Refactoring Finite Element Computation

Matthew Knepley

Mathematics and Computer Science Division
Argonne National Laboratory

CDI Workshop on
Symbolic, Numeric, Algebraic Computing and Optimization
Washington D.C.
What is the Way Forward?

How do we move Scientific Computing forward?
What is the Way Forward?

How do we move Scientific Computing forward?

- Performance
  - Bandwidth on multicore chips
What is the Way Forward?

How do we move Scientific Computing forward?

- Performance
  - Bandwidth on multicore chips
- Experimentation
  - Solvers \((solved)\)
  - Elements
  - Models
What is the Way Forward?

How do we move Scientific Computing forward?

- Performance
  - Bandwidth on multicore chips
- Experimentation
  - Solvers \((solved)\)
  - Elements
  - Models
- Coupling
  - How does this interact with the discretization…
  - or solver?
Outline

1. Synergy
2. Optimizing Linear Operator Construction
3. Other Thrusts
4. Conclusions
I think the biggest problems in scientific computing are
I think the biggest problems in scientific computing are
- Lack of usable implementations of modern algorithms
  - Multigrid
  - Simplicial spectral elements
I think the biggest problems in scientific computing are

- Lack of usable implementations of modern algorithms
  - Multigrid
  - Simplicial spectral elements
- Lack of comparison among classes of algorithms
  - Meshes
  - Discretizations
I think the biggest problems in scientific computing are

- Lack of usable implementations of modern algorithms
  - Multigrid
  - Simplicial spectral elements
- Lack of comparison among classes of algorithms
  - Meshes
  - Discretizations

I was hoping that CDI would reorient thinking from

"what is the convergence rate (in $h$) of this finite element?"

"how many digits of accuracy per flop for this finite element?"

M. Knepley (ANL)
I think the biggest problems in scientific computing are

- Lack of usable implementations of modern algorithms
  - Multigrid
  - Simplicial spectral elements
- Lack of comparison among classes of algorithms
  - Meshes
  - Discretizations

I was hoping that CDI would reorient thinking from

- characterizing the solution (FEM)
  - “what is the convergence rate (in $h$) of this finite element?”
Problems

I think the biggest problems in scientific computing are

- Lack of usable implementations of modern algorithms
  - Multigrid
  - Simplicial spectral elements
- Lack of comparison among classes of algorithms
  - Meshes
  - Discretizations

I was hoping that CDI would reorient thinking from

- characterizing the solution (FEM)
  - “what is the convergence rate (in $h$) of this finite element?”
I think the biggest problems in scientific computing are

- Lack of usable implementations of modern algorithms
  - Multigrid
  - Simplicial spectral elements
- Lack of comparison among classes of algorithms
  - Meshes
  - Discretizations

I was hoping that CDI would reorient thinking from

- characterizing the solution (FEM)
  - “what is the convergence rate (in $h$) of this finite element?”

to

- characterizing the computation (Ferari)
  - “how many digits of accuracy per flop for this finite element?”
Synergy
Interaction with Systems

We have to bridge the gap with Systems to enable Scientific Computing

Database Systems

Operating Systems

Programming Languages
We have to bridge the gap with Systems to enable Scientific Computing.
Interaction with Systems

We have to bridge the gap with Systems to enable Scientific Computing

Database Systems
Datamining

Operating Systems
Distributed Computing

Programming Languages
Interaction with Systems

We have to bridge the gap with Systems to enable Scientific Computing

- Database Systems
- Datamining
- Operating Systems
- Distributed Computing
- Programming Languages
- Code Generation
I think compilers are victims of their own success (ala Rob Pike)
Future Compilers

I think compilers are victims of their own success (ala Rob Pike)

- Efforts to modularize compilers retain the same primitives
  - JIT
  - LLVM
I think compilers are victims of their own success (ala Rob Pike)

- Efforts to modularize compilers retain the same primitives
  - JIT
  - LLVM
- Raise the level of abstraction (DSL)
  - FFC, a variational form compiler
Representation Hierarchy

