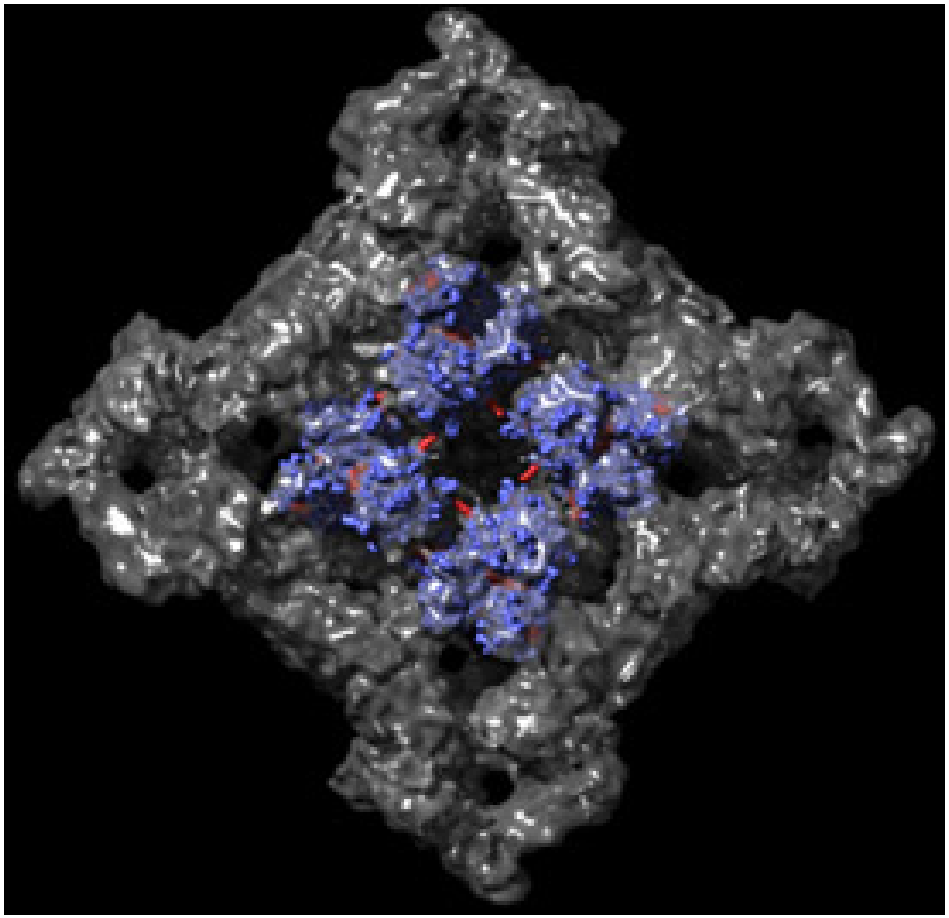
PyLith

- Multiple problems
  - Dynamic rupture
  - Quasi-static relaxation
- Multiple models
  - Fault constitutive models
  - Nonlinear visco-elastic-plastic
  - Finite deformation
- Multiple Meshes
  - 1D, 2D, 3D
  - Hex and tet meshes



Classical DFT in Three Dimensions

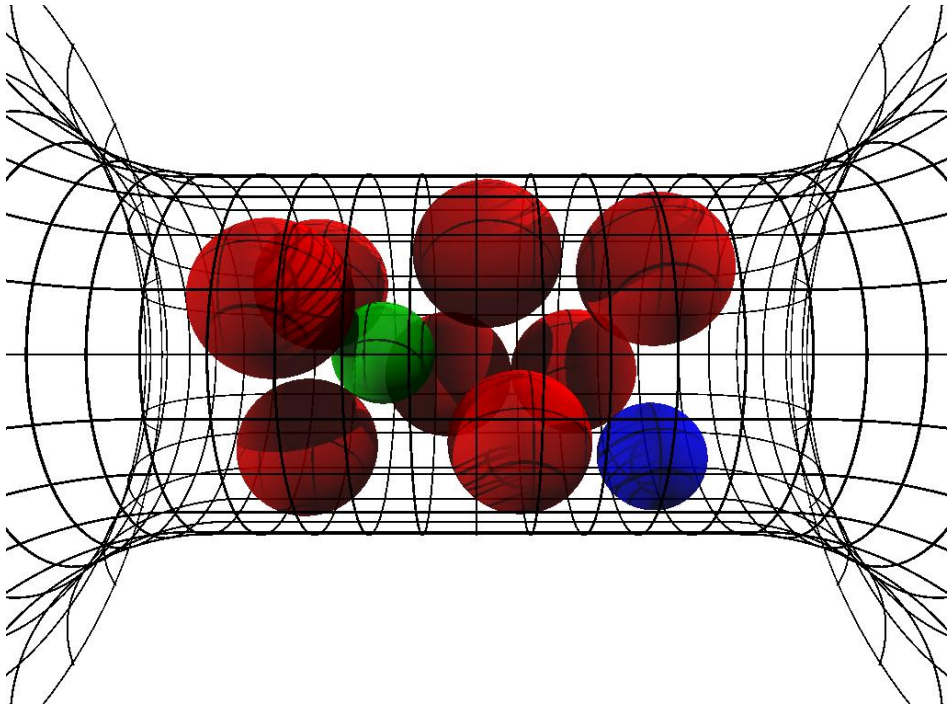
I wrote the first 3D Classical DFT with true hard sphere chemical potential using fundamental measure theory. It used an $\mathcal{O}(N \log N)$ algorithm based upon the FFT. We examined the physics of ion channels, such as the ryanodine receptor. Advanced electrostatics allowed prediction of I-V curves for 100+ solutions, including polyvalent species.



The implementation is detailed in An Efficient Algorithm for Classical Density Functional Theory in Three Dimensions: Ionic Solutions, JCP, 2012.

Classical DFT in Three Dimensions

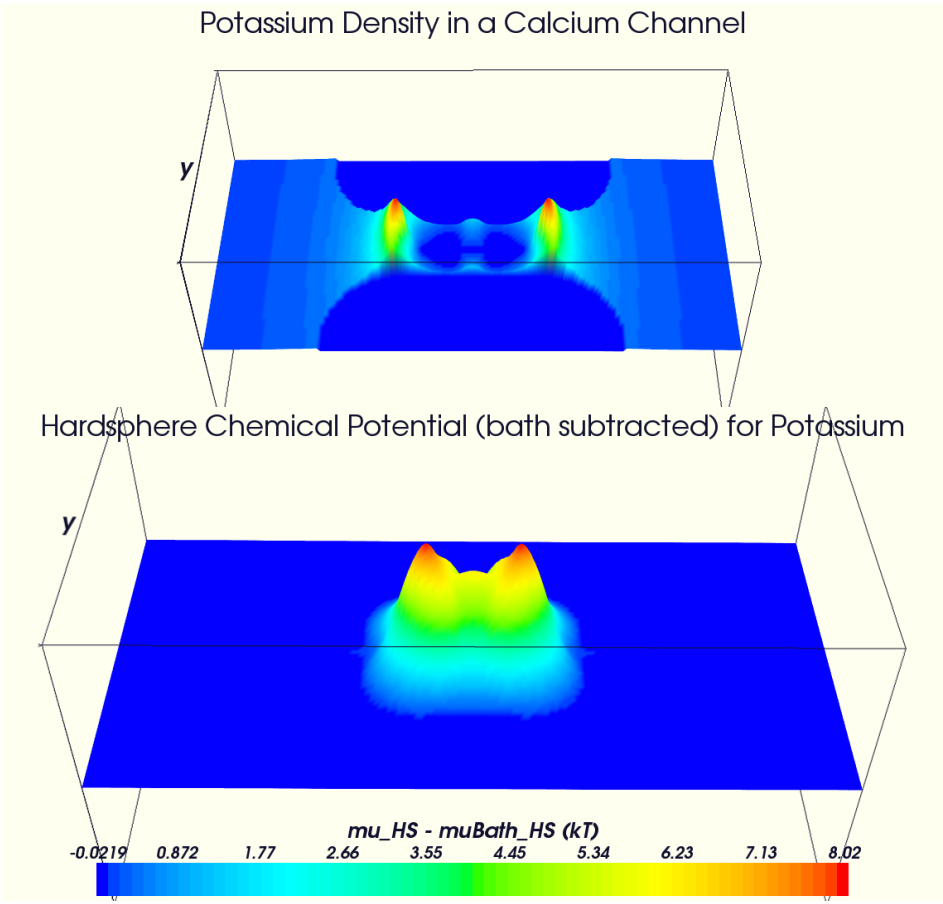
I wrote the first 3D Classical DFT with true hard sphere chemical potential using fundamental measure theory. It used an $\mathcal{O}(N \log N)$ algorithm based upon the FFT. We examined the physics of ion channels, such as the ryanodine receptor. Advanced electrostatics allowed prediction of I-V curves for 100+ solutions, including polyvalent species.



