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
starting with the numerics of PDEs, and mathematics of the computation, through the distillation into high quality numerical libraries, to scientific discovery through computing.
starting with the numerics of PDEs, and mathematics of the computation, through the distillation into high quality numerical libraries, to scientific discovery through computing.
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

PETSc  CIG Rep  CIG EC  Rush

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
\[ \nabla^2 \varphi_{\text{protein}}(\mathbf{r}) = -\sum_i \frac{q_i \delta(\mathbf{r} - \mathbf{r}_i)}{\varepsilon_0 \varepsilon_{\text{protein}}} \]

\[ \nabla^2 \varphi_{\text{solvent}}(\mathbf{r}) = 0 \]
\[
\sigma(\vec{r}) + \hat{\epsilon} \int_{\Gamma} \frac{\partial}{\partial n(\vec{r})} \frac{\sigma(\vec{r}') d^2 \vec{r}'}{4\pi ||\vec{r} - \vec{r}'||} = -\hat{\epsilon} \sum_{k=1}^{Q} \frac{\partial}{\partial n(\vec{r})} \frac{q_k}{4\pi ||\vec{r} - \vec{r}_k||} \tag{1}
\]

\[
(I + \hat{\epsilon} \mathcal{D}^*) \sigma(\vec{r}) = \tag{2}
\]

where we define

\[
\hat{\epsilon} = \frac{1}{2} \frac{\epsilon_1 - \epsilon_2}{\epsilon_1 + \epsilon_2} < 0
\]
The *reaction* potential is given by

\[
\phi^R(\vec{r}) = \int_{\Gamma} \frac{\sigma(\vec{r}') d^2 \vec{r}'}{4 \pi \varepsilon_1 ||\vec{r} - \vec{r}'||}
\]

which defines the electrostatic part of the solvation free energy

\[
\Delta G_{es} = \frac{1}{2} q^T \phi^R \\
= \frac{1}{2} q^T Lq \\
= \frac{1}{2} q^T C A^{-1} B q
\]
Boundary element discretizations of the solvation problem (Eq. 1):

- can be expensive to solve
- are more accurate than required by intermediate design iterations
Boundary Integral-Based Electrostatics Estimation

**Coulomb Field Approximation**: uniform normal field

\[
\left(1 - \frac{\hat{\epsilon}}{2}\right) \sigma_{CFA} = Bq
\]  
(3)

**Preconditioning**: consider only local effects

\[
\sigma_P = Bq
\]  
(4)

**Lower Bound**: no good physical motivation

\[
\left(1 + \frac{\hat{\epsilon}}{2}\right) \sigma_{LB} = Bq
\]  
(5)
We will need the single layer operator $S$

$$S\tau(\vec{r}) = \int \frac{\tau(\vec{r}') d^2 \vec{r}'}{4\pi ||\vec{r} - \vec{r}'||}$$
The potential at the boundary $\Gamma$ given by

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

can also be obtained by solving an exterior Neumann problem for $\tau$,

$$\phi^{Coulomb}(\vec{r}) = S\tau$$

$$= S(I - 2D^*)^{-1}(\frac{2}{\hat{\epsilon}}Bq)$$

$$= \frac{2}{\hat{\epsilon}}S(I - 2D^*)^{-1}Bq$$

so that the solvation energy is given by

$$\frac{1}{2}q^T CA^{-1} Bq = \frac{1}{\hat{\epsilon}} q^T B^T (I - 2D^*)^{-T} S(I + \hat{\epsilon}D^*)^{-1} Bq$$
It is well known that \( (\text{Hsaio and Wendland}) \)

\[
SD^* = DS
\]

and

\[
S = S^{1/2} S^{1/2}
\]

which means that we can define a Hermitian operator \( H \) similar to \( D^* \)

\[
H = S^{1/2} D^* S^{-1/2}
\]

leading to an energy

\[
\frac{1}{2} q^T C A^{-1} B q = \frac{1}{\hat{\epsilon}} q^T B^T S^{1/2} (\mathcal{I} - 2H)^{-1} (\mathcal{I} + \hat{\epsilon} H)^{-1} S^{1/2} B q
\]
The spectrum of $\mathcal{D}^*$ is in $[-\frac{1}{2}, \frac{1}{2})$, and the energy is

$$\frac{1}{2} q^T CA^{-1} Bq = \sum_i \frac{1}{\hat{\epsilon}} (1 - 2\lambda_i)^{-1} (1 + \hat{\epsilon}\lambda_i)^{-1} x_i^2$$

where

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

and

$$\tilde{x} = V^T S^{1/2} Bq$$
The BIBEE approximations yield the following bounds

