Camellia: A Software Framework for a Discontinuous Petrov-Galerkin Methodology

Nathan V. Roberts

Argonne Leadership Computing Facility

Trilinos User Group Meeting
October 28, 2014

U.S. DEPARTMENT OF ENERGY
Office of Science

Argonne NATIONAL LABORATORY

Collaborators

My collaborators in this work:

- Jesse Chan (Rice)
- Leszek Demkowicz (UT)
- Truman Ellis (UT)
- Paul Fischer (Argonne, UIUC)

Outline

DPG in Brief

DPG approach:

- *Petrov*-Galerkin: test and trial spaces differ
- discontinuous test and trial spaces
- optimal test functions computed on the fly so that

$$(v_{e_i}^{\text{opt}}, v)_V = b(e_i, v)\, \forall v \in V$$

- key choice: which norm to use on the test space?

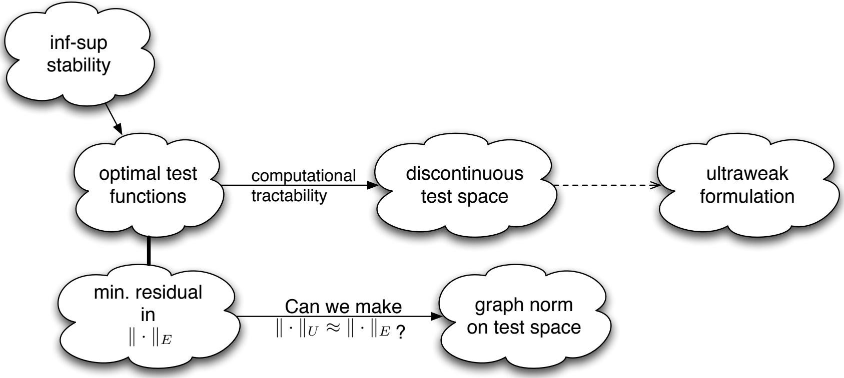
DPG features:

- automatic stability
- SPD stiffness matrix
- Error in u_h is minimized in the energy norm

$$||u_h||_E = \sup_{v \in V} \frac{b(u_h, v)}{||v||_V} = ||b(u_h, \cdot)||_{V'}$$

- Can measure the error in the energy norm to drive adaptivity.

DPG in Brief: Concept Map



* Note: we approximate the infinite-dimensional test space by taking the polynomial order k for the trial and "enriching" it somewhat: $k_{\text{test}} = k_{\text{trial}} + \Delta k$—in all that follows, $\Delta k = 1$ or $\Delta k = 2$.

Building the ultraweak formulation

PDE

$$\triangle \phi = f$$

First-Order System

$$\nabla \cdot \psi = f$$
$$\psi - \nabla \phi = 0$$

Integration by Parts

$$(\psi \cdot n, v)_{\Gamma_h} - (\psi, \nabla v)_{\Omega_h} = (f, v)_{\Omega_h}$$
$$(\psi, q)_{\Omega_h} + (\phi, q \cdot n)_{\Gamma_h} - (\phi, \nabla \cdot q)_{\Omega_h} = 0$$

Ultraweak (DPG) Variational Formulation

$$(\widehat{\psi}_n, v)_{\Gamma_h} - (\psi, \nabla v)_{\Omega_h}$$
$$+(\psi, q)_{\Omega_h} + (\widehat{\phi}, q_n)_{\Gamma_h} - (\phi, \nabla \cdot q)_{\Omega_h} = (f, v)_{\Omega_h}$$

$$b((\phi, \psi, \widehat{\phi}, \widehat{\psi}_n), (v, q)) = (f, v)_{\Omega_h}$$
$$\text{“}b(u, v) = l(v)\text{”}$$

The DPG Solve

Computational steps for solving with DPG:

1. On each element, construct the Gram matrix $G_{jk} \overset{\text{def}}{=} (v_j, v_k)_V$.
2. On each element, solve: $G_{jk} T_{ki} = B_{ji} \overset{\text{def}}{=} b(e_i, v_j)$ for the optimal test coefficients T_{ki}.
3. Since the stiffness matrix is given by

