

Space-Time Discontinuous Petrov-Galerkin Finite Elements for Transient Computational Fluid Dynamics

Truman Ellis

Advisors: Leszek Demkowicz, Robert Moser



INSTITUTE FOR COMPUTATIONAL
ENGINEERING & SCIENCES

Table of Contents

1 Motivation

2 DPG Overview

3 Space-Time DPG

4 Camellia: DPG for the Masses

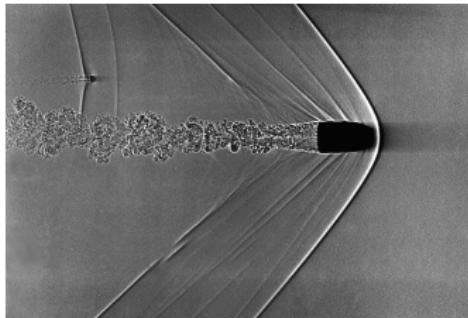
5 Other DPG Research

Navier-Stokes Equations

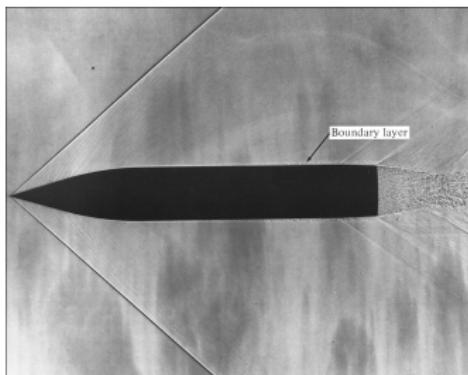
Numerical Challenges

Robust simulation of unsteady fluid dynamics remains a challenging issue.

- Resolving solution features (sharp, localized viscous-scale phenomena)
 - Shocks
 - Boundary layers - resolution needed for drag/load
 - Turbulence (non-localized)
- Stability of numerical schemes
 - Coarse/adaptive grids
 - Higher order



Shock

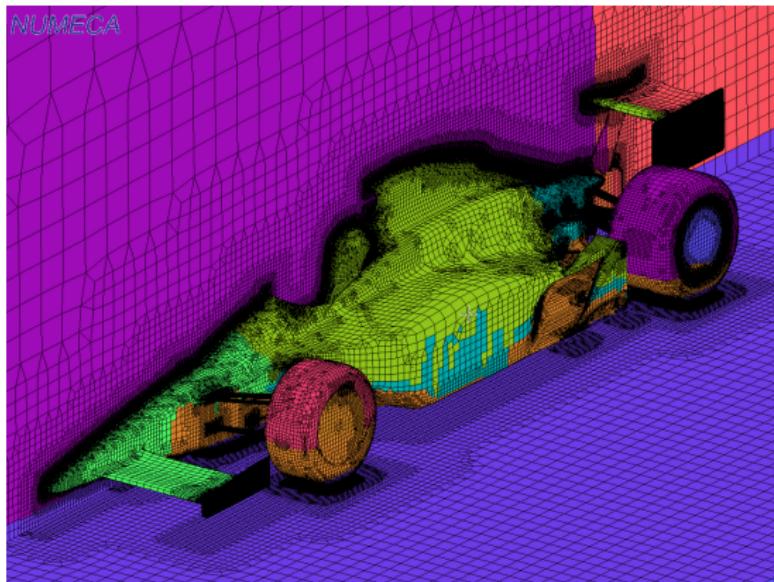


Boundary layer

Motivation

Initial Mesh Design is Expensive and Time-Consuming

- Surface mesh must accurately represent geometry
- Volume mesh needs sufficient resolution for asymptotic regime
- Engineers often forced to work by trial and error
- Bad in the context of HPC

