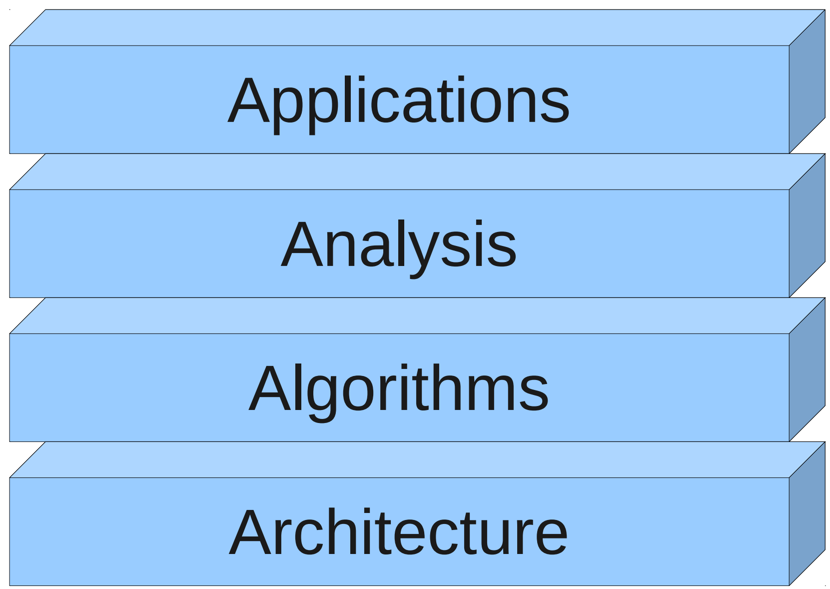
Metanumerical Computing for PDE: Accomplishments and Opportunities for High-Level Finite Element Tools

Robert C. Kirby
Texas Tech University

WOLFHPC May 31, 2011

The Stack



Or is it a graph...?

