PECOS

Predictive Engineering and Computational Sciences

Camellia

A Discontinuous Petrov-Galerkin Toolbox Using Trilinos

Nate Roberts

The University of Texas at Austin

March 2-3, 2012

Acknowledgments

Collaborators:

- Leszek Demkowicz (UT/PECOS)
- Denis Ridzal (Sandia/CSRI)
- Pavel Bochev (Sandia/CSRI)
- Jesse Chan (UT/PECOS)

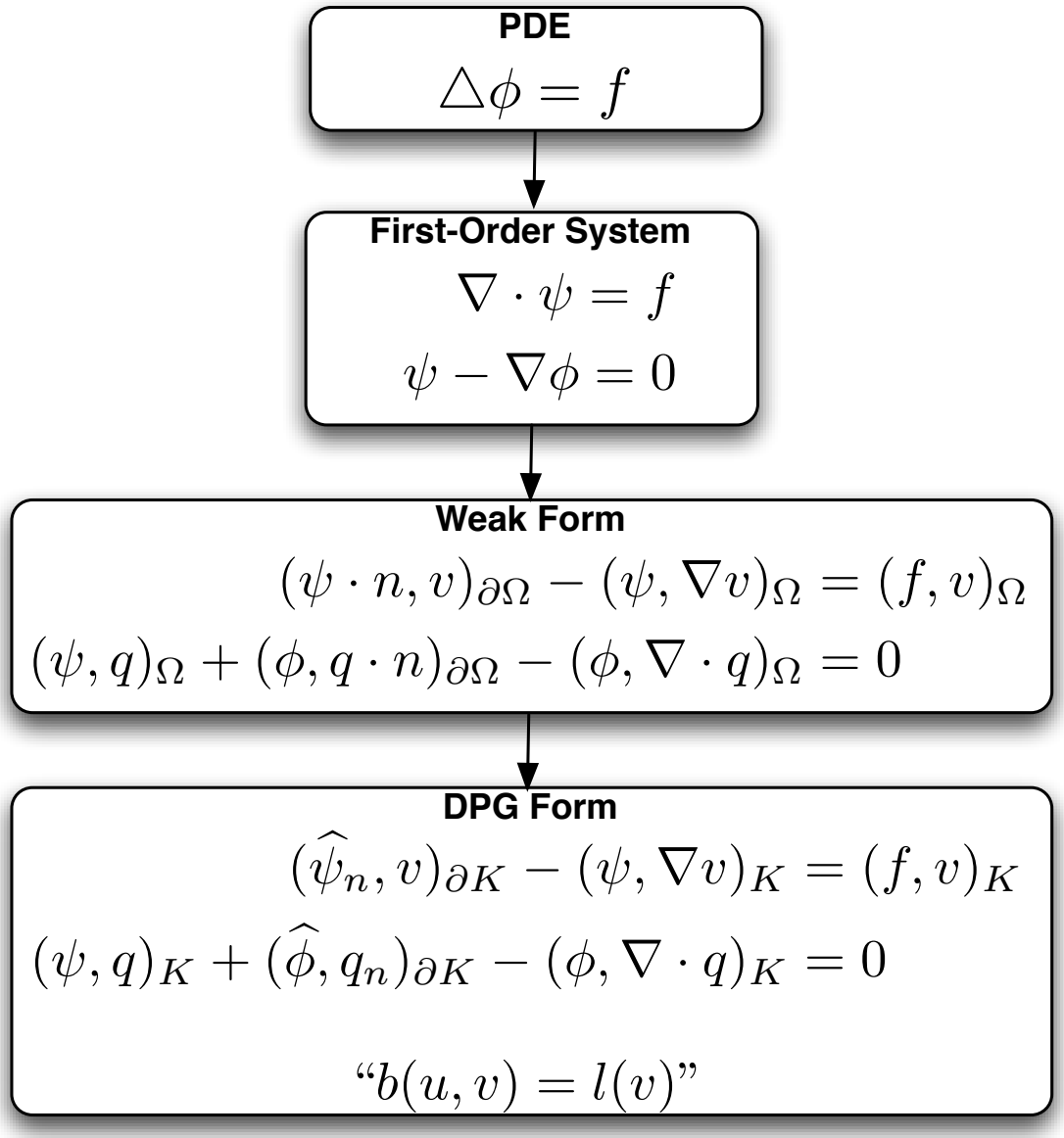
Support:

- Sandia (summer internships)
- PECOS (GRA funding)
- ICES (CSEM Fellowship)

Outline

1. DPG Intro
2. Goals for Camellia
3. Selected Design Ideas within Camellia
4. A Performance Issue, and its Resolution
5. Current and Future Work

From Strong-Form PDE to DPG Form



Solving with DPG

Continuous Test Space

DPG Form

$$b(u_h, v) = l(v)$$

Optimal Test Functions

For each $u \in U_h$, find
$v_u \in V : (v_u, w)_V = b(u, w) \forall w \in V$

Discrete Test Space

DPG Form

$$b(u_h, v_h) = l(v_h)$$

Optimal Test Functions

For each $u \in U_h$, find
$v_u \in V_{p+\triangle p} : (v_u, w)_V = b(u, w)$
$\forall w \in V_{p+\triangle p}$

Stiffness Matrix

$$K_{ij} = b(e_i, v_{e_j}) = (v_{e_i}, v_{e_j})_V = (v_{e_j}, v_{e_i})_V = b(e_j, v_{e_i}) = K_{ji}$$

Optimality

$$||u - u_h||_E \leq ||u - w_h||_E \quad \forall w_h \in U_h$$

$$\left(||u||_E = \sup_{||v||_V = 1} b(u, v) = ||v_u||_V \right)$$

Goals for Camellia

Goals for Camellia (Achieved)

- Define $b(u, v)$ in the *continuous* space (separation of concerns)
- Arbitrary, hp-adaptive 2D meshes (quads and triangles)
- "Reasonable" speed and scalability
- Provide prebuilt versions of inner products commonly used in DPG research (mathematician's and quasi-optimal)

Goals for Camellia (Aspirational)

- Support for nonlinear PDEs (Navier-Stokes in particular)
- Better scalability
- Longer term: 3D meshes

Trilinos Support

Provided by Intrepid

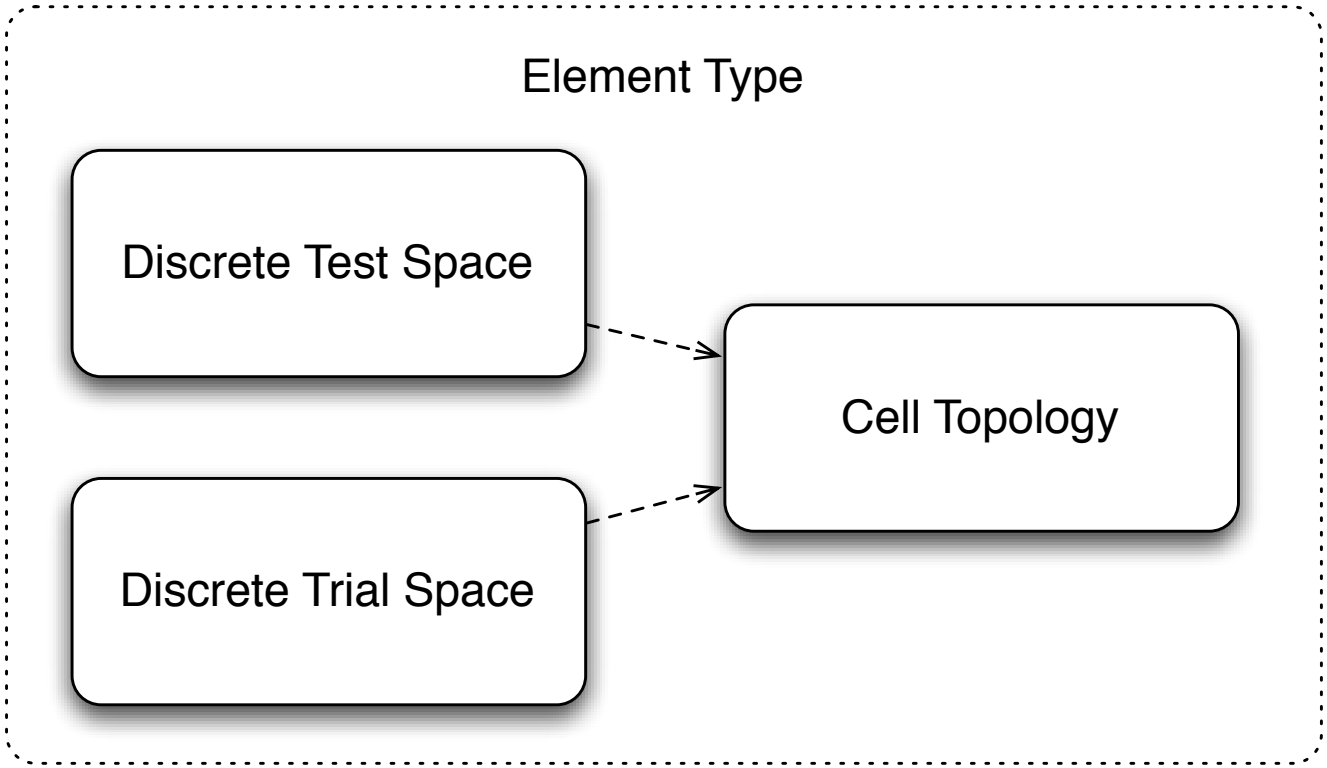
- Conforming basis functions
- Basis value transformations from reference to physical space
- Numerical integration facilities