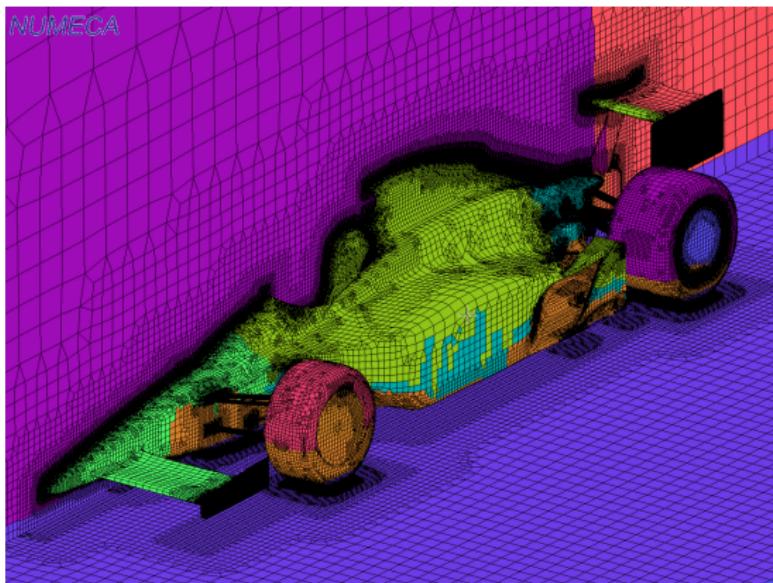


Formula 1 Mesh by Numeca

Motivation

Initial Mesh Design is Expensive and Time-Consuming

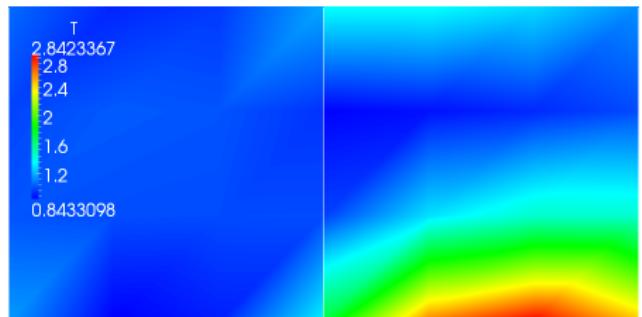
- Surface mesh must accurately represent geometry
- Volume mesh needs sufficient resolution for asymptotic regime
- Engineers often forced to work by trial and error
- Bad in the context of HPC
- **We desire an automated computational technology**



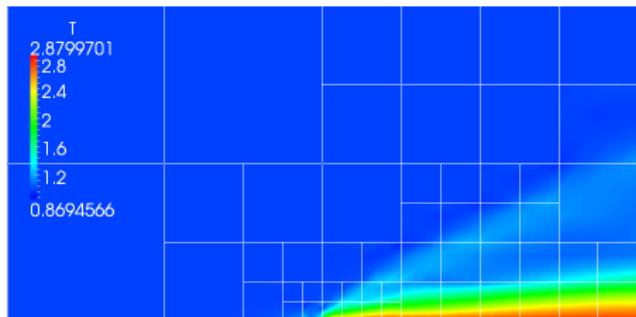
Formula 1 Mesh by Numeca

DPG on Coarse Meshes

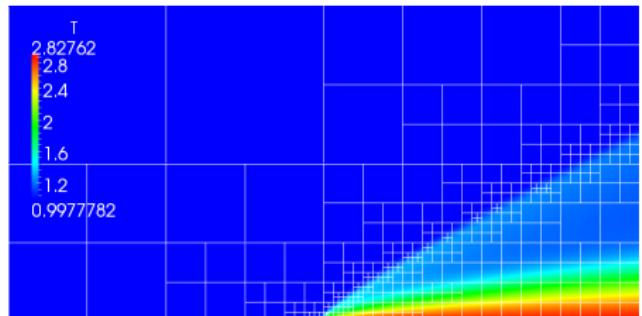
Adaptive Solve of the Carter Plate Problem¹ $Re = 1000$



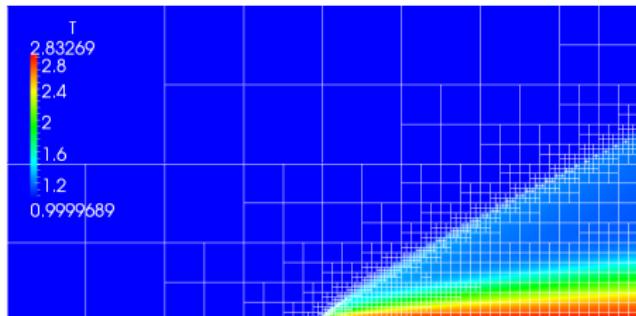
Temperature on Initial Mesh



Temperature after 4 Refinements



Temperature after 8 Refinements



Temperature after 11 Refinements

¹J.L. Chan. "A DPG Method for Convection-Diffusion Problems". PhD thesis. University of Texas at Austin, 2013.

Lessons from Other Methods

Streamline Upwind Petrov-Galerkin: Adaptively changing the test space can produce a method with better stability.

Discontinuous Galerkin: Discontinuous basis functions are a legitimate option for finite element methods.

Hybridized DG: Mesh interface unknowns can facilitate static condensation -- reducing the number of DOFs in the global solve.