\[ \frac{1}{2} q^T C_{CFA}^{-1} B q = \sum_i \frac{1}{\hat{\epsilon}} (1 - 2\lambda_i)^{-1} \left( 1 - \frac{\hat{\epsilon}}{2} \right)^{-1} x_i^2 \]  
(6)

\[ \frac{1}{2} q^T C_{P}^{-1} B q = \sum_i \frac{1}{\hat{\epsilon}} (1 - 2\lambda_i)^{-1} x_i^2 \]  
(7)

\[ \frac{1}{2} q^T C_{LB}^{-1} B q = \sum_i \frac{1}{\hat{\epsilon}} (1 - 2\lambda_i)^{-1} \left( 1 + \frac{\hat{\epsilon}}{2} \right)^{-1} x_i^2 \]  
(8)

where we note that

\[ |\hat{\epsilon}| < \frac{1}{2} \]
Electrostatic solvation free energies of met-enkephalin structures

Snapshots taken from a 500-ps MD simulation at 10-ps intervals.
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

- 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

- Enabling an optimal solver
  - Fields, auxiliary operators
Why?

Impedence Mismatch in Interface

Interface is Too General:
- Solver not told about discretization data, e.g. fields
- Cannot take advantage of problem structure
  - blocking
  - saddle point structure

Interface is Too Specific:
- Assembly code specialized to each discretization
  - dimension, cell shape, hybrid
- 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
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

Vertices: 2, 3, 4, 5
Cells: 0, 1
Depth 0: 3
Depth 1: 1, 3, 4, 5
Sample Meshes
Interpolated quadrilateral mesh

Vertices
- 9
- 10
- 11
- 12
- 13
- 14

Edges
- 2
- 3
- 4
- 5
- 6
- 7
- 8

Cells
- 0
- 1

Residual Evaluation
Sample Meshes
Optimized quadrilateral mesh

Vertices:
- 0
- 1
- 2
- 3
- 4
- 5
- 6
- 7

Cells:
- 0
- 1

Depth 0

Depth 1
Sample Meshes
Interpolated tetrahedral mesh

Vertices
Edges
Faces
Cells

Depth 0
Depth 1
Depth 2
Depth 3
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

We begin with the basic covering relation,
cone(0) = \{2, 3, 4\}
reverse arrows to get the dual operation,
support(9) = \{3, 4, 6\}
add the transitive closure of the relation,

closure(0) = \{0, 2, 3, 4, 7, 8, 9\}
and the transitive closure of the dual, $\text{star}(7) = \{7, 2, 3, 0\}$
and augment with lattice operations.

\[ \text{meet}(0, 1) = \{4\} \]
and augment with lattice operations.

\[ \text{join}(8, 9) = \{4\} \]
I developed a single residual evaluation routine independent of spatial dimension, cell geometry, and finite element:

\[ F(\vec{u}) = 0 \]

<table>
<thead>
<tr>
<th>Dim</th>
<th>Cell Types</th>
<th>Discretizations</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Simplex</td>
<td>Lagrange FEM</td>
</tr>
<tr>
<td>2</td>
<td>Tensor Product</td>
<td>H(div) FEM*</td>
</tr>
<tr>
<td>3</td>
<td>Polyhedral</td>
<td>H(curl) FEM*</td>
</tr>
<tr>
<td>6†</td>
<td>Prism</td>
<td>DG FEM *‡</td>
</tr>
</tbody>
</table>

† Peter Brune, ANL
* FEniCS Project
‡ Blaise Bourdin, LSU

We have also implemented a polyhedral FVM, but this required changes to the residual evaluation.
I developed a single residual evaluation routine independent of spatial dimension, cell geometry, and finite element:

\[ F(\vec{u}) = 0 \]

<table>
<thead>
<tr>
<th>Dim</th>
<th>Cell Types</th>
<th>Discretizations</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Simplex</td>
<td>Lagrange FEM</td>
</tr>
<tr>
<td>2</td>
<td>Tensor Product</td>
<td>H(div) FEM*</td>
</tr>
<tr>
<td>3</td>
<td>Polyhedral</td>
<td>H(curl) FEM*</td>
</tr>
<tr>
<td>6†</td>
<td>Prism</td>
<td>DG FEM *†</td>
</tr>
</tbody>
</table>

† Peter Brune, ANL
* FEniCS Project
‡ Blaise Bourdin, LSU

We have also implemented a polyhedral FVM, but this required changes to the residual evaluation.
I developed a single residual evaluation routine independent of spatial dimension, cell geometry, and finite element:

\[ F(\vec{u}) = 0 \]