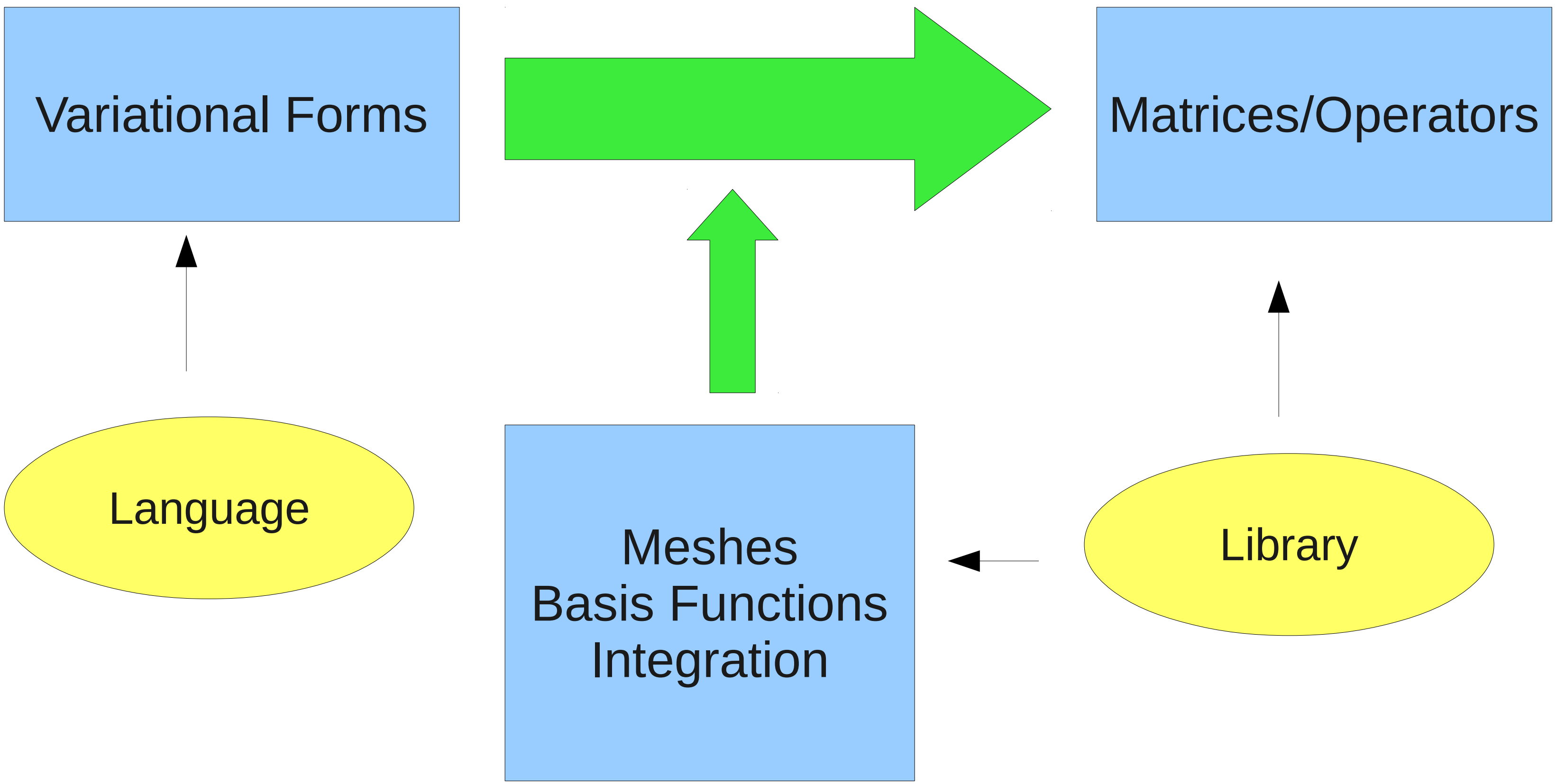
K's Recursive Conundrum

- Computers were invented to automate tedious and error-prone tasks
- Programming a computer is a tedious and error-prone task
- So get a computer to do it

Metanumerical Computing: Using high-level mathematical structure to generate, reason about, manipulate, and/or optimize numerical code

Pieces to consider

$$\int_\Omega \nabla u \cdot \nabla v - f v \, dx \qquad\qquad Ax - b$$



Library approach (e.g. Deal.II)

- 2007 Wilkinson Award Winner
- Library of basis functions, quadrature, meshes, degrees of freedom, etc
  - All codes require these pieces
  - Reduce programmer time for *hp* adaptivity
- “High-level”, but no fancy automation

Language approach

- Natural grammar for variational forms
- Enumeration? 
- Language:
  - DSL (Analysa, FreeFEM)
  - DSEL (Life, Sundance, FEniCS)