Least-Squares FEM: The finite element method is most powerful in a minimum residual context (i.e. as a Ritz method).

Space-Time FEM: Highly adaptive methods should have adaptive time integration. Superior framework for problems with moving boundaries. Requires a method that is both temporally and spatially stable.

Mixed FEM: A first order formulation gives more flexibility.

Table of Contents

1 Motivation

2 DPG Overview

3 Space-Time DPG

4 Camellia: DPG for the Masses

5 Other DPG Research

Overview of DPG

DPG is a Minimum Residual Method

Find $u \in U$ such that

$$b(u, v) = l(v) \quad \forall v \in V$$

with operator $B : U \rightarrow V'$ defined by $b(u, v) = \langle Bu, v \rangle_{V' \times V}$.

This gives the operator equation

$$Bu = l \quad \in V'.$$

We wish to minimize the residual $Bu - l \in V'$:

$$u_h = \arg \min_{w_h \in U_h} \frac{1}{2} \|Bu - l\|_{V'}^2 .$$

Dual norms are not computationally tractable. Inverse Riesz map moves the residual to a more accessible space:

$$u_h = \arg \min_{w_h \in U_h} \frac{1}{2} \|R_V^{-1}(Bu - l)\|_V^2 .$$

Overview of DPG

DPG is a Minimum Residual Method

Taking the Gâteaux derivative to be zero in all directions $\delta u \in U_h$ gives,

$$(R_V^{-1}(Bu_h - l), R_V^{-1}B\delta u)_V = 0, \quad \forall \delta u \in U,$$

which by definition of the Riesz map is equivalent to

$$\langle Bu_h - l, R_V^{-1}B\delta u_h \rangle = 0 \quad \forall \delta u_h \in U_h,$$

with optimal test functions $v_{\delta u_h} := R_V^{-1}B\delta u_h$ for each trial function δu_h .

Resulting Petrov-Galerkin System

This gives a simple bilinear form

$$b(u_h, v_{\delta u_h}) = l(v_{\delta u_h}),$$

with $v_{\delta u_h} \in V$ that solves the auxiliary problem

$$(v_{\delta u_h}, \delta v)_V = \langle R_V v_{\delta u_h}, \delta v \rangle = \langle B\delta u_h, \delta v \rangle = b(\delta u_h, \delta v) \quad \forall \delta v \in V.$$

Overview of DPG

DPG is the Most Stable Petrov-Galerkin Method

Babuška's theorem guarantees that *discrete stability and approximability imply convergence*. If bilinear form $b(u, v)$, with $M := \|b\|$ satisfies the discrete inf-sup condition with constant γ_h ,

$$\sup_{v_h \in V_h} \frac{|b(u, v)|}{\|v_h\|_V} \geq \gamma_h \|u_h\|_U ,$$

then the Galerkin error satisfies the bound

$$\|u_h - u\|_U \leq \frac{M}{\gamma_h} \inf_{w_h \in U_h} \|w_h - u\|_U .$$

Optimal test function realize the supremum guaranteeing that $\gamma_h \geq \gamma$.

Energy Norm

If we use the energy norm, $\|u\|_E := \|Bu\|_{V'}$ in the error estimate, then $M = \gamma = 1$. Babuška's theorem implies that the minimum residual method is the most stable Petrov-Galerkin method (assuming exact optimal test functions).

Overview of DPG

Optimal Test Functions

Plots of Poisson, Convection, Convection-Diffusion, etc

Overview of DPG

Other Features

Discontinuous Petrov-Galerkin

- Continuous test space produces global solve for optimal test functions
- Discontinuous test space results in an embarrassingly parallel solve

Hermitian Positive Definite Stiffness Matrix

Property of all minimum residual methods

$$b(u_h, v_{\delta u_h}) = (v_{u_h}, v_{\delta u_h})_V = \overline{(v_{\delta u_h}, v_{u_h})_V} = \overline{b(\delta u_h, v_{u_h})}$$

Error Representation Function

Energy norm of Galerkin error (residual) can be computed without exact solution

$$\|u_h - u\|_E = \|B(u_h - u)\|_{V'} = \|Bu_h - l\|_{V'} = \|R_V^{-1}(Bu_h - l)\|_V$$

Overview of DPG

High Performance Computing

Eliminates human intervention

- Stability
- Robustness
- Adaptivity
- Automaticity
- Compute intensive
- Embarrassingly parallel local solves
- Factorization recyclable
- Low communication
- SPD stiffness matrix
- Multiphysics