We can divide the work into levels:

- Model

- Algorithm

- Implementation
An example from the Spiral Project:

- DFT
- FFT
- FFTW, BLAS, ATLAS, OSKI
An example from the FEniCS Project:

- Navier-Stokes (FFC)
- FEM (FIAT)
- FErari
Optimizing Linear Operator Construction

Outline

1. Synergy

2. Optimizing Linear Operator Construction
   - Problem Statement
   - Plan of Attack
   - Results
   - Partial Geometries
   - Mixed Integer Linear Programming

3. Other Thrusts

4. Conclusions
Element integrals can be decomposed into *analytic* and *geometric* parts.

\[
\int_T \nabla \phi_i(x) \cdot \nabla \phi_j(x) \, dA = \int_T \frac{\partial \phi_i(x)}{\partial x_\alpha} \frac{\partial \phi_j(x)}{\partial x_\alpha} \, dA \tag{1}
\]

\[
= \int_{T_{ref}} \frac{\partial \phi_i(\xi)}{\partial x_\alpha} \frac{\partial \phi_j(\xi)}{\partial x_\alpha} \left| J \right| \, dA \tag{2}
\]

\[
= \frac{\partial \xi_\beta}{\partial x_\alpha} \frac{\partial \xi_\gamma}{\partial x_\alpha} \left| J \right| \int_{T_{ref}} \frac{\partial \phi_i(\xi)}{\partial \xi_\beta} \frac{\partial \phi_j(\xi)}{\partial \xi_\gamma} \, dA \tag{3}
\]

\[
= G_{\beta\gamma}(T) K_{\beta\gamma}^{ij} \tag{4}
\]

Coefficients are also put into the geometric part.
Element Matrix Formation

- Element matrix $K$ is now made up of small tensors
- Contract all tensor elements with each the geometry tensor $G(\mathcal{T})$

<p>| | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>-1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>-1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

<p>| | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>-4</td>
<td>-4</td>
<td>0</td>
<td>4</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>4</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>4</td>
<td>0</td>
<td>-4</td>
<td>-4</td>
</tr>
</tbody>
</table>

<p>| | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>3</td>
<td>-4</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>-4</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>-4</td>
<td>0</td>
<td>8</td>
<td>4</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>-4</td>
<td>0</td>
<td>0</td>
<td>4</td>
<td>8</td>
<td>-4</td>
<td>-8</td>
</tr>
</tbody>
</table>

<p>| | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>-4</td>
<td>0</td>
<td>-4</td>
<td>0</td>
<td>8</td>
<td>4</td>
<td>-8</td>
</tr>
<tr>
<td>-4</td>
<td>0</td>
<td>0</td>
<td>4</td>
<td>-8</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>4</td>
<td>0</td>
<td>0</td>
<td>-4</td>
<td>8</td>
<td>4</td>
</tr>
<tr>
<td>-8</td>
<td>-4</td>
<td>4</td>
<td>8</td>
<td>-4</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

<p>| | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>-4</td>
<td>-4</td>
<td>-4</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>-4</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>-4</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
Element matrix $K$ can be precomputed
- FFC

Can be extended to nonlinearities and curved geometry

Many redundancies among tensor elements of $K$
- Could try to optimize the tensor contraction...
Given vectors $v_i \in \mathbb{R}^m$, minimize $\text{flops}(v^Tg)$ for arbitrary $g \in \mathbb{R}^m$

- The set $v_i$ is not at all random
- Not a traditional compiler optimization
- How to formulate as an optimization problem?
If $v_{i}^{T}g$ is known, is $\text{flops}(v_{j}^{t}g) < 2m - 1$?