The Intrepid facilities operate on *batches* of like elements.

Provided by Epetra

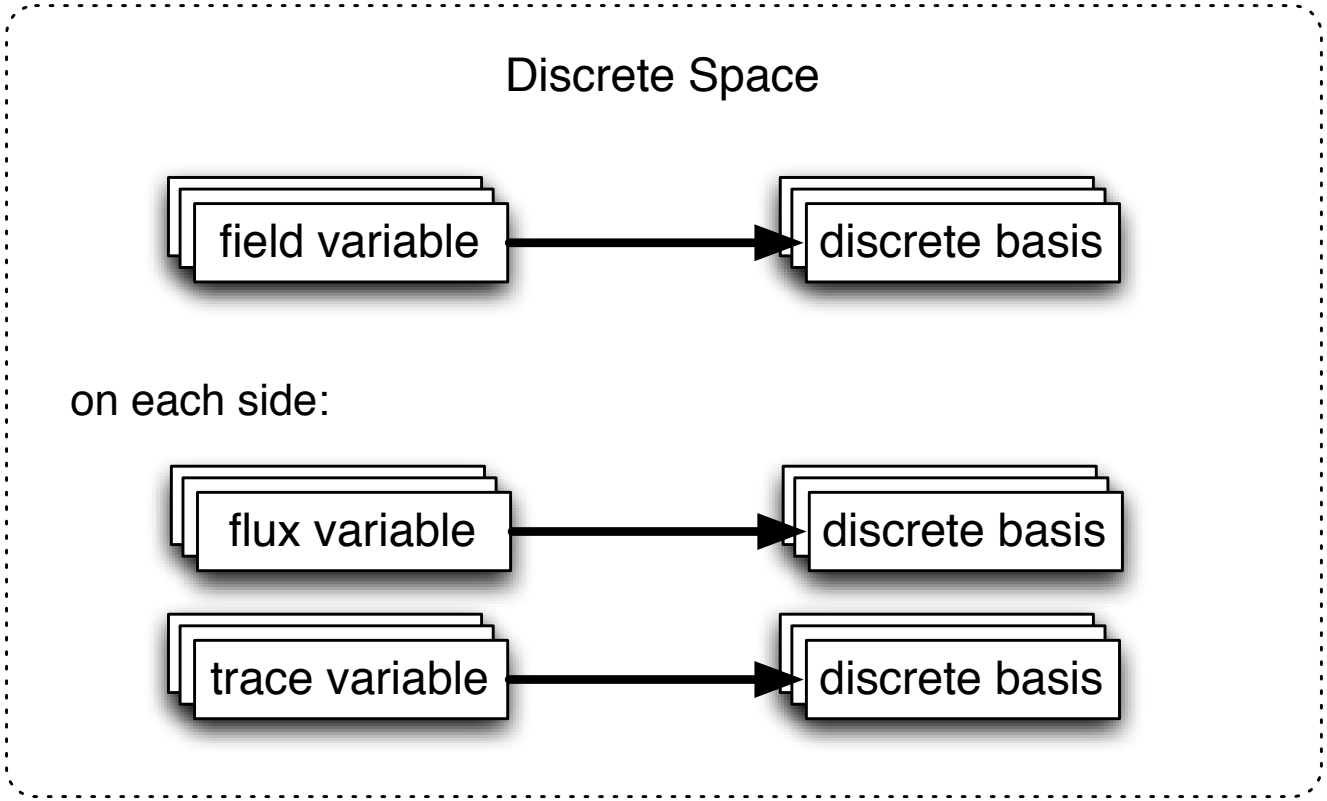
- Distributed storage types (including Epetra::FECrsMatrix)
- Abstract linear solver interfaces (easy to switch solvers)