Stampede Supercomputer at TACC



Mira Supercomputer at Argonne

Table of Contents

- 1 Motivation
- 2 DPG Overview
- 3 Space-Time DPG
- 4 Camellia: DPG for the Masses
- 5 Other DPG Research

Space-Time DPG

Extending DPG to Transient Problems

- Time stepping techniques are not ideally suited to highly adaptive grids

Space-Time DPG for Convection-Diffusion

Space-Time Divergence Form

Equation is parabolic in space-time.

$$\frac{\partial u}{\partial t} + \beta \cdot \nabla u - \epsilon \Delta u = f$$

This is just a composition of a constitutive law and conservation of mass.

$$\sigma - \epsilon \nabla u = 0$$

$$\frac{\partial u}{\partial t} + \nabla \cdot (\beta u - \sigma) = f$$

We can rewrite this in terms of a space-time divergence.

$$\begin{aligned} \frac{1}{\epsilon} \sigma - \nabla u &= 0 \\ \nabla_{xt} \cdot \begin{pmatrix} \beta u - \sigma \\ u \end{pmatrix} &= f \end{aligned}$$

Space-Time DPG for Convection-Diffusion

Ultra-Weak Formulation with Discontinuous Test Functions

Multiply by test function and integrate by parts over space-time element K.

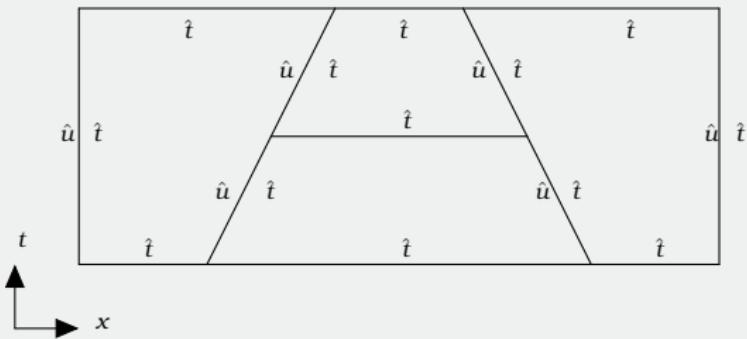
$$\begin{aligned} \left(\frac{1}{\epsilon} \boldsymbol{\sigma}, \boldsymbol{\tau} \right)_K + (u, \nabla \cdot \boldsymbol{\tau})_K - \langle \hat{u}, \boldsymbol{\tau} \cdot \mathbf{n}_x \rangle_{\partial K} &= 0 \\ - \left(\left(\begin{array}{c} \beta u - \sigma \\ u \end{array} \right), \nabla_{xt} v \right)_K + \langle \hat{t}, v \rangle_{\partial K} &= f \end{aligned}$$

where

$$\hat{u} := \text{tr}(u)$$

$$\hat{t} := \text{tr}(-\boldsymbol{\sigma}) \cdot \mathbf{n}_x + \text{tr}(u) \cdot n_t$$

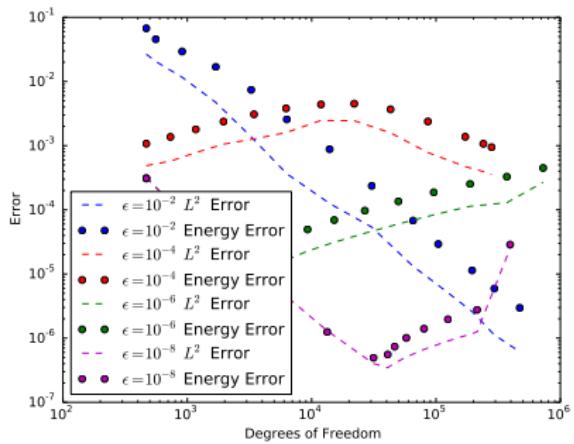
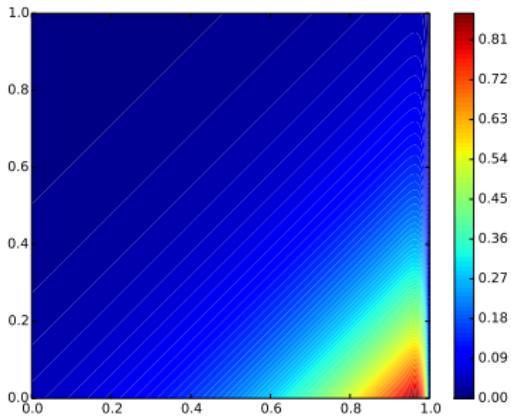
Support of Trace Variables