We can use binary relations among the vectors:

- **Equality**
  - If $v_{j} = v_{i}$, then $\text{flops}(v_{j}^{t}g) = 0$

- **Colinearity**
  - If $v_{j} = \alpha v_{i}$, then $\text{flops}(v_{j}^{t}g) = 1$

- **Hamming distance**
  - If $\text{dist}_H(v_{j}, v_{i}) = k$, then $\text{flops}(v_{j}^{t}g) = 2k$
Algorithm for Binary Relations

- Construct a weighted graph on $v_i$
  - The weight $w(i,j)$ is $\text{flops}(v_j^Tg)$ given $v_i^Tg$
  - With the above relations, the graph is symmetric

- Find a minimum spanning tree
  - Use Prim or Kruskal for worst case $O(n^2 \log n)$

- Traverse the MST, using the appropriate calculation for each edge
  - Roots require a full dot product
Coplanarity

- Ternary relation
  - If \( v_k = \alpha v_i + \beta v_j \), then \( \text{flops}(v_k^T g) = 3 \)
  - Does not fit our undirected graph paradigm

- Use a hypergraph?
  - No good MST algorithm

- Appeal to geometry?
## Preliminary Results

<table>
<thead>
<tr>
<th>Order</th>
<th>Entries</th>
<th>Base MAPs</th>
<th>FErari MAPs</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>6</td>
<td>24</td>
<td>7</td>
</tr>
<tr>
<td>2</td>
<td>21</td>
<td>84</td>
<td>15</td>
</tr>
<tr>
<td>3</td>
<td>55</td>
<td>220</td>
<td>45</td>
</tr>
<tr>
<td>4</td>
<td>120</td>
<td>480</td>
<td>176</td>
</tr>
<tr>
<td>5</td>
<td>231</td>
<td>924</td>
<td>443</td>
</tr>
<tr>
<td>6</td>
<td>406</td>
<td>1624</td>
<td>867</td>
</tr>
</tbody>
</table>
Partial Geometry

Given a set $V$ and a set of lines $L \subseteq \mathcal{P}(V)$, $(V, L)$ is a partial geometry if

- there is at most one line through each pair of points
- each line has at least three point

Note that

- Typical geometries have exactly one line through each pair of points
- Encoded by ternary relations, like coplanarity, which satisfy
  
  $$R(x, y, z) \land R(y, z, p) \Rightarrow R(x, y, p) \land R(y, z, p)$$

- Generalizes to higher arity relations
Definitions

- **vertex**
  - A point lying in two or more lines

- **closure**
  - The transitive closure $\bar{S}$ under $R$ of some $S \subset V$
  - $z \in V \land \exists x, y \in S \exists R(x, y, z) \Rightarrow z \in \bar{S}$

- **independent set**
  - A set $S$ such that for any $S' \subset S$, $\bar{S}' \neq S$

- **basis**
  - An independent set $S$ such that $\bar{S} = V$
Goal

We want a basis of minimal cost, which now means size.

- Something like a “minimum spanning hypertree”
- Closure operation produces a DAG
  - Use topological sort to get computation sequence
- Complexity is unknown
- Unfortunate example shows bases of differing size
  - At odds with matroid theory
Geometric Reduction

- Eliminate *parallel* lines (no vertices)
  - Can add any two points on the line to a minimal basis
- Eliminate single vertex lines
  - Can add any non-vertex on the line to a minimal basis
- Eliminate non-vertices from basis
  - Each line has at least two vertices
    - If two vertices are already present, discard point
    - Otherwise, switch with a vertex
  - The generated set is the same, and the size has not increased
We want to show that all reduced bases are the same size.

- Remove a vertex $p$ from the basis $B$
  - Now there is a set $Ex(p)$ which is no longer in $\bar{B}$
- Choose $q$ from $Ex(p)$
- Reverse the generation path from $p$ to $q$
  - If we generate $p$, we generate all of $Ex(p)$

Now

- We have an easy algorithm for a minimal basis
- Matroid results apply
Modeling the Problem

- **Objective**: is cost of dot products (tensor contractions in FEM)
  - Set of vectors $V$ with a given arbitrary vector $g$

- The original MINLP has a nonconvex, nonlinear objective

- Reformulate to obtain a MILP using auxiliary binary variables
Modeling the Problem