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

   we can compute

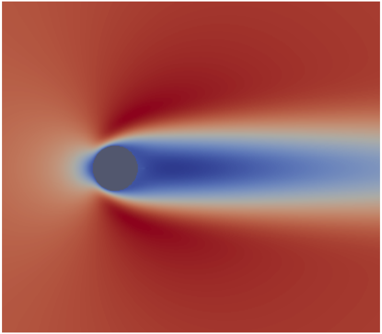
$$(v_{e_i}, v_{e_j})_V = (G^{-1}B)^T G G^{-1} B = B^T G^{-1} B = B^T_{ki} T_{kj}.$$

That is, once we've determined the optimal test functions, just need a matrix-matrix multiply to determine the local stiffness matrix!

DPG Applications to Date

DPG is a general framework, and has been successfully applied to a host of PDE problems, including:

- convection-dominated diffusion
- acoustics/wave propagation
- linear elasticity
- Maxwell's equations (cloaking problem)
- Burgers' equations
- Euler equations
- compressible Navier-Stokes
- Stokes
- incompressible Navier-Stokes



flow past a cylinder, $\mathrm{Re} = 40$

[1] Bold items have Camellia-based implementations.

Classical Stokes Problem

The classical strong form of the Stokes problem in $\Omega \subset \mathbb{R}^2$ is given by

$$\begin{aligned} -\mu\Delta\boldsymbol{u} + \nabla p &= \boldsymbol{f} && \text{in } \Omega, \\ \nabla\cdot\boldsymbol{u} &= 0 && \text{in } \Omega, \\ \boldsymbol{u} &= \boldsymbol{u}_D && \text{on } \partial\Omega, \end{aligned}$$

where μ is (constant) viscosity, p pressure, $\boldsymbol{u}$ velocity, and $\boldsymbol{f}$ a vector forcing function.

DPG Applied to Stokes

To apply DPG, we need a first-order system. We introduce $\boldsymbol{\sigma} = \mu \nabla \boldsymbol{u}$:

$$\begin{aligned}
-\nabla \cdot \boldsymbol{\sigma} + \nabla p &= \boldsymbol{f} && \text{in } \Omega, \\
\nabla \cdot \boldsymbol{u} &= 0 && \text{in } \Omega, \\
\boldsymbol{\sigma} - \mu \nabla \boldsymbol{u} &= 0 && \text{in } \Omega.
\end{aligned}$$

Testing with $(\boldsymbol{v}, q, \boldsymbol{\tau})$, and integrating by parts, we have

$$\begin{aligned}
(\boldsymbol{\sigma} - p\boldsymbol{I}, \nabla \boldsymbol{v})_{\Omega_h} - \left\langle \widehat{\boldsymbol{t}}_n, \boldsymbol{v} \right\rangle_{\Gamma_h} &= (\boldsymbol{f}, \boldsymbol{v})_{\Omega_h} \\
(\boldsymbol{u}, \nabla q)_{\Omega_h} - \langle \widehat{\boldsymbol{u}} \cdot \boldsymbol{n}, q \rangle_{\Gamma_h} &= 0 \\
(\boldsymbol{\sigma}, \boldsymbol{\tau})_{\Omega_h} + (\mu \boldsymbol{u}, \nabla \cdot \boldsymbol{\tau})_{\Omega_h} - \langle \widehat{\boldsymbol{u}}, \boldsymbol{\tau} \boldsymbol{n} \rangle_{\Gamma_h} &= \boldsymbol{0},
\end{aligned}$$

where traction $\boldsymbol{t}_n \stackrel{\text{def}}{=} (\boldsymbol{\sigma} - p\boldsymbol{I})\boldsymbol{n}$, and the hatted variables $\widehat{\boldsymbol{t}}_n$ and $\widehat{\boldsymbol{u}}$ are new unknowns representing the traces of the corresponding variables at the boundary.