- Trace \hat{u} defined on spatial boundaries
- Flux \hat{t} defined on all boundaries

Space-Time Convection-Diffusion

Robust Norms



$$u = e^{-lt} (e^{\lambda_1(x-1)} - e^{\lambda_2(x-1)})$$

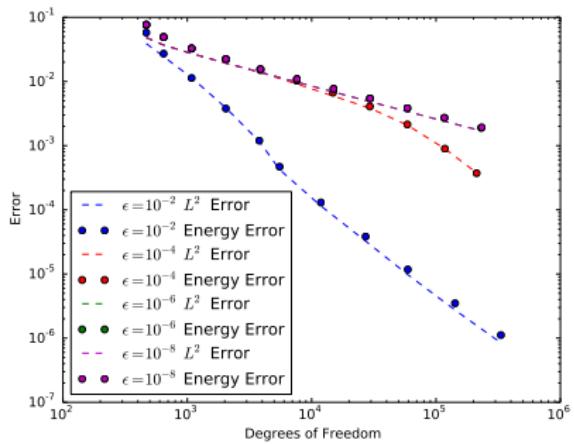
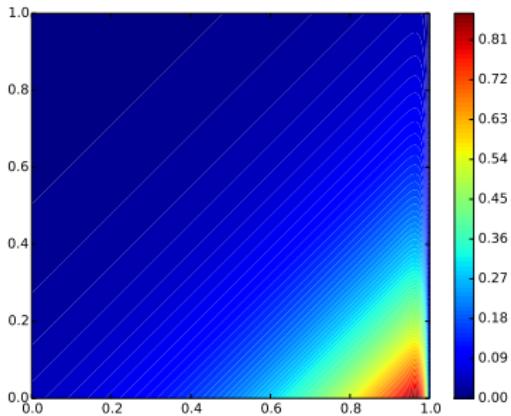
$$\lambda_{1,2} = \frac{-1 \pm \sqrt{1 - 4l\epsilon}}{-2\epsilon}$$

where $l = 3$, $\epsilon = 10^{-2}$

$$\begin{aligned} \| (v, \tau) \|^2 &= \left\| \nabla \cdot \tau - \tilde{\beta} \cdot \nabla_{xt} v \right\|^2 \\ &+ \left\| \frac{1}{\epsilon} \tau + \nabla v \right\|^2 + \| v \|^2 + \| \tau \|^2 \end{aligned}$$

Space-Time Convection-Diffusion

Robust Norms



$$u = e^{-lt} (e^{\lambda_1(x-1)} - e^{\lambda_2(x-1)})$$

$$\lambda_{1,2} = \frac{-1 \pm \sqrt{1 - 4l\epsilon}}{-2\epsilon}$$

where $l = 3$, $\epsilon = 10^{-2}$

$$\begin{aligned} \| (v, \tau) \|^2 &= \left\| \nabla \cdot \tau - \tilde{\beta} \cdot \nabla_{xt} v \right\|^2 \\ &\quad + \min \left(\frac{1}{h^2}, \frac{1}{\epsilon} \right) \| \tau \|^2 \\ &\quad + \epsilon \| \nabla v \|^2 + \| \beta \cdot \nabla v \|^2 + \| v \|^2 \end{aligned}$$

Space-Time Navier-Stokes

First Order System

$$\frac{1}{\mu} \mathbb{D} - (\nabla \mathbf{u} + (\nabla \mathbf{u})^T) + \frac{2}{3} \nabla \cdot \mathbf{u} \mathbb{I} = 0$$

$$\frac{Pr}{C_p \mu} \mathbf{q} + \nabla T = 0$$

$$\nabla_{xt} \cdot \begin{pmatrix} \rho \mathbf{u} \\ \rho \end{pmatrix} = f_c$$

$$\nabla_{xt} \cdot \begin{pmatrix} \rho \mathbf{u} \otimes \mathbf{u} + \rho R T \mathbb{I} - \mathbb{D} \\ \rho \mathbf{u} \end{pmatrix} = \mathbf{f}_m$$

$$\nabla_{xt} \cdot \begin{pmatrix} \rho \mathbf{u} (C_v T + \frac{1}{2} \mathbf{u} \cdot \mathbf{u}) + \rho R T \mathbf{u} + \mathbf{q} - \mathbf{u} \cdot \mathbb{D} \\ \rho (C_v T + \frac{1}{2} \mathbf{u} \cdot \mathbf{u}) \end{pmatrix} = f_e,$$