Variables

\( \alpha_{ij} = \) Basis expansion coefficients
\( y_i = \) Binary variable indicating membership in the basis
\( s_{ij} = \) Binary variable indicating nonzero coefficient \( \alpha_{ij} \)
\( z_{ij} = \) Binary variable linearizes the objective function (equivalent to \( y_i y_j \))
\( U = \) Upper bound on coefficients

Constraints

Eq. (6b) : Basis expansion
Eq. (6c) : Exclude nonbasis vector from the expansion
Eq. (6d) : Remove offdiagonal coefficients for basis vectors
Eq. (??) : Exclude vanishing coefficients from cost
MINLP Model

minimize \[ \sum_{i=1}^{n} \left\{ y_i (2m - 1) + (1 - y_i) \left( 2 \sum_{j=1, j \neq i}^{n} y_j - 1 \right) \right\} \]  \hspace{1cm} (6a)

subject to \[ v_i = \sum_{j=1}^{n} \alpha_{ij} v_j \] \hspace{1cm} i = 1, \ldots, n \hspace{1cm} (6b)

\[ -Uy_j \leq \alpha_{ij} \leq Uy_j \] \hspace{1cm} i = 1, \ldots, n, j \hspace{1cm} (6c)

\[ -U(1 - y_i) \leq \alpha_{ij} \leq U(1 - y_i) \] \hspace{1cm} i = 1, \ldots, n, j \hspace{1cm} (6d)

\[ y_i \in \{0, 1\} \] \hspace{1cm} i = 1, \ldots, n \hspace{1cm} (6e)
Optimizing Linear Operator Construction

Mixed Integer Linear Programming

Original Formulation

Equivalent MILP Model: \( z_{ij} = y_i \cdot y_j \)

\[
\text{minimize} \quad 2m \sum_{i=1}^{n} y_i + 2 \sum_{i=1}^{n} \sum_{j=1, j \neq i}^{n} (y_j - z_{ij}) - n \quad (6a)
\]

subject to

\[
v_i = \sum_{j=1}^{n} \alpha_{ij} v_j \quad i = 1, \ldots, n \quad (6b)
\]

\[-Uy_j \leq \alpha_{ij} \leq Uy_j \quad i = 1, \ldots, n, \ j = 1, \ldots, n \quad (6c)
\]

\[-U(1 - y_i) \leq \alpha_{ij} \leq U(1 - y_i) \quad i = 1, \ldots, n, \ j = 1, \ldots, n \quad (6d)
\]

\[z_{ij} \leq y_i, \quad z_{ij} \leq y_j, \quad z_{ij} \geq y_i + y_j - 1, \quad i = 1, \ldots, n, \ j = 1, \ldots, n \quad (6e)
\]

\[y_i \in \{0, 1\}, \quad z_{ij} \in \{0, 1\} \quad i = 1, \ldots, n, \ j = 1, \ldots, n \quad (6f)
\]
Sparse Coefficient Formulation

- Take advantage of sparsity of $\alpha_{ij}$ coefficient
- Introduce binary variables $s_{ij}$ to model existence of $\alpha_{ij}$
- Add constraints $-Us_{ij} \leq \alpha_{ij} \leq Us_{ij}$
MINLP Model

\[
\text{minimize} \quad \sum_{i=1}^{n} \left\{ y_i (2m - 1) + (1 - y_i) \left( 2 \sum_{j=1,j \neq i}^{n} s_{ij} - 1 \right) \right\} 
\]

subject to

\[
v_i = \sum_{j=1}^{n} \alpha_{ij} v_j \quad i = 1, \ldots, n
\]

\[-U s_{ij} \leq \alpha_{ij} \leq U s_{ij} \quad i = 1, \ldots, n, j
\]

\[-U (1 - y_i) \leq \alpha_{ij} \leq U (1 - y_i) \quad i = 1, \ldots, n, j
\]

\[s_{ij} \leq y_j \quad i = 1, \ldots, n, j
\]

\[y_i \in \{0, 1\}, \quad s_{ij} \in \{0, 1\} \quad i = 1, \ldots, n, j
\]
Sparse Coefficient Formulation