Design Idea: Element Type



To take maximal advantage of Intrepid's support for batches of elements while operating on a non-uniform mesh, we introduce the notion of the element type. Elements of the same type agree in terms of their discrete test and trial spaces.

Design Idea: Discrete Space



The discrete spaces, in turn, are defined by mappings from variables to the bases used to represent them.

Design Pattern: Factory

Each element has a type; each type has a pair of discrete spaces; each discrete space is made up of several bases. How do we keep this design from taking a prohibitive amount of memory?

For this, among other considerations: want a way to guarantee that we only create one copy of each basis, discrete space, and element type. This is known as a *Factory* in the design pattern literature.

Create Factories for:

- Basis
- Discrete Space
- Element Type

Performance Story: Quasi-Optimal Inner Product

The "optimal test norm"—a choice of test space norm that deliver best approximation in $||\cdot||_U$—is given by

$$||v||_V \stackrel{\text{def}}{=} \sup_{||u||_U=1} b(u,v).$$

- When $U = L^2$, can determine analytically using Cauchy-Schwarz.
- Approximate with "localized" version, the *quasi-optimal* test norm.
- Camellia can determine quasi-optimal test norm automatically.

However, when I first implemented this and used it in place of the "mathematician's" norm, I observed a dramatic reduction in overall speed. Why?

Performance Story: Quasi-Optimal Inner Product

The mathematician's norm for my Stokes formulation is:

$$||(\boldsymbol{q}_1, \boldsymbol{q}_2, v_1, v_2, v_3)||_V^2 \stackrel{\text{def}}{=} \sum_{i=1}^{2} ||\boldsymbol{q}_i||_{H(\text{div})}^2 + \sum_{i=1}^{3} ||v_i||_{H(\text{grad})}^2$$

Compare the quasi-optimal test norm:

$$\left|\left|\frac{q_{11}}{2\mu} + \frac{\partial}{\partial x} v_1\right|\right|_{L^2}^2 + \left|\left|\frac{q_{11}}{2\mu} + \frac{q_{22}}{2\mu}\right|\right|_{L^2}^2 + \left|\left|\frac{q_{12}}{2\mu} + \frac{q_{21}}{2\mu} + \frac{\partial}{\partial y} v_1 + \frac{\partial}{\partial x} v_2\right|\right|_{L^2}^2$$
$$+ \left|\left|\frac{q_{22}}{2\mu} + \frac{\partial}{\partial y} v_2\right|\right|_{L^2}^2 + ||q_{12} - q_{21}||_{L^2}^2 + \left|\left|\nabla \cdot \boldsymbol{q}_1 - \frac{\partial}{\partial x} v_3\right|\right|_{L^2}^2$$
$$+ \sum_{i=1}^{2} ||\boldsymbol{q}_i||_{L^2}^2 + \sum_{i=1}^{3} ||v_i||_{L^2}^2$$

Design Idea: BasisCache

- Discrete spaces $\implies$ quadrature points.
- Will use each basis repeatedly—especially in quasi-optimal test norm!—in both reference and physical space.