Space-Time Navier-Stokes

Conserved quantities

$$C_c := \rho$$

$$\mathbf{C}_m := \rho \mathbf{u}$$

$$C_e := \rho(C_v T + \frac{1}{2} \mathbf{u} \cdot \mathbf{u})$$

Euler fluxes

$$\mathbf{F}_c := \rho \mathbf{u}$$

$$\mathbb{F}_m := \rho \mathbf{u} \otimes \mathbf{u} + \rho R T \mathbb{I}$$

$$\mathbf{F}_e := \rho \mathbf{u} \left(C_v T + \frac{1}{2} \mathbf{u} \cdot \mathbf{u} \right) + \rho R T \mathbf{u}$$

Viscous fluxes

$$\mathbf{K}_c := \mathbf{0}$$

$$\mathbb{K}_m := \mathbb{D}$$

$$\mathbf{K}_e := -\mathbf{q} + \mathbf{u} \cdot \mathbb{D}$$

Viscous variables

$$\mathbb{M}_{\mathbb{D}} := \frac{1}{\mu} \mathbb{D}$$

$$\mathbf{M}_q := \frac{Pr}{C_p \mu} \mathbf{q}$$

Viscous relations

$$\mathbf{G}_{\mathbb{D}} := 2 \mathbf{u}$$

$$G_q := -T$$

Space-Time Navier-Stokes

Define Group Variables

Group terms

$$C := \{C_c, \mathbf{C}_m, C_e\}$$

$$F := \{\mathbf{F}_c, \mathbb{F}_m, \mathbf{F}_e\}$$

$$K := \{\mathbf{K}_c, \mathbb{K}_m, \mathbf{K}_e\}$$

$$M := \{\mathbb{M}_{\mathbb{D}}, \mathbf{M}_{\mathbf{q}}\}$$

$$G := \{\mathbf{G}_{\mathbb{D}}, G_{\mathbf{q}}\}$$

$$f := \{f_c, \mathbf{f}_m, f_e\}$$

Group variables

$$W := \{\rho, \mathbf{u}, T\}$$

$$\hat{W} := \{2\hat{\mathbf{u}}, -\hat{T}\}$$

$$\Sigma := \{\mathbb{D}, \mathbf{q}\}$$

$$\hat{t} := \{\hat{t}_e, \hat{\mathbf{t}}_m, , \hat{t}_e\}$$

$$\Psi := \{\mathbb{S}, \tau\}$$

$$V := \{v_c, \mathbf{v}_m, , v_e\} .$$

Navier-Stokes variational formulation is

$$(M, \Psi) + (G, \nabla \cdot \Psi) - \langle \hat{W}, \Psi \cdot \mathbf{n}_x \rangle = 0$$

$$- \left(\left(\begin{array}{c} F - K \\ C \end{array} \right), \nabla_{xt} V \right) + \langle \hat{t}, V \rangle = (f, V) .$$

Table of Contents

- 1 Motivation
- 2 DPG Overview
- 3 Space-Time DPG
- 4 Camellia: DPG for the Masses
- 5 Other DPG Research

Camellia: DPG for the Masses

Overview

Design Goal

Make DPG research and experimentation as simple as possible, while maintaining computational efficiency and scalability.

Mature support for:

- Rapid specification of DPG variational forms, inner products, etc.
- Distributed computation of stiffness matrix
- 1D - 3D geometries
- Curvilinear elements
- h - and p -refinements (anisotropic in h)
- Arbitrarily irregular meshes
- Modular refinement strategy interface

Experimental support for:

- Space-time computations
- Iterative solvers (tested up to 32,768 cores)

Convection-Diffusion in Three Slides

Building the Bilinear Form

```

VarFactory vf;
//fields:
VarPtr u = vf.fieldVar("u", L2);
VarPtr sigma = vf.fieldVar("sigma", VECTOR_L2);

// traces:
VarPtr u_hat = vf.traceVar("u_hat");
VarPtr t_n = vf.fluxVar("t_n");

// test:
VarPtr v = vf.testVar("v", HGRAD);
VarPtr tau = vf.testVar("tau", HDIV);

double eps = .01;
FunctionPtr beta_x = Function::constant(1);
FunctionPtr beta_y = Function::constant(2);
FunctionPtr beta = Function::vectorize(beta_x, beta_y);

BFPtr bf = Teuchos::rcp( new BF(vf) );

bf->addTerm((1/eps) * sigma, tau);
bf->addTerm(u, tau->div());
bf->addTerm(-u_hat, tau->dot_normal());

bf->addTerm(sigma - beta * u, v->grad());
bf->addTerm(t_n, v);

RHSPtr rhs = RHS::rhs();