Formulation for Navier-Stokes

To derive a corresponding Navier-Stokes formulation, recall that the Navier-Stokes equations may be written

$$
\begin{aligned}
-\nabla p + \nabla \cdot \boldsymbol{\sigma} &= \boldsymbol{f} + \boldsymbol{u} \cdot \nabla \boldsymbol{u} \\
\boldsymbol{\sigma} - \mu \nabla \boldsymbol{u} &= 0 \\
\nabla \cdot \boldsymbol{u} &= 0
\end{aligned}
$$

where $\mu = \frac{1}{\mathrm{Re}}$. Since $\boldsymbol{\sigma} = \mu \nabla \boldsymbol{u}$, we can write

$$
\begin{aligned}
-\nabla p + \nabla \cdot \boldsymbol{\sigma} - \frac{1}{\mu} \boldsymbol{u} \cdot \boldsymbol{\sigma} &= \boldsymbol{f} \\
\boldsymbol{\sigma} - \mu \nabla \boldsymbol{u} &= 0 \\
\nabla \cdot \boldsymbol{u} &= 0
\end{aligned}
$$

Formulation for Navier-Stokes

$$
\begin{aligned}
-\nabla p + \nabla \cdot \boldsymbol{\sigma} - \frac{1}{\mu}\boldsymbol{u} \cdot \boldsymbol{\sigma} &= \boldsymbol{f} \\
\boldsymbol{\sigma} - \mu \nabla \boldsymbol{u} &= 0 \\
\nabla \cdot \boldsymbol{u} &= 0
\end{aligned}
$$

Linearizing about $u + du = (\boldsymbol{u} + \boldsymbol{du}, \boldsymbol{\sigma} + \boldsymbol{d\sigma}, p + dp)$, we have

$$
\begin{aligned}
b_{\text{Stokes}}(du, v) - \left(\boldsymbol{du} \cdot \boldsymbol{\sigma} + \frac{1}{\mu}\boldsymbol{u} \cdot \boldsymbol{d\sigma}, \boldsymbol{v} \right)_{\Omega_h} \\
= (\boldsymbol{f}, \boldsymbol{v})_{\Omega_h} - b_{\text{Stokes}}(u, v) + \left(\frac{1}{\mu}\boldsymbol{u} \cdot \boldsymbol{\sigma}, \boldsymbol{v} \right)_{\Omega_h}.
\end{aligned}
$$

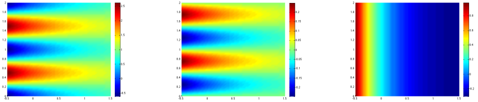
For now, we use the graph norm for Navier-Stokes (for high Reynolds numbers, we should do something else).

Kovasznay Flow

A common test case for Navier-Stokes is an analytic solution due to Kovasznay [3]:

$$u_1 = 1 - e^{\lambda x}\cos(2\pi y)$$
$$u_2 = \frac{\lambda}{2\pi}e^{\lambda x}\sin(2\pi y)$$
$$p = \frac{1}{2}e^{2\lambda x} + C$$

where $\lambda = \frac{\mathrm{Re}}{2} - \sqrt{\left(\frac{\mathrm{Re}}{2}\right)^2 + (2\pi)^2}$. We use $\Omega = (-0.5, 1.5) \times (0, 2)$ as our domain, and choose the constant C so that p has zero average on Ω.



(a) u_1 (b) u_2 (c) p

Figure : Kovasznay flow for $\mathrm{Re} = 40$: u_1, u_2 and p.