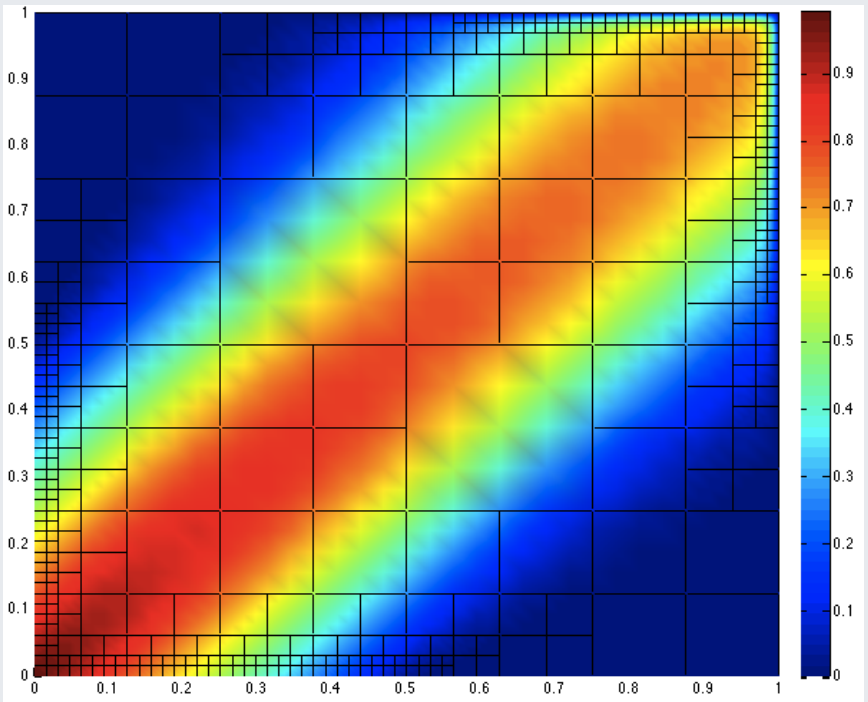
- Procedural idea: compute these values once, store them, and reuse them.
- Object-oriented analog: a cache of basis values (lazily computed, with a limited lifespan).
- Core computational components—BilinearForm, RHS, and InnerProduct—take a BasisCache as an argument.

Once implemented, the quasi-optimal test norm was nearly as fast as the mathematician's norm...

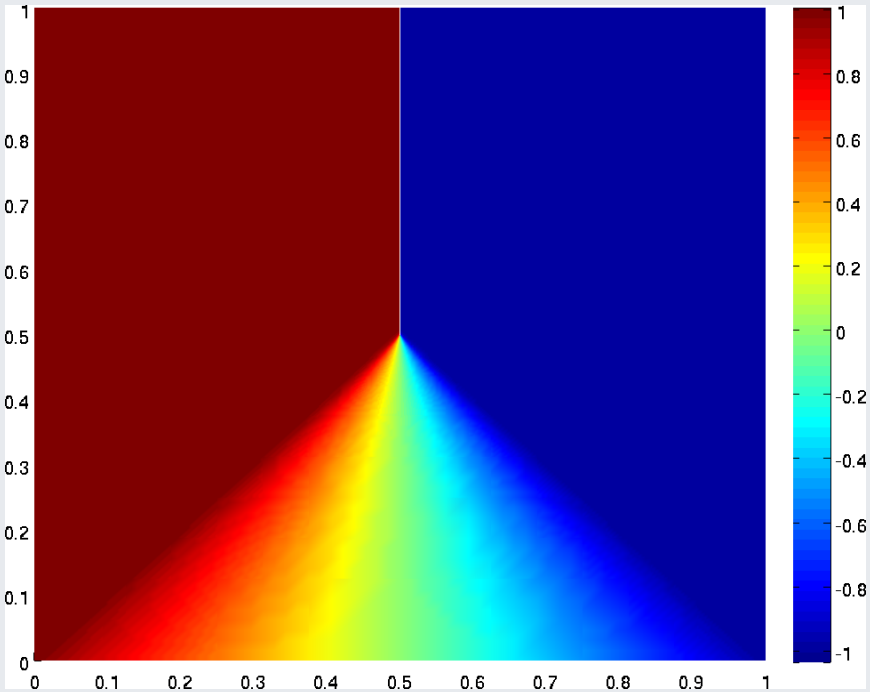
Current and Future Work

- Nonlinear PDE support
- Better scalability (distributed mesh)
- Application to Navier-Stokes

Convection-Diffusion, $\epsilon = 10^{-2}$



Burgers', $\epsilon = 10^{-4}$



Thank you!

Questions?