```

Find $u \in L^2(\Omega_h)$, $\sigma \in \mathbf{L}^2(\Omega_h)$,
 $\hat{u} \in H^{\frac{1}{2}}(\Gamma_h)$, $\hat{t}_n \in H^{-\frac{1}{2}}(\Gamma_h)$
such that

$$\begin{aligned} \frac{1}{\epsilon} (\sigma, \tau) + (u, \nabla \cdot \tau) - \langle \hat{u}, \tau \cdot \mathbf{n} \rangle \\ - (\beta u - \sigma, \nabla v) + \langle \hat{t}_n, v \rangle = (f, v) \end{aligned}$$

for all $v \in H^1(K)$, $\tau \in \mathbf{H}(\text{div}, K)$.

where $\epsilon = 10^{-2}$, $\beta = (1, 2)^T$ and

$$f = 0.$$

Convection-Diffusion in Three Slides

Boundary Conditions and Mesh

```

int k = 2;
int delta_k = 2;
MeshPtr mesh = MeshFactory::quadMesh(bf, k+1, delta_k);
BCPtr bc = BC::bc();

SpatialFilterPtr y_equals_one = SpatialFilter::matchingY(1.0);
SpatialFilterPtr y_equals_zero = SpatialFilter::matchingY(0);
SpatialFilterPtr x_equals_one = SpatialFilter::matchingX(1.0);
SpatialFilterPtr x_equals_zero = SpatialFilter::matchingX(0.0);

FunctionPtr zero = Function::zero();
FunctionPtr x = Function::xn(1);
FunctionPtr y = Function::yn(1);
bc->addDirichlet(t_n, y_equals_zero, -2 * (1-x));
bc->addDirichlet(t_n, x_equals_zero, -1 * (1-y));
bc->addDirichlet(u_hat, y_equals_one, zero);
bc->addDirichlet(u_hat, x_equals_one, zero);

```

Create a square mesh $[0, 1] \times [0, 1]$ with boundary conditions

- $\hat{t}_n = 2x - 2$ on $y = 0$
- $\hat{t}_n = x - 1$ on $x = 0$
- $\hat{u} = 0$ on $y = 1$
- $\hat{u} = 0$ on $x = 1$

Note

- Can subclass `SpatialFilter` to match any geometry
- Adding new mesh readers is straightforward

Convection-Diffusion in Three Slides

Test Norm, Solving, and Adaptivity

```
IPPtr ip = bf->graphNorm();

SolutionPtr soln = Solution::solution(mesh, bc, rhs, ip);

double threshold = 0.20;
RefinementStrategy refStrategy(soln, threshold);

int numRefs = 10;

ostringstream refName;
refName << "ConvectionDiffusion";
HDF5Exporter exporter(mesh, refName.str());

for (int refIndex=0; refIndex < numRefs; refIndex++) {
    soln->solve();

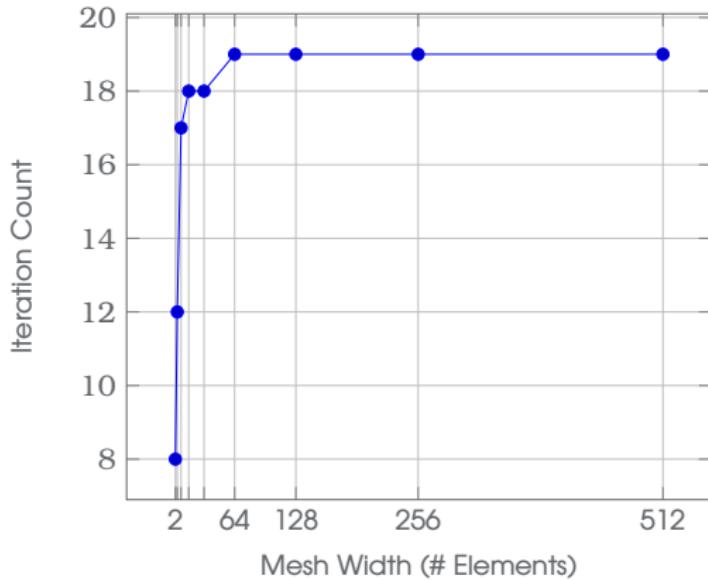
    double energyError = soln->energyErrorTotal();
    cout << "After " << refIndex << "_refinements, _energy_error_is_" << energyError << endl;

    exporter.exportSolution(soln, vf, refIndex);

    if (refIndex != numRefs)
        refStrategy.refine();
}
```