The implementation is detailed in An Efficient Algorithm for Classical Density Functional Theory in Three Dimensions: Ionic Solutions, JCP, 2012.

Classical DFT in Three Dimensions

I wrote the first 3D Classical DFT with true hard sphere chemical potential using fundamental measure theory. It used an $\mathcal{O}(N \log N)$ algorithm based upon the FFT. We examined the physics of ion channels, such as the ryanodine receptor. Advanced electrostatics allowed prediction of I-V curves for 100+ solutions, including polyvalent species.



The implementation is detailed in An Efficient Algorithm for Classical Density Functional Theory in Three Dimensions: Ionic Solutions, JCP, 2012.

Projects Using My Mesh

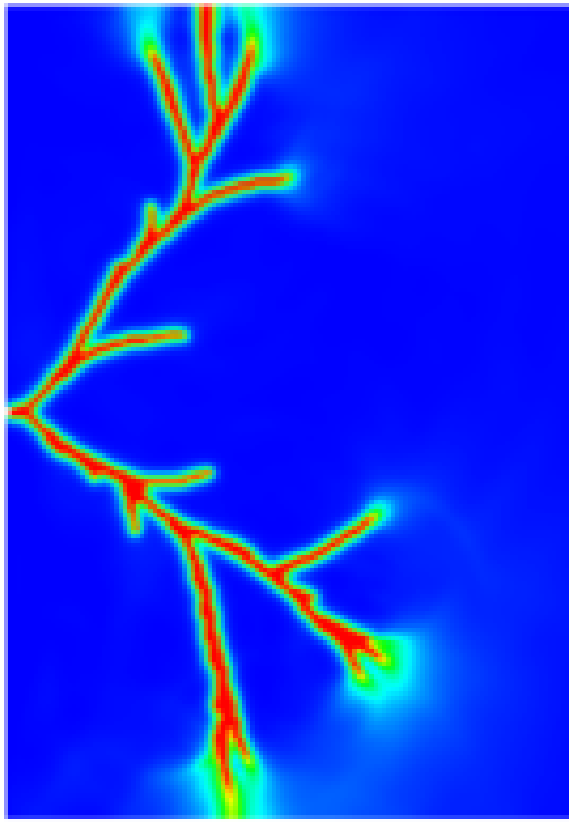
Fracture simulation, Blaise Bourdin

- Full variational formulation
  - Phase field for crack
  - Linear or quadratic penalty
- Cracks are not prescribed
  - Arbitrary crack geometry
  - Arbitrary crack intersections
- Multiple materials and composite toughness

Projects Using My Mesh

Fracture simulation, Blaise Bourdin

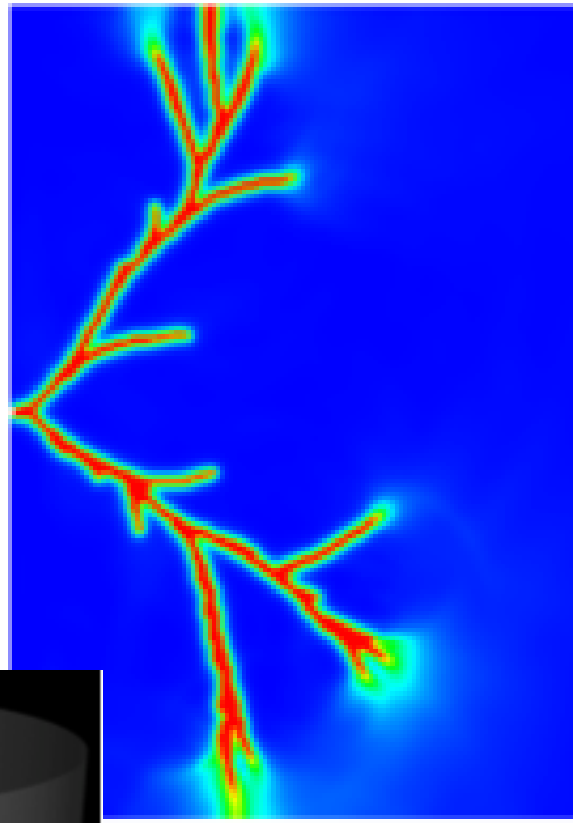
- Full variational formulation
  - Phase field for crack
  - Linear or quadratic penalty
- Cracks are not prescribed
  - Arbitrary crack geometry
  - Arbitrary crack intersections
- Multiple materials and composite toughness

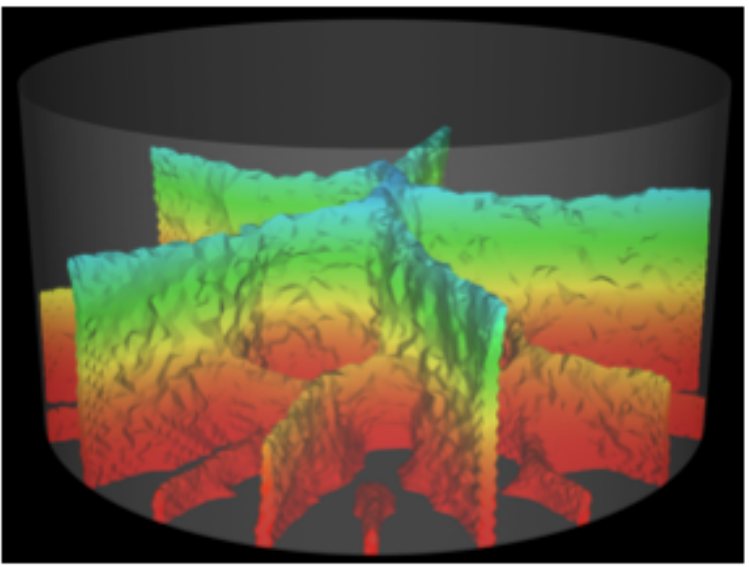


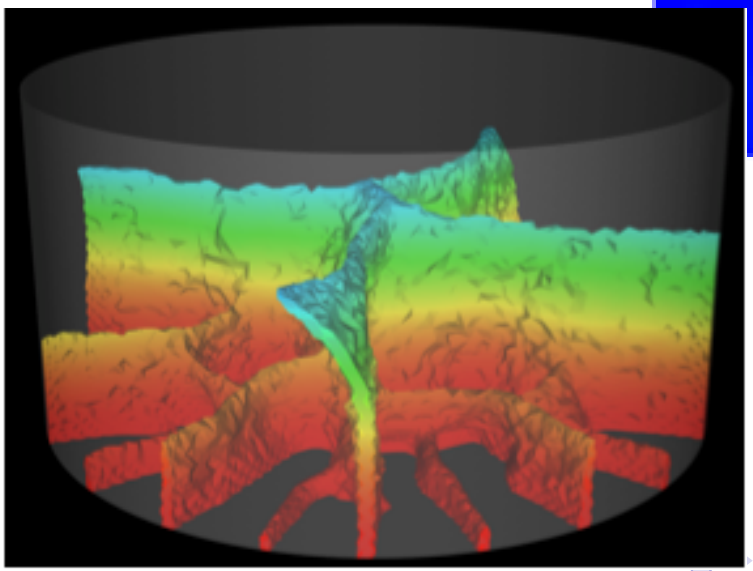
Projects Using My Mesh

Fracture simulation, Blaise Bourdin

- Full variational formulation
  - Phase field for crack
  - Linear or quadratic penalty
- Cracks are not prescribed
  - Arbitrary crack geometry
  - Arbitrary crack intersections
- Multiple materials and composite toughness