Equivalent MILP Model

\[
\begin{align*}
\text{minimize} & \quad 2m \sum_{i=1}^{n} y_i + 2 \sum_{i=1}^{n} \sum_{j=1,j \neq i}^{n} (s_{ij} - z_{ij}) - n \\
\text{subject to} & \quad v_i = \sum_{j=1}^{n} \alpha_{ij} v_j \\
& \quad -Us_{ij} \leq \alpha_{ij} \leq Us_{ij} \\
& \quad -U(1 - y_i) \leq \alpha_{ij} \leq U(1 - y_i) \\
& \quad z_{ij} \leq y_i, \quad z_{ij} \leq s_{ij}, \quad z_{ij} \geq y_i + s_{ij} - 1, \\
& \quad y_i \in \{0, 1\}, \quad z_{ij} \in \{0, 1\}, \quad s_{ij} \in \{0, 1\}
\end{align*}
\] (7a)

(7b)

(7c)

(7d)

(7e)

(7f)
Results

Initial Formulation

- Initial formulation only had sparsity in the $\alpha_{ij}$
- MINTO was not able to produce some optimal solutions
  - Report results after 36000 seconds

<table>
<thead>
<tr>
<th>Elements</th>
<th>Default Flops</th>
<th>MILP Flops</th>
<th>MILP LPs</th>
<th>MILP CPU</th>
<th>Sparse Coef. MILP Flops</th>
<th>Sparse Coef. MILP LPs</th>
<th>Sparse Coef. MILP CPU</th>
</tr>
</thead>
<tbody>
<tr>
<td>$P_1$ 2D</td>
<td>42</td>
<td>42</td>
<td>33</td>
<td>0.10</td>
<td>34</td>
<td>187</td>
<td>0.43</td>
</tr>
<tr>
<td>$P_2$ 2D</td>
<td>147</td>
<td>147</td>
<td>2577</td>
<td>37.12</td>
<td>67</td>
<td>6030501</td>
<td>36000</td>
</tr>
<tr>
<td>$P_1$ 3D</td>
<td>170</td>
<td>166</td>
<td>79</td>
<td>0.49</td>
<td>146</td>
<td>727</td>
<td>3.97</td>
</tr>
<tr>
<td>$P_2$ 3D</td>
<td>935</td>
<td>935</td>
<td>25283</td>
<td>36000</td>
<td>829</td>
<td>33200</td>
<td>36000</td>
</tr>
</tbody>
</table>
Formulation with Sparse Basis

- We can also take account of the sparsity in the basis vectors
- Count only the flops for nonzero entries
  - Significantly decreases the flop count

<table>
<thead>
<tr>
<th>Elements</th>
<th>Sparse Coefficient Flops</th>
<th>Sparse Basis Flops</th>
</tr>
</thead>
<tbody>
<tr>
<td>$P_1$ 2D</td>
<td>34</td>
<td>12</td>
</tr>
<tr>
<td>$P_1$ 3D</td>
<td>146</td>
<td>26</td>
</tr>
</tbody>
</table>
Outline

1. Synergy
2. Optimizing Linear Operator Construction
3. Other Thrusts
4. Conclusions
Sieve is an interface for
- general topologies
- functions over these topologies (bundles)
- traversals

One relation handles all hierarchy
- Vast reduction in complexity
  - Dimension independent code
  - A single communication routine to optimize
- Expansion of capabilities
  - Partitioning and distribution
  - Hybrid meshes
  - Complicated structures and embedded boundaries
  - Unstructured multigrid
Finite Element Integrator and Tabulator by Rob Kirby

http://www.fenics.org/fiat

FIAT understands

- Reference element shapes (line, triangle, tetrahedron)
- Quadrature rules
- Polynomial spaces
- Functionals over polynomials (dual spaces)
- Derivatives

User can build arbitrary elements by specifying the Ciarlet triple \((K, P, P')\)

FIAT is part of the FEniCS project, as is the PETSc Sieve module
FFC is a compiler for variational forms.