Towards a Robust Iterative Solver

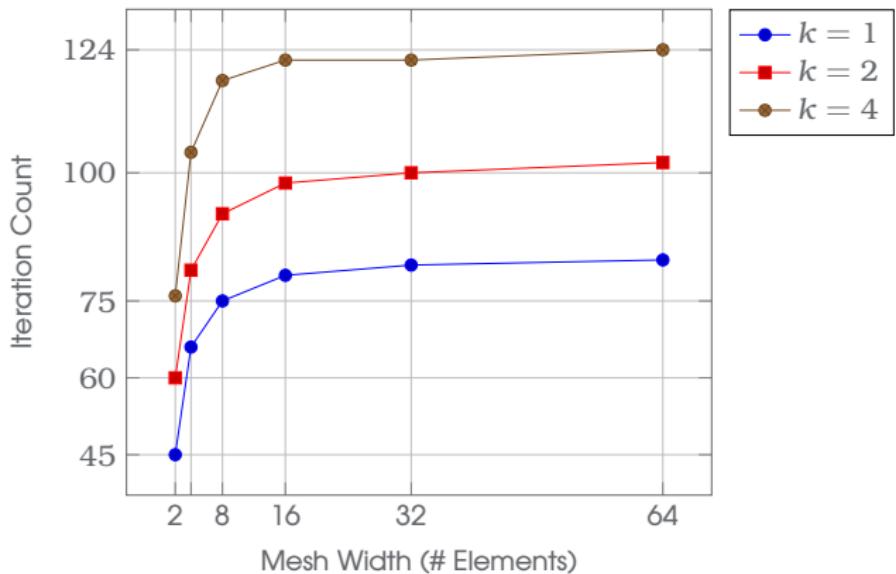
Poisson 1D, p -multigrid Preconditioners, $k = 16$



Poisson 1D: number of CG iterations to reduce error by a factor of 10^{10} using p -multigrid preconditioners with Schwarz smoother with 0 overlap for $k = 16$. The results for $k = 1, 2, 4$, and 8 are essentially identical.

Towards a Robust Iterative Solver

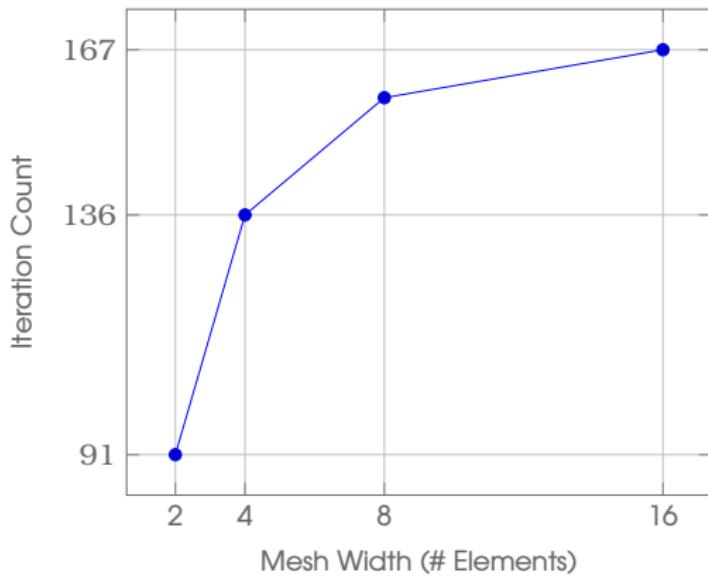
Stokes 2D, p -multigrid Preconditioners, Schwarz algebraic overlap 0



Stokes 2D: number of CG iterations to reduce error by a factor of 10^{10} using p -multigrid, algebraic Schwarz smoother with 0 overlap.

Towards a Robust Iterative Solver

Stokes 3D, p -multigrid Preconditioners, $k = 2$, Schwarz with Incomplete Cholesky



Stokes 3D: number of CG iterations to reduce error by a factor of 10^{10} using statically condensed system matrix with p -multigrid, algebraic Schwarz smoother, using Incomplete Cholesky factorization (fill ratio 5) for the Schwarz blocks.

Table of Contents

- 1 Motivation
- 2 DPG Overview
- 3 Space-Time DPG
- 4 Camellia: DPG for the Masses
- 5 Other DPG Research

Other DPG Research

- Entropy scaling for physically meaningful test norms
- DPG for non-Hilbert L_p spaces

Thank You!