Projects Using My Mesh

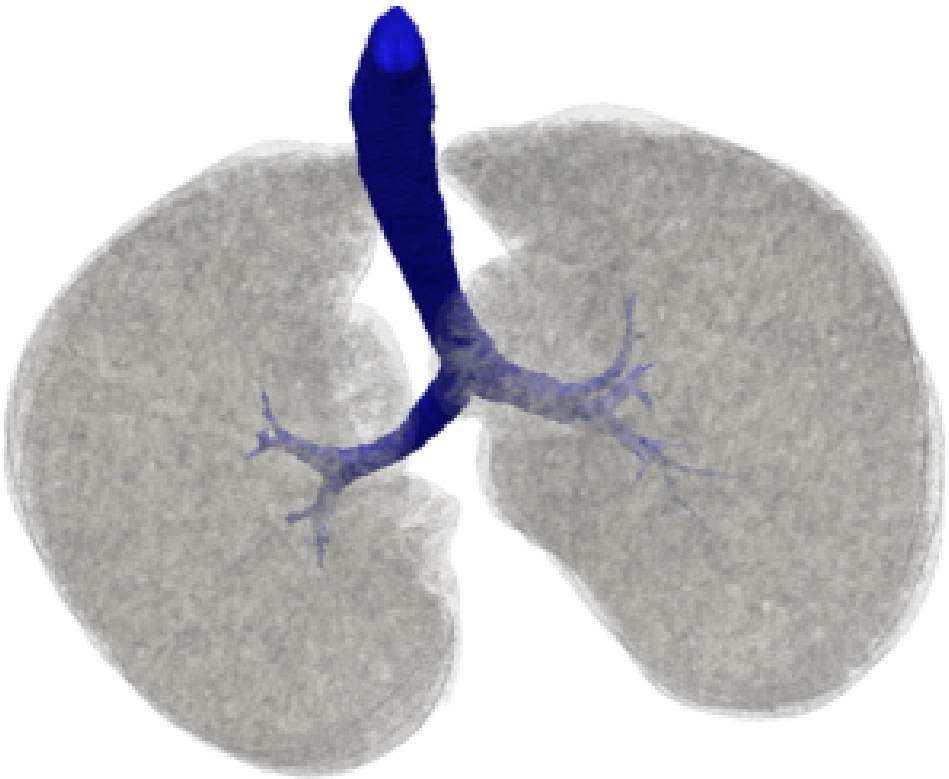
Respiration modeling, HiFlow3

- Multi-purpose finite element software
- Arose from EMCL at Karlsruhe Institute of Technology
- Flow behavior in the human respiratory system

Projects Using My Mesh

Respiration modeling, HiFlow3

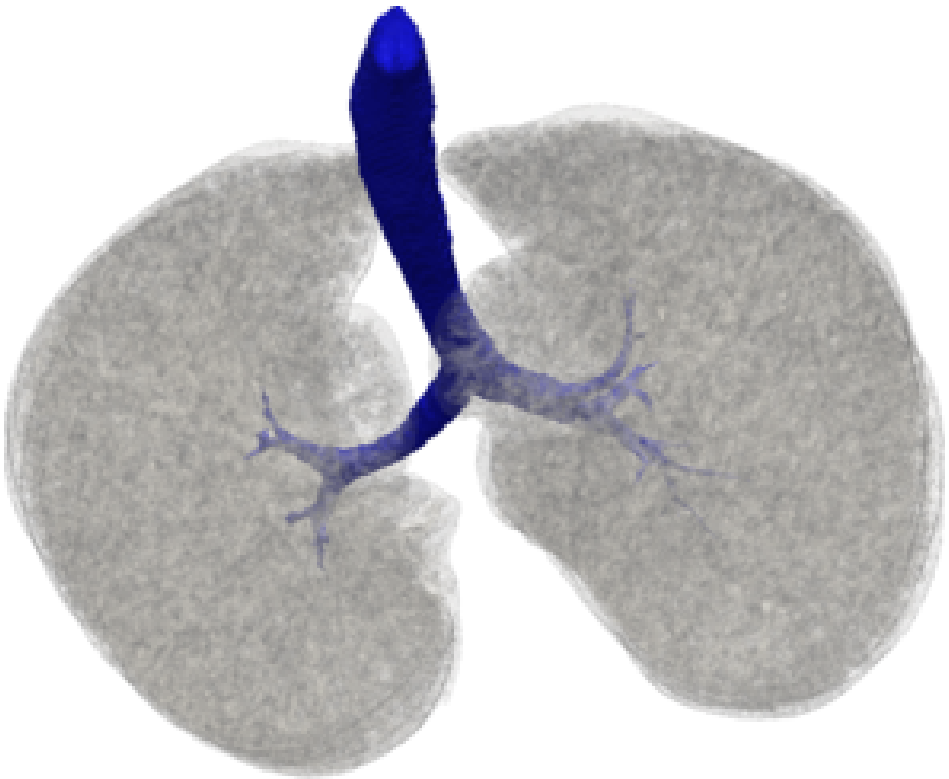
- Multi-purpose finite element software
- Arose from EMCL at Karlsruhe Institute of Technology
- Flow behavior in the human respiratory system

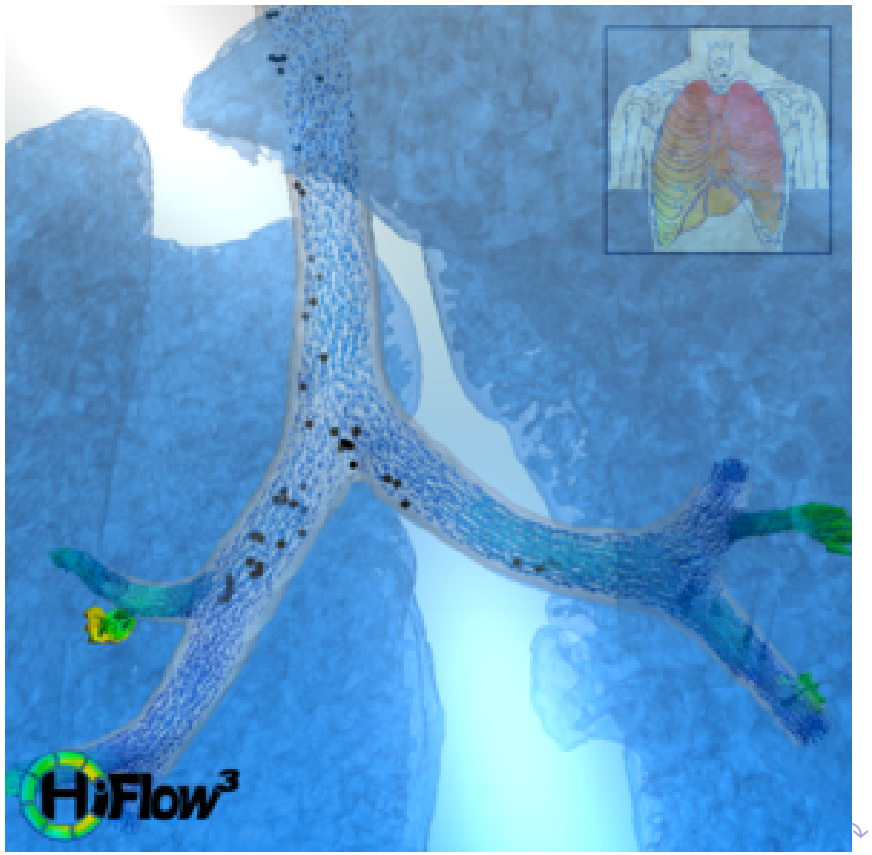


Projects Using My Mesh

Respiration modeling, HiFlow3

- Multi-purpose finite element software
- Arose from EMCL at Karlsruhe Institute of Technology
- Flow behavior in the human respiratory system