<table>
<thead>
<tr>
<th>Dim</th>
<th>Cell Types</th>
<th>Discretizations</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Simplex</td>
<td>Lagrange FEM</td>
</tr>
<tr>
<td>2</td>
<td>Tensor Product</td>
<td>H(div) FEM*</td>
</tr>
<tr>
<td>3</td>
<td>Polyhedral</td>
<td>H(curl) FEM*</td>
</tr>
<tr>
<td>6†</td>
<td>Prism</td>
<td>DG FEM †</td>
</tr>
</tbody>
</table>

† Peter Brune, ANL
* FEniCS Project
‡ Blaise Bourdin, LSU

We have also implemented a polyhedral FVM, but this required changes to the residual evaluation.
I developed a single residual evaluation routine independent of spatial dimension, cell geometry, and finite element:

\[ F(\bar{u}) = 0 \]

<table>
<thead>
<tr>
<th>Dim</th>
<th>Cell Types</th>
<th>Discretizations</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Simplex</td>
<td>Lagrange FEM</td>
</tr>
<tr>
<td>2</td>
<td>Tensor Product</td>
<td>H(div) FEM*</td>
</tr>
<tr>
<td>3</td>
<td>Polyhedral</td>
<td>H(curl) FEM*</td>
</tr>
<tr>
<td>6†</td>
<td>Prism</td>
<td>DG FEM *‡</td>
</tr>
</tbody>
</table>

† Peter Brune, ANL  
* FEniCS Project  
‡ Blaise Bourdin, LSU

We have also implemented a polyhedral FVM, but this required changes to the residual evaluation.
I developed a single residual evaluation routine independent of spatial dimension, cell geometry, and finite element:

\[ F(\vec{u}) = 0 \]

<table>
<thead>
<tr>
<th>Dim</th>
<th>Cell Types</th>
<th>Discretizations</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Simplex</td>
<td>Lagrange FEM</td>
</tr>
<tr>
<td>2</td>
<td>Tensor Product</td>
<td>H(div) FEM*</td>
</tr>
<tr>
<td>3</td>
<td>Polyhedral</td>
<td>H(curl) FEM*</td>
</tr>
<tr>
<td>6†</td>
<td>Prism</td>
<td>DG FEM *‡</td>
</tr>
</tbody>
</table>

† Peter Brune, ANL  
* FEniCS Project  
‡ Blaise Bourdin, LSU

We have also implemented a polyhedral FVM, but this required changes to the residual evaluation.
We consider weak forms dependent only on fields and gradients,

$$\int_{\Omega} \phi \cdot f_0(u, \nabla u) + \nabla \phi : \vec{f}_1(u, \nabla u) = 0.$$  \hspace{1em} (9)

Discretizing we have

$$\sum_{e} E_e^T \left[ B^T W^q f_0(u^q, \nabla u^q) + \sum_k D_k^T W^q \vec{f}_1^k(u^q, \nabla u^q) \right] = 0$$ \hspace{1em} (10)

- $f_n$: 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
- $E$: assembly operator
Residual Evaluation

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>
}
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

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

M. Knepley (UC)
Batch Integration
Integrate element batch

DMFIntegrateResidualFEM(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);
    }
}
Residual Evaluation

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>
        }
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>
    }
}
Residual Evaluation
Element Integration
Calculate $u_q$ and $\nabla u_q$

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)>
                <Call contributions f_0, f_1, and f_1>
            }
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];
            }
         }
      }
      <Call f_0 and f_1>
      <Loop over element vector entries (f, fc)>
      <Add contributions from 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>
}

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];
            }
        }
}
Porting to the GPU meant changing only the element integration function:

- 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
```

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

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

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

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

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

Traditional PDE codes:

- **Cannot compare different discretizations**
  - Automated FEM Discretizations for the Stokes Equation, BIT, 2008
  - Efficient Assembly of H(div) and H(curl) Conforming Finite Elements, 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, JGR, 2013

- **Run 1D, 2D, and 3D problems**
  - Ibid.

- **Enabling an optimal solver without programming**
  - Ibid.
  - Composable linear solvers for multiphysics, IPDPS, 2012
  - On the rise of strongly tilted mantle plume tails, PEPI, 2011
Resolution

Traditional PDE codes:

- Cannot compare different discretizations
  - *Efficient Assembly of H(div) and H(curl) Conforming Finite Elements*, SISC, 2009

- Compare different mesh types

- Run 1D, 2D, and 3D problems
  - Ibid.

- Enabling an optimal solver without programming
  - Ibid.
  - *Composable linear solvers for multiphysics*, IPDPS, 2012
  - *On the rise of strongly tilted mantle plume tails*, PEPI, 2011
Resolution

Traditional PDE codes:

- Cannot compare different discretizations
  - Automated FEM Discretizations for the Stokes Equation, BIT, 2008
  - Efficient Assembly of H(div) and H(curl) Conforming Finite Elements, 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, JGR, 2013

- Run 1D, 2D, and 3D problems
  - Ibid.

- Enabling an optimal solver without programming
  - Ibid.
  - Composable linear solvers for multiphysics, IPDPS, 2012
  - On the rise of strongly tilted mantle plume tails, PEPI, 2011
Resolution

Traditional PDE codes:

- Cannot compare different discretizations
  - Automated FEM Discretizations for the Stokes Equation, BIT, 2008
  - Efficient Assembly of H(div) and H(curl) Conforming Finite Elements, 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, JGR, 2013

- Run 1D, 2D, and 3D problems
  - Ibid.

- Enabling an optimal solver without programming
  - Ibid.
  - Composable linear solvers for multiphysics, IPDPS, 2012
  - On the rise of strongly tilted mantle plume tails, PEPI, 2011
Future Work

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

1. Operator Approximation
2. Residual Evaluation
3. Applications
PyLith is an open source, parallel simulator for crustal deformation problems developed by myself, Brad Aagaard, and Charles Williams. PyLith employs a finite element discretization on unstructured meshes and is built on the PETSc libraries from ANL.
Applications

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
  - 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 behavior
- **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
Applications

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
Applications

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
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
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
I wrote the first 3D Classical DFT with true hard sphere chemical potential using fundamental measure theory. It used an $O(N \log N)$ 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.

I wrote the first 3D Classical DFT with true hard sphere chemical potential using fundamental measure theory. It used an $O(N \log N)$ 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.

I wrote the first 3D Classical DFT with true hard sphere chemical potential using fundamental measure theory. It used an $O(N \log N)$ 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.

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
Applications

People Using My Mesh

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
Applications

People Using My Mesh

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
HiFlow3

- Multi-purpose finite element software
- Arose from EMCL at Karlsruhe Institute of Technology
- Flow behavior in the human respiratory system
HiFlow3

- Multi-purpose finite element software
- Arose from EMCL at Karlsruhe Institute of Technology
- Flow behavior in the human respiratory system
HiFlow3

- Multi-purpose finite element software
- Arose from EMCL at Karlsruhe Institute of Technology
- Flow behavior in the human respiratory system
TerraFERMA (Cian Wilson and Marc Spiegelman, Columbia)

- Magma Dynamics
- Flexible model builder
- Finite element
- Nested FieldSplit solver
Applications

People Using PETSc Composable Solvers

Sam Weatherley and Richard Katz (Oxford)

- Magma Dynamics
- Finite volume
- Nested FieldSplit solver
- Small scale parallel ($10^2$–$10^3$)

![Diagram of fluid model](image)

Fluid Model ($\omega/\nu_{50} = 50$)

- Compaction & Buoyancy
- Buoyancy Only

- 16% Melt
- 0.1% Melt

Porosity, $\phi/0.003$

C. R. Wilson et al.

Coupled Modeling Frontiers 2012 16
Sam Weatherley and Richard Katz (Oxford)

- Magma Dynamics
- Finite volume
- Nested FieldSplit solver
- Small scale parallel ($10^2$–$10^3$)

Fluid Model ($\omega/\nu_0 = 50$)

- Compaction & Buoyancy
  - 16% Melt
- Buoyancy Only
  - 0.1% Melt

C. R. Wilson et al. Coupled Modeling Frontiers 2012 16
PTatin (Dave May, ETHZ)
- Lithospheric and Mantle dynamics
- Finite element
- Lagrangian particles
- Nested FieldSplit solver
- Large scale parallel ($10^3$–$10^5$)

Fluid Model ($\omega/\nu_0 = 50$)
- Compaction & Buoyancy
  - 16% Melt
- Buoyancy Only
  - 0.1% Melt
Computational Leaders have always embraced the latest technology and been inspired by physical problems.
Computational Leaders have always embraced the latest technology and been inspired by physical problems.
Computational Leaders have always embraced the latest technology and been inspired by physical problems.

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

Enabling Scientific Discovery
Additional Slides
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_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
```
**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
```
**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
```
**ex55**: Allen-Cahn problem in 2D

Run flexible GMRES with 5 levels of multigrid as the preconditioner

```bash
./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

```bash
-pc_mg_galerkin
```

Use SVD as the coarse grid saddle point solver

```bash
-mg_coarse_ksp_type preonly -mg_coarse_pc_type svd
```
**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
```
**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
```
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`
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
ex55: Allen-Cahn problem in 2D

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