$$\int_\Omega \nabla u \cdot \nabla v \, dx$$

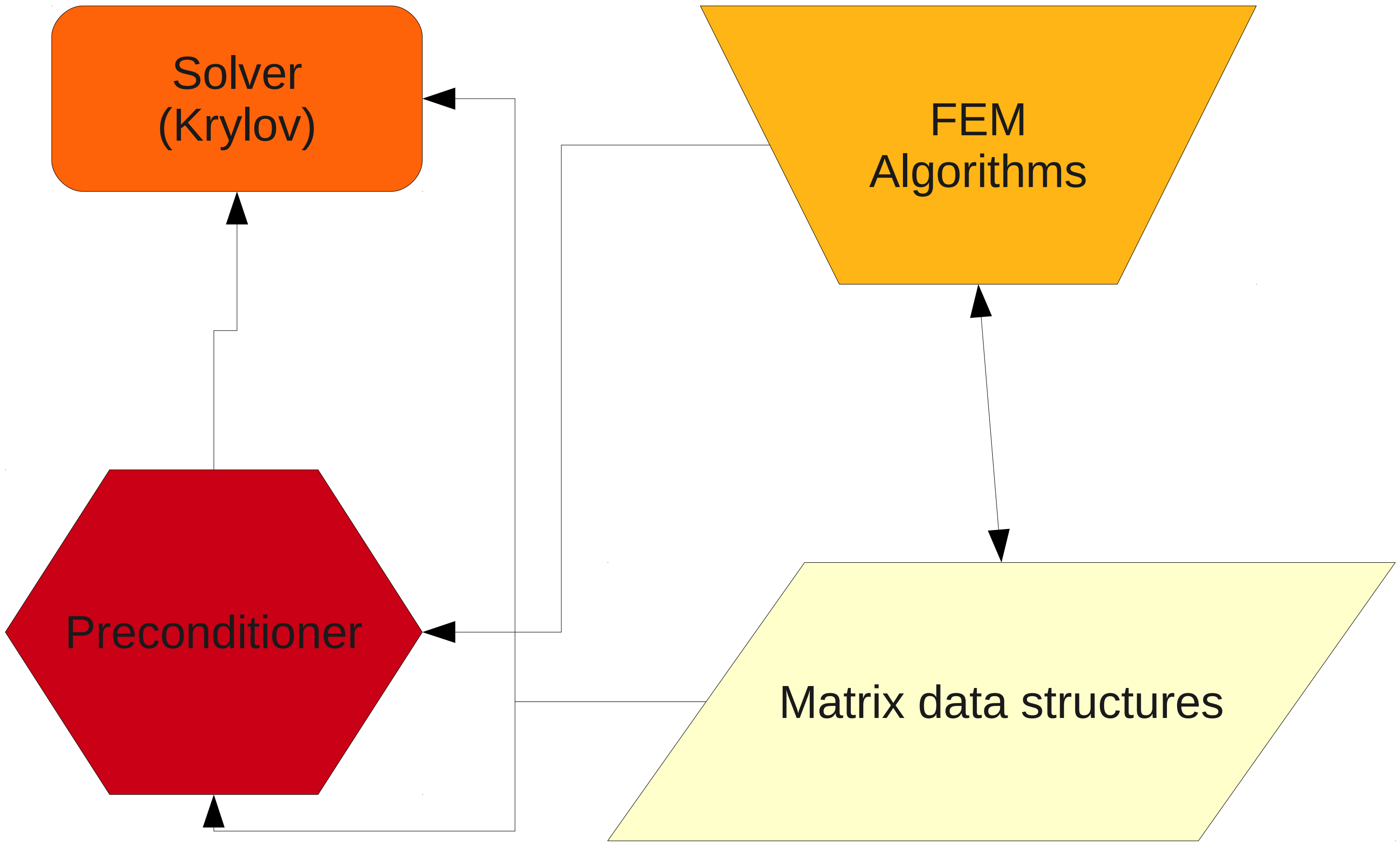
$$\int_\Omega w \cdot \nabla u \cdot v \, dx$$

$$\int_\Omega (\nabla \times u) \cdot (\nabla \times v) \; dx$$

Performance Challenges

- Data locality (global)
  - Mesh entities (unstructured)
  - Sparse matrices
- Dense calculations:
  - Elementwise operations
- Interactions:
  - App, solver, PC, FEM algorithms

Interactions



Some PDE projects

Sundance

- C++ Run-time
- Trilinos
- “All differentiation”

FEniCS

- Tool suite
- Generate C++
- Python top-level

Goals for automation

- Developer perspective:
  - Reliability, Time-to-code
- Application perspective:
  - “Works”, Feature set, Time-to-answer
- Hardware perspective
  - Flexibility to target hardware (MPI, CUDA, etc)

Sundance basic overview

- General form for FE variational problems

$$G[u,v] = \sum_r \int_{\Omega_r} \mathcal{G}_r\left(\{D_\alpha v\}_\alpha, \{D_\beta u\}_\beta, x\right)\, d\mu_r = 0, \quad v \in V$$