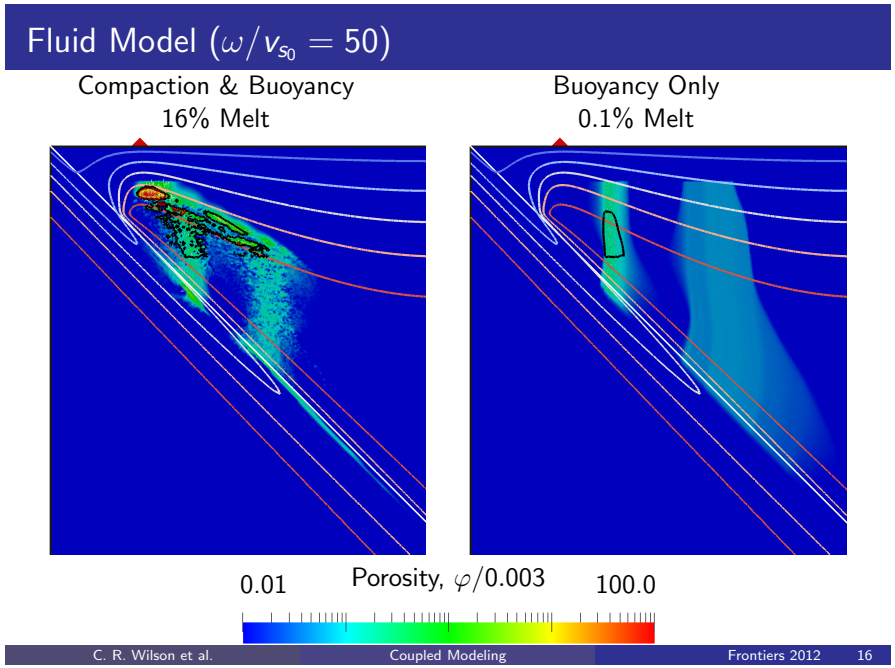




Projects Using PETSc Composable Solvers

Magma Dynamics, TerraFERMA (Columbia)

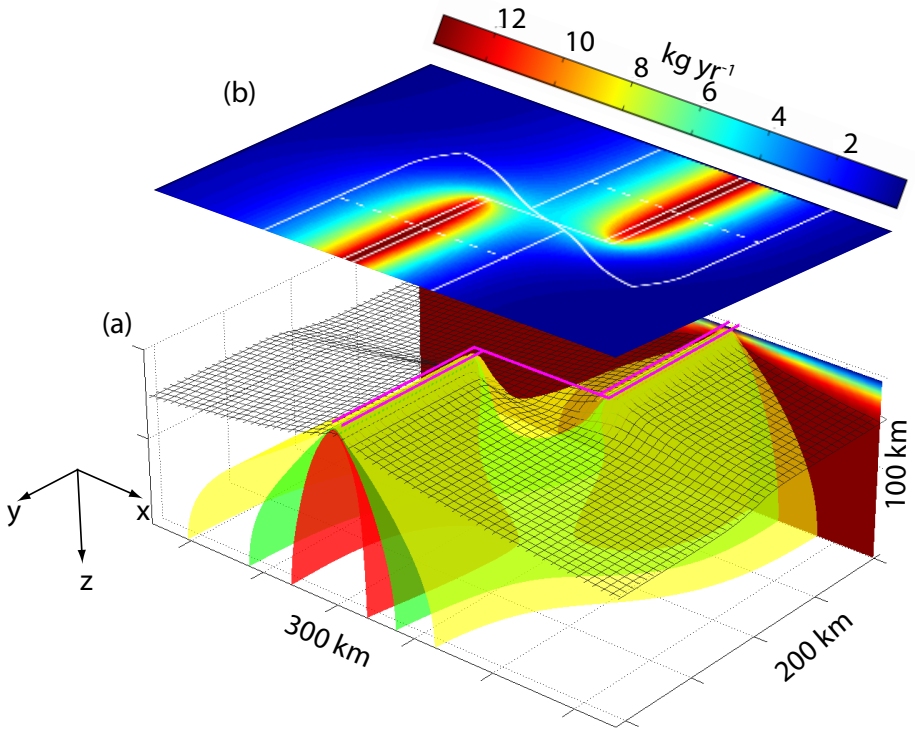
- Cian Wilson and Marc Spiegelman
- Flexible model builder
- Finite element
- Nested FieldSplit solver

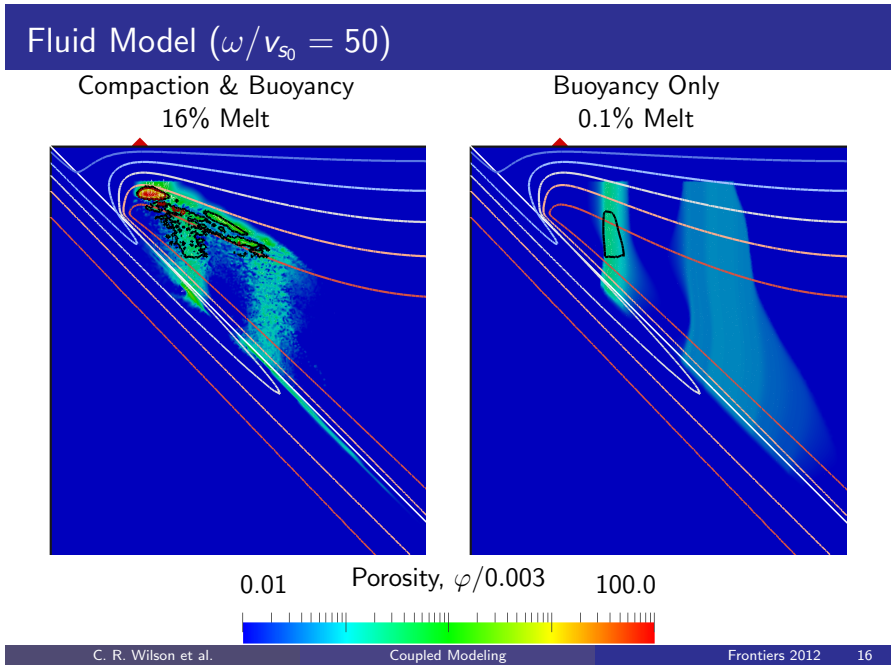


Projects Using PETSc Composable Solvers

Magma Dynamics (Oxford)

- Sam Weatherley and Richard Katz
- Finite volume
- Nested FieldSplit solver
- Small scale parallel (10^2–10^3)