Here is a mixed-form Poisson equation:

\[ a((\tau, w), (\sigma, u)) = L((\tau, w)) \quad \forall (\tau, w) \in V \]

where

\[ a((\tau, w), (\sigma, u)) = \int_{\Omega} \tau \sigma - \nabla \cdot \tau u + w \nabla \cdot u \, dx \]

\[ L((\tau, w)) = \int_{\Omega} wf \, dx \]
FFC is a compiler for variational forms.

\[ BDM1 = \text{FiniteElement}("\text{Brezzi-Douglas-Marini}", "\text{triangle}", 1) \]
\[ DG0 = \text{FiniteElement}("\text{Discontinuous Lagrange}", "\text{triangle}", 0) \]

\[ \text{element} = BDM1 + DG0 \]
\[ (\tau, w) = \text{TestFunctions}(\text{element}) \]
\[ (\sigma, u) = \text{TrialFunctions}(\text{element}) \]

\[ f = \text{Function}(DG0) \]

\[ a = (\text{dot}(\tau, \sigma) - \text{div}(\tau)u + w\text{div}(...))\text{dx} \]
\[ L = w*f\text{dx} \]
FFC is a compiler for variational forms.

Here is a discontinuous Galerkin formulation of the Poisson equation:

\[ a(v, u) = L(v) \quad \forall v \in V \]

where

\[
\begin{align*}
a(v, u) &= \int_{\Omega} \nabla u \cdot \nabla v \, dx \\
&\quad + \sum_S \int_{S} - <\nabla v > \cdot [[u]]_n - [[v]]_n \cdot <\nabla u > - (\alpha/h)vu \, dS \\
&\quad + \int_{\partial \Omega} -\nabla v \cdot [[u]]_n - [[v]]_n \cdot \nabla u - (\gamma/h)vu \, ds \\
L(v) &= \int_{\Omega} vf \, dx
\end{align*}
\]
FFC is a compiler for variational forms.

\[
\begin{align*}
\text{DG1} &= \text{FiniteElement}("\text{Discontinuous Lagrange}\), \ "\text{triangle}\), 1) \\
v &= \text{TestFunctions}(\text{DG1}) \\
u &= \text{TrialFunctions}(\text{DG1}) \\
f &= \text{Function}(\text{DG1}) \\
g &= \text{Function}(\text{DG1}) \\
n &= \text{FacetNormal}"\text{triangle}\) \\
h &= \text{MeshSize}"\text{triangle}\) \\
a &= \text{dot}(\text{grad}(v), \ \text{grad}(u))\ast dx \\
&\quad - \text{dot}(\text{avg}(\text{grad}(v)), \ \text{jump}(u, n))\ast dS \\
&\quad - \text{dot}(\text{jump}(v, n), \ \text{avg}(\text{grad}(u)))\ast dS \\
&\quad + \alpha/h\ast \text{dot}(\text{jump}(v, n) + \text{jump}(u, n))\ast dS \\
&\quad - \text{dot}(\text{grad}(v), \ \text{jump}(u, n))\ast ds \\
&\quad - \text{dot}(\text{jump}(v, n), \ \text{grad}(u))\ast ds \\
&\quad + \gamma/h\ast v\ast u\ast ds \\
L &= v\ast f\ast dx + v\ast g\ast ds
\end{align*}
\]
Outline

1. Synergy
2. Optimizing Linear Operator Construction
3. Other Thrusts
4. Conclusions
Conclusions

Better mathematical abstractions bring concrete benefits

- Vast reduction in complexity
  - Declarative, rather than imperative, specification
  - Dimension independent code

- Opportunities for optimization
  - Higher level operations missed by traditional compilers
  - Single communication routine to optimize

- Expansion of capabilities
  - Easy model definition
  - Arbitrary elements
  - Complex geometries and embedded boundaries