- Algebraic system defined by derivatives

$$u_h = \sum_{i=1}^{N} u_i \phi_i, \quad v_h = \sum_{i=1}^{N} v_i \psi_i$$

$$G[u_h, v_h] = 0, \quad v \in V_h \leftrightarrow \frac{\partial G}{\partial v_i} = 0, \quad 1 \le i \le N$$

Sundance AD

- “Low-level” code never generated
- Weak form expression graph analyzed at run-time
- Form evaluation mapped to Evaluation Engine kernels for operators

FeniCS Overview

- Begun 2003 (RCK, Logg, Hoffman, …)
- Collection of tools
  - Form compilers (ffc, syfc)
  - Optimizers (FErari)
  - Basis functions (FIAT)
  - Meshes, Linear algebra (DOLFIN)
  - Vis (Viper)
- Relies on generating code from UFL

FEniCS Code Generation

- AST represented in Python (embedded)
- Form analyzed (similar canonical form to Sundance)
- Tensor vs. Quadrature
- C++ generated for element matrix assembly
- Link against DOLFIN
- See K,Logg (ACM TOMS 2005-6) and also Oelgaard, Wells, Rognes

Code snippet (Sundance)

```
Expr source=exp(u);
Expr eqn
  = Integral(interior, (grad*u)*(grad*v)+v*source, quad4);
```