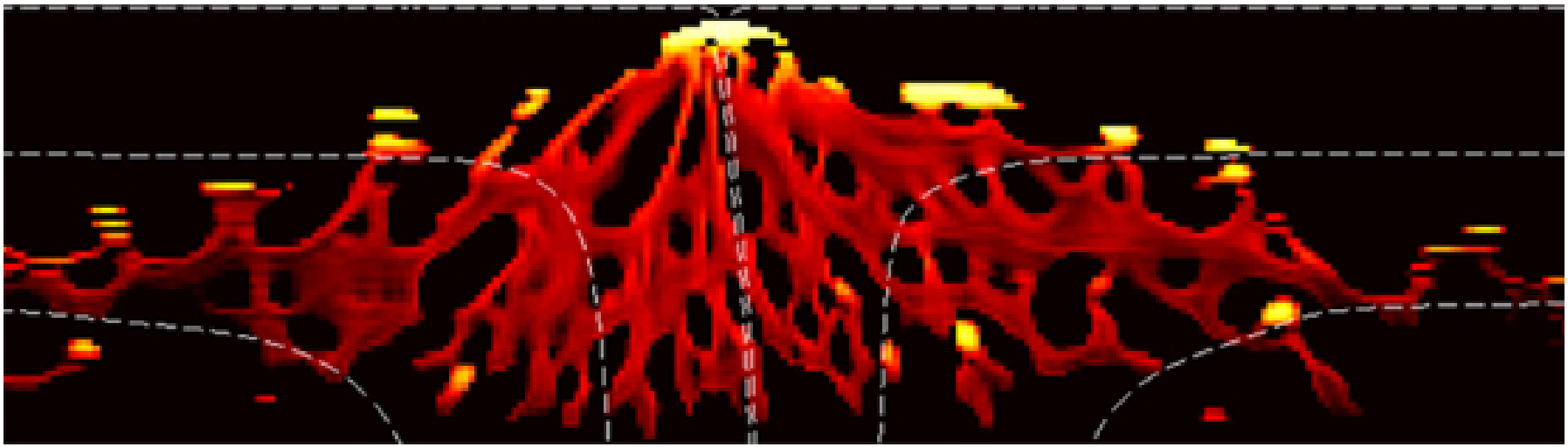




Projects Using PETSc Composable Solvers

Magma Dynamics (Oxford)

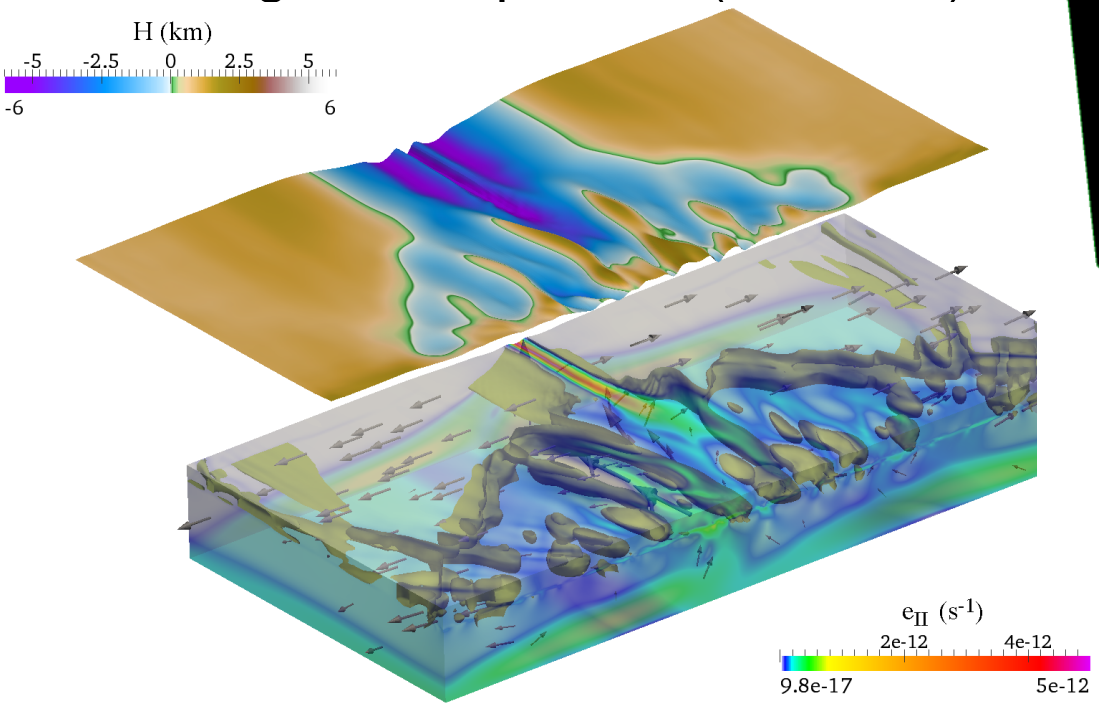
- Sam Weatherley and Richard Katz
- Finite volume
- Nested FieldSplit solver
- Small scale parallel (10^2–10^3)