Kovasznay: $||(\boldsymbol{u}, p, \boldsymbol{\sigma})||_{L^2}$ Convergence

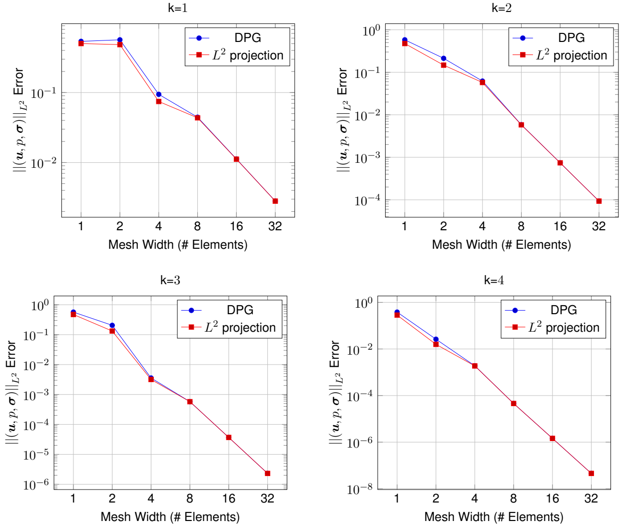
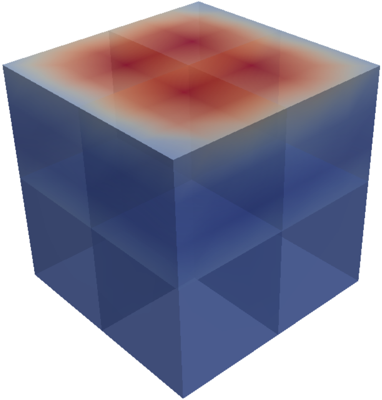


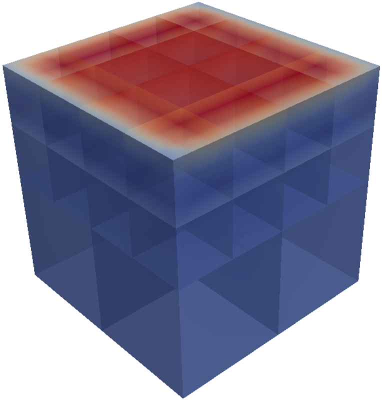
Figure : $\mathrm{Re} = 40$ flow results for DPG Navier-Stokes: L^2 error of all field variables.

Stokes Cavity Flow 3D



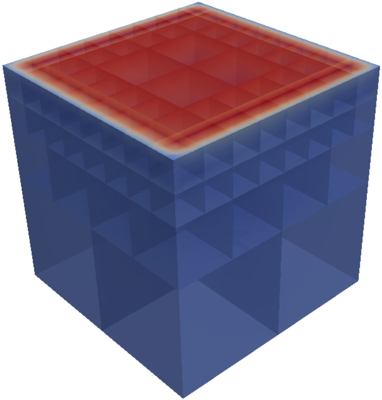
Starting with a $2 \times 2 \times 2$ quadratic mesh, we refine 6 times. Refinement 0.

Stokes Cavity Flow 3D



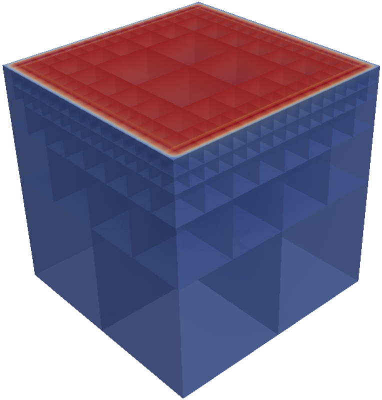
Starting with a $2 \times 2 \times 2$ quadratic mesh, we refine 6 times. Refinement 1.

Stokes Cavity Flow 3D



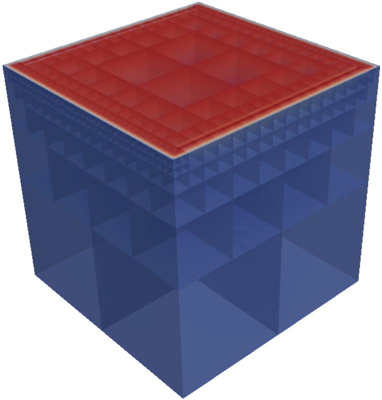
Starting with a $2 \times 2 \times 2$ quadratic mesh, we refine 6 times. Refinement 2.

Stokes Cavity Flow 3D