Code snippet (ffc)

```
a = inner(grad(u),grad(v))*dx - exp(u)*v*dx
```

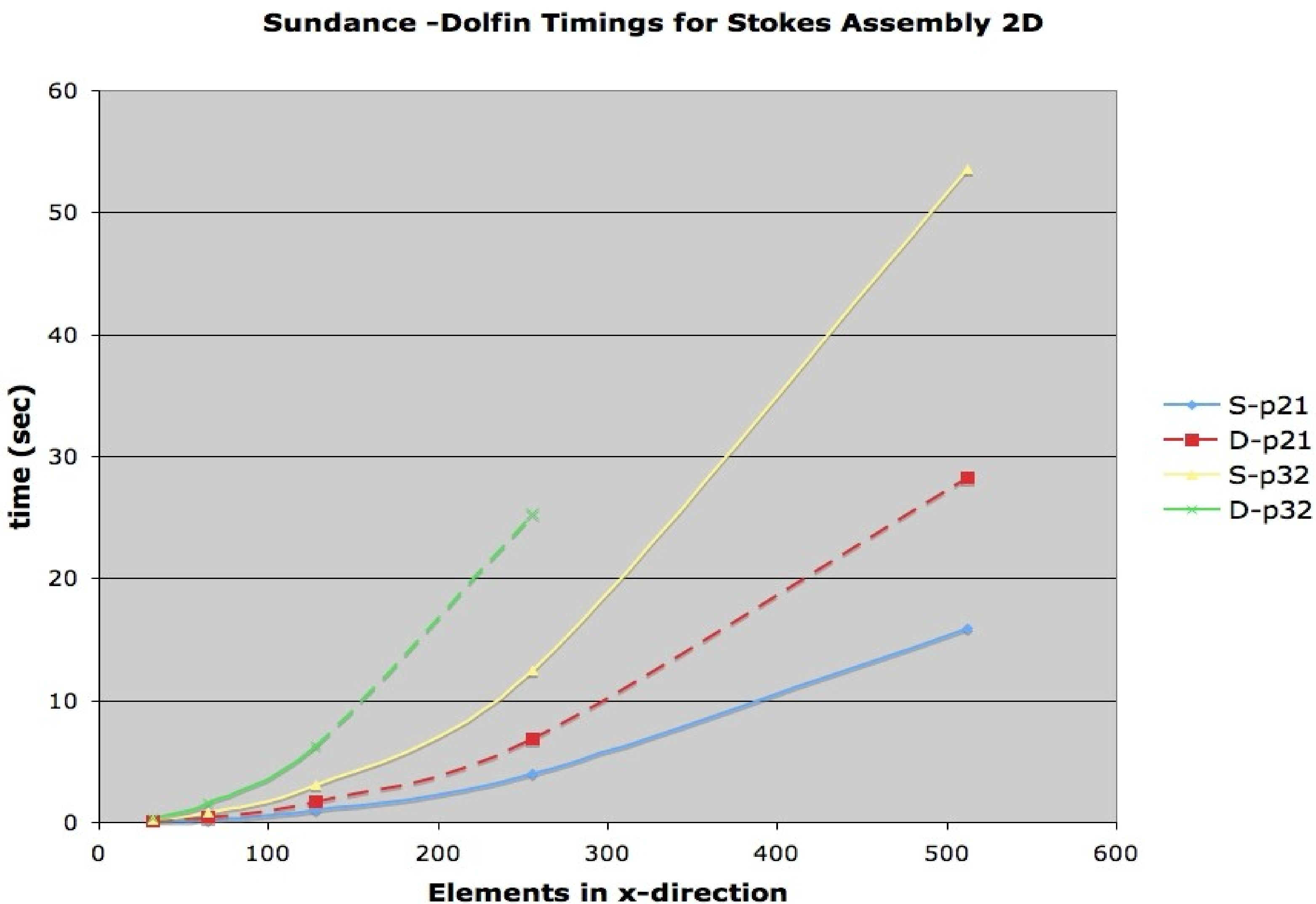
Let's look at some code

- Given demos:
  - Sundance: Poisson-Boltzman
  - DOLFIN: Nonlinear Poisson
- These are documented/distributed
- Break to shell...

Where do they shine?

- Sundance:
  - Performance (Parallel & serial)
  - Mathematical framework
  - UQ
- FeniCS:
  - FEM features (DG, H(div), H(curl), etc)
  - Active global user community (+Ubuntu)
  - Integrated Python environment
  - Standards (Rathberger → CUDA)

Sundance versus FEniCS



Matrix-free?

- 16x16x16 hex mesh, assemble Poisson

Degree	PreAlloc,OneMat	Apply	Mat-free GEMM	Mat-free tensor
1	3.24E-002	3.66E-004	7.84E-004	4.73E-003
2	6.36E-001	3.83E-003	3.97E-003	1.71E-002
3	7.30E+000	1.13E-002	1.05E-002	4.88E-002
4	3.04E+002	6.56E-001	2.85E-002	9.54E-002

Insert >> construct >> apply

Matrix-free algorithms?

Preconditioners?

Matrix Construction

- Assume mesh & DOF done “right”
- All work is in element matrix construction

$$A_{T,ij} = \int_T D_1 \phi_i D_2 \psi_j \, dx$$

Matrices of basis functions at
quadrature points: local density

Optimizing Matrix Construction

DGEMM

- Element computations batched
- Use a library
- Coarse-grained

FErari

- Discrete structure in matrix construction
- Joint with Knepley, Logg, Scott, Terrel
- For each basis, degree, form, generate specific code
- Fine-grained

Ongoing work: Matrix-Free

- Reduced costs

	Basic	Spectral
Work per cell	$\mathcal{O}(n^{2d})$	$\mathcal{O}(n^{d+1})$
Mem per cell	$\mathcal{O}(n^{2d})$	$\mathcal{O}(n^{d})$

- Separation of concerns:

$$A = \mathcal{A}^t A_e \mathcal{A}$$

- Manycore possibilities

Towards Manycore?

- Sundance (Arch-neutral interface)
  - Intrepid/Kokkos
- FFC (Arch-specific back-ends)
  - Rathberger, et al – generate CUDA from UFL (preliminary)
- Bernstein polynomials:
  - RCK (Numerische, 2010), RCK+Kieu (submitted), Ainsworth
  - High order, simplex, spectral complexity, de Rham complex!

Conclusions

- Successes of PDE automation:
  - Map variational forms onto code onto algorithms
  - User experience
  - Reasonable – good performance
- Ongoing challenges:
  - Architecture-awareness
  - New architecture → new algorithms?
  - Portability to new platforms