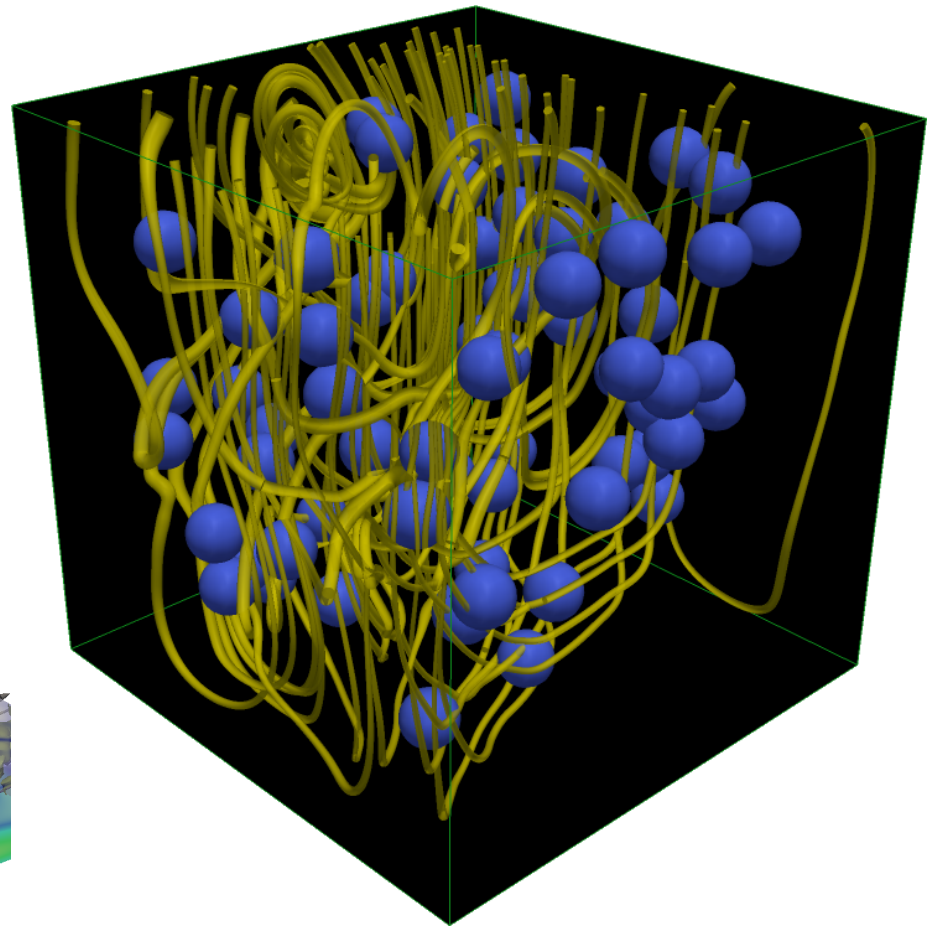


Projects Using PETSc Composable Solvers

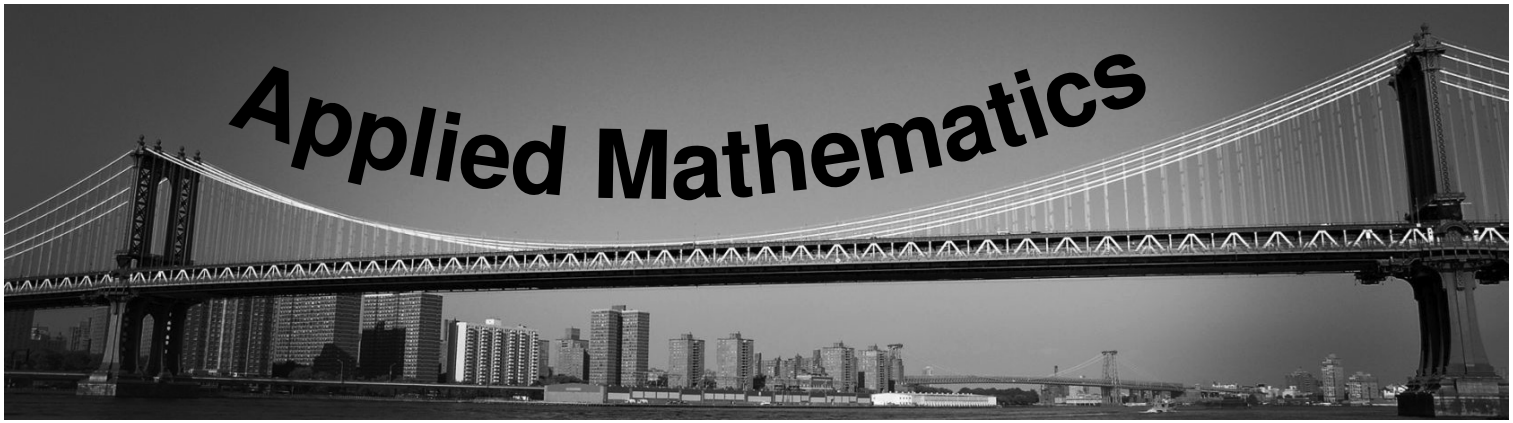
Lithospheric and Mantle dynamics, PTatin3d (ETHZ)

- Dave May
- Finite element
- Lagrangian particles
- Nested FieldSplit solver
- Large scale parallel (10^3–10^5)



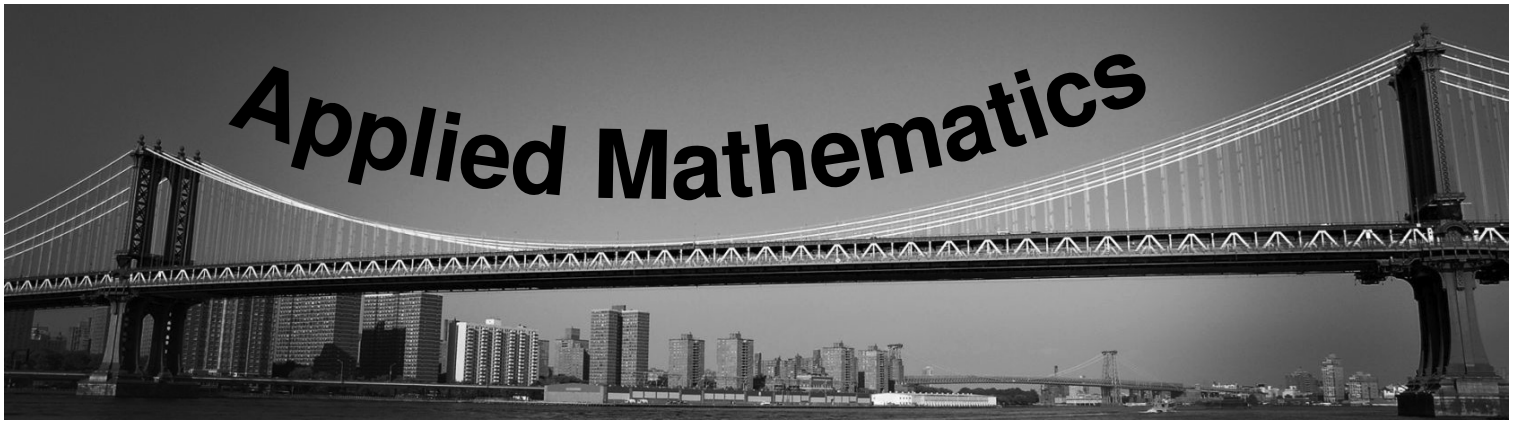


Impact of Mathematics on Science

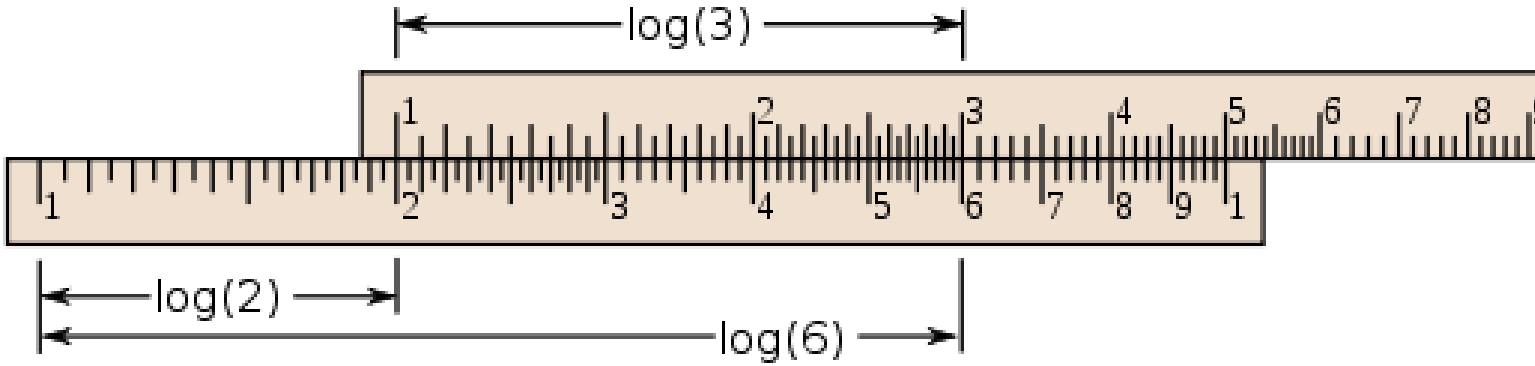


Computational Leaders have always
embraced the latest technology
and been inspired by physical problems,

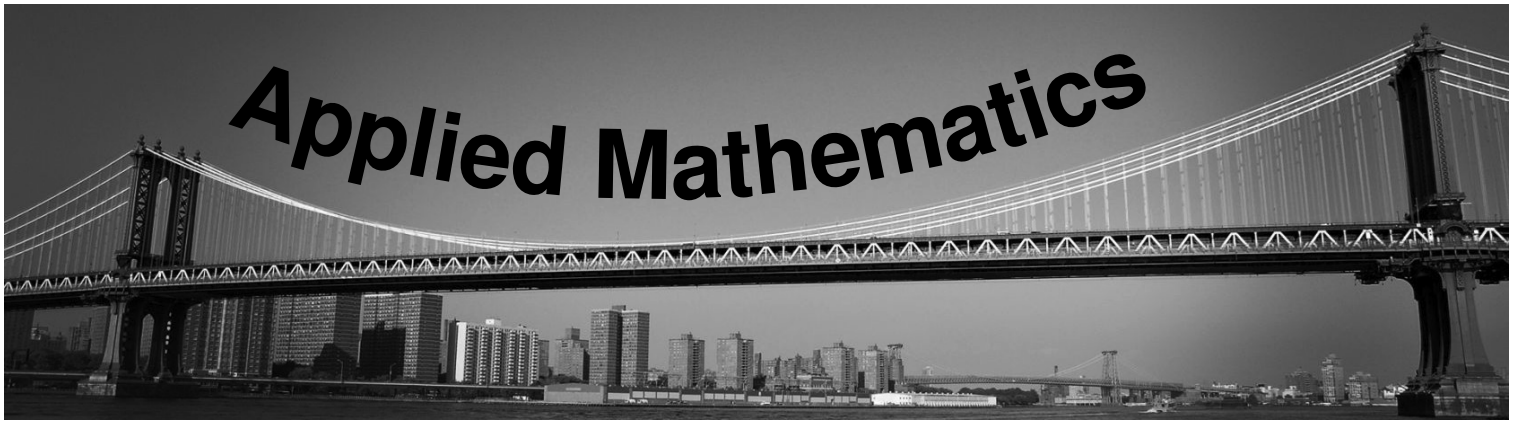
Impact of Mathematics on Science



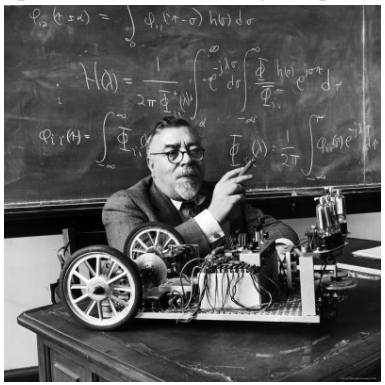
Computational Leaders have always
embraced the latest technology
and been inspired by physical problems,

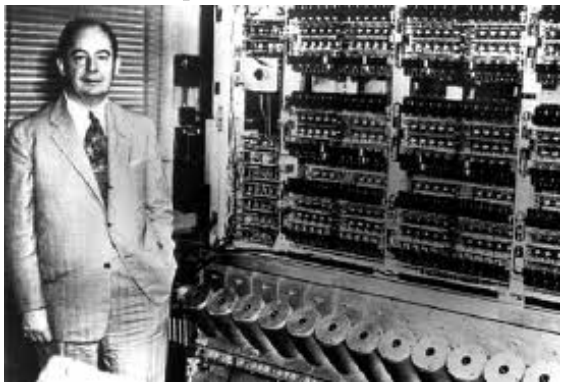


Impact of Mathematics on Science

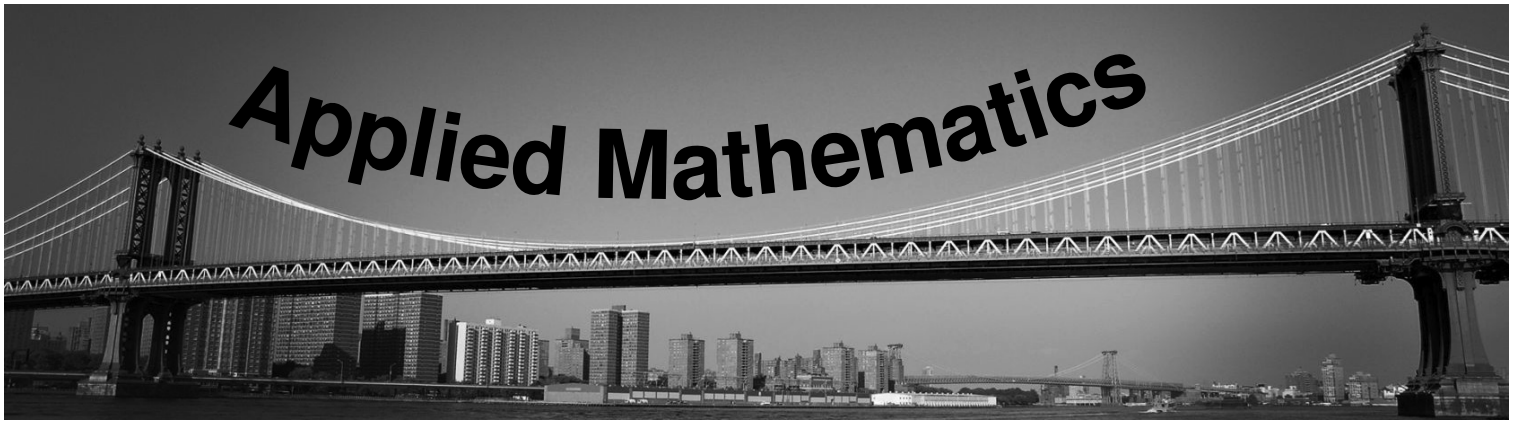


Computational Leaders have always
embraced the latest technology
and been inspired by physical problems,





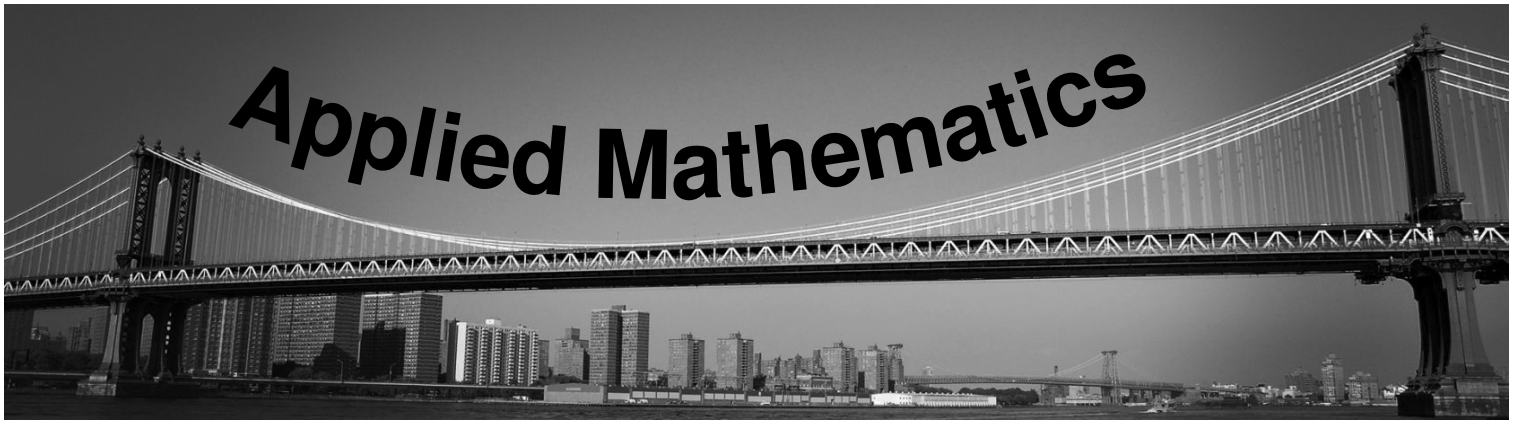
Impact of Mathematics on Science



Computational Leaders have always
embraced the latest technology
and been inspired by physical problems,

PETSc

Impact of Mathematics on Science



Computational Leaders have always
embraced the latest technology
and been inspired by physical problems,

Enabling Scientific Discovery

Additional Slides

Programming with Options

ex55: Allen-Cahn problem in 2D

- constant mobility
- triangular elements

Geometric multigrid method for saddle point variational inequalities:

```
./ex55 -ksp_type fgmres -pc_type mg -mg_levels_ksp_type fgmres
-mg_levels_pc_type fieldsplit -mg_levels_pc_fieldsplit_detect_saddle_point
-mg_levels_pc_fieldsplit_type schur -da_grid_x 65 -da_grid_y 65
-mg_levels_pc_fieldsplit_factorization_type full
-mg_levels_pc_fieldsplit_schur_precondition user
-mg_levels_fieldsplit_1_ksp_type gmres -mg_coarse_ksp_type preonly
-mg_levels_fieldsplit_1_pc_type none   -mg_coarse_pc_type svd
-mg_levels_fieldsplit_0_ksp_type preonly
-mg_levels_fieldsplit_0_pc_type sor    -pc_mg_levels 5
-mg_levels_fieldsplit_0_pc_sor_forward -pc_mg_galerkin
-snes_vi_monitor -ksp_monitor_true_residual -snes_atol 1.e-11
-mg_levels_ksp_monitor -mg_levels_fieldsplit_ksp_monitor
-mg_levels_ksp_max_it 2 -mg_levels_fieldsplit_ksp_max_it 5
```

Programming with Options

ex55: Allen-Cahn problem in 2D

Run flexible GMRES with 5 levels of multigrid as the preconditioner

```
./ex55 -ksp_type fgmres -pc_type mg -pc_mg_levels 5
  -da_grid_x 65 -da_grid_y 65
```

Use the Galerkin process to compute the coarse grid operators

```
-pc_mg_galerkin
```

Use SVD as the coarse grid saddle point solver

```
-mg_coarse_ksp_type preonly -mg_coarse_pc_type svd
```

Programming with Options

ex55: Allen-Cahn problem in 2D

Run flexible GMRES with 5 levels of multigrid as the preconditioner

```
./ex55 -ksp_type fgmres -pc_type mg -pc_mg_levels 5
  -da_grid_x 65 -da_grid_y 65
```

Use the Galerkin process to compute the coarse grid operators

```
-pc_mg_galerkin
```

Use SVD as the coarse grid saddle point solver

```
-mg_coarse_ksp_type preonly -mg_coarse_pc_type svd
```

Programming with Options

ex55: Allen-Cahn problem in 2D

Run flexible GMRES with 5 levels of multigrid as the preconditioner

```
./ex55 -ksp_type fgmres -pc_type mg -pc_mg_levels 5
  -da_grid_x 65 -da_grid_y 65
```

Use the Galerkin process to compute the coarse grid operators

```
-pc_mg_galerkin
```

Use SVD as the coarse grid saddle point solver

```
-mg_coarse_ksp_type preonly -mg_coarse_pc_type svd
```

Programming with Options

ex55: Allen-Cahn problem in 2D

Run flexible GMRES with 5 levels of multigrid as the preconditioner

```
./ex55 -ksp_type fgmres -pc_type mg -pc_mg_levels 5
  -da_grid_x 65 -da_grid_y 65
```

Use the Galerkin process to compute the coarse grid operators

```
-pc_mg_galerkin
```

Use SVD as the coarse grid saddle point solver

```
-mg_coarse_ksp_type preonly -mg_coarse_pc_type svd
```

Programming with Options

ex55: Allen-Cahn problem in 2D

Smoother: Flexible GMRES (2 iterates) with a Schur complement PC

```
-mg_levels_ksp_type fgmres -mg_levels_pc_fieldsplit_detect_saddle_point
-mg_levels_ksp_max_it 2 -mg_levels_pc_type fieldsplit
-mg_levels_pc_fieldsplit_type schur
-mg_levels_pc_fieldsplit_factorization_type full
-mg_levels_pc_fieldsplit_schur_precondition diag
```

Schur complement solver: GMRES (5 iterates) with no preconditioner

```
-mg_levels_fieldsplit_1_ksp_type gmres
-mg_levels_fieldsplit_1_pc_type none -mg_levels_fieldsplit_ksp_max_it 5
```

Schur complement action: Use only the lower diagonal part of A00

```
-mg_levels_fieldsplit_0_ksp_type preonly
-mg_levels_fieldsplit_0_pc_type sor
-mg_levels_fieldsplit_0_pc_sor_forward
```

Programming with Options

ex55: Allen-Cahn problem in 2D

Smoother: Flexible GMRES (2 iterates) with a Schur complement PC

```
-mg_levels_ksp_type fgmres -mg_levels_pc_fieldsplit_detect_saddle_point
-mg_levels_ksp_max_it 2 -mg_levels_pc_type fieldsplit
-mg_levels_pc_fieldsplit_type schur
-mg_levels_pc_fieldsplit_factorization_type full
-mg_levels_pc_fieldsplit_schur_precondition diag
```

Schur complement solver: GMRES (5 iterates) with no preconditioner

```
-mg_levels_fieldsplit_1_ksp_type gmres
-mg_levels_fieldsplit_1_pc_type none -mg_levels_fieldsplit_ksp_max_it 5
```

Schur complement action: Use only the lower diagonal part of A00

```
-mg_levels_fieldsplit_0_ksp_type preonly
-mg_levels_fieldsplit_0_pc_type sor
-mg_levels_fieldsplit_0_pc_sor_forward
```

Programming with Options

ex55: Allen-Cahn problem in 2D

Smoother: Flexible GMRES (2 iterates) with a Schur complement PC

```
-mg_levels_ksp_type fgmres -mg_levels_pc_fieldsplit_detect_saddle_point
-mg_levels_ksp_max_it 2 -mg_levels_pc_type fieldsplit
-mg_levels_pc_fieldsplit_type schur
-mg_levels_pc_fieldsplit_factorization_type full
-mg_levels_pc_fieldsplit_schur_precondition diag
```

Schur complement solver: GMRES (5 iterates) with no preconditioner

```
-mg_levels_fieldsplit_1_ksp_type gmres
-mg_levels_fieldsplit_1_pc_type none -mg_levels_fieldsplit_ksp_max_it 5
```

Schur complement action: Use only the lower diagonal part of A00

```
-mg_levels_fieldsplit_0_ksp_type preonly
-mg_levels_fieldsplit_0_pc_type sor
-mg_levels_fieldsplit_0_pc_sor_forward
```

Programming with Options

ex55: Allen-Cahn problem in 2D

Smoother: Flexible GMRES (2 iterates) with a Schur complement PC

```
-mg_levels_ksp_type fgmres -mg_levels_pc_fieldsplit_detect_saddle_point
-mg_levels_ksp_max_it 2 -mg_levels_pc_type fieldsplit
-mg_levels_pc_fieldsplit_type schur
-mg_levels_pc_fieldsplit_factorization_type full
-mg_levels_pc_fieldsplit_schur_precondition diag
```

Schur complement solver: GMRES (5 iterates) with no preconditioner

```
-mg_levels_fieldsplit_1_ksp_type gmres
-mg_levels_fieldsplit_1_pc_type none -mg_levels_fieldsplit_ksp_max_it 5
```

Schur complement action: Use only the lower diagonal part of A00

```
-mg_levels_fieldsplit_0_ksp_type preonly
-mg_levels_fieldsplit_0_pc_type sor
-mg_levels_fieldsplit_0_pc_sor_forward
```

Nonlinear Preconditioning

- Major Point: Composable structures for computation reduce system complexity and generate real application benefits
- Minor Point: Numerical libraries are communication medium for scientific results
- Minor Point: Optimal solvers can be constructed on the fly to suit the problem
- Slides for Stokes PCs
- Slide with programming with options

Nonlinear Preconditioning

- NPC in PETSc
- Paper with Barry and Peter
- Cite Peter and Jed paper for use cases

Parallel Fast Multipole Method

- Using mesh partitioner to develop schedule removes load balance barrier
- Partitioner can be proved to work with Teng's result
- Simple parallelization can be proved to work with overlap
- Ex: Work with May, 512 GPU paper

GPU Computing

- Papers with Andy about FEM Integration
- Paper with PETSc about solvers
- Conferences with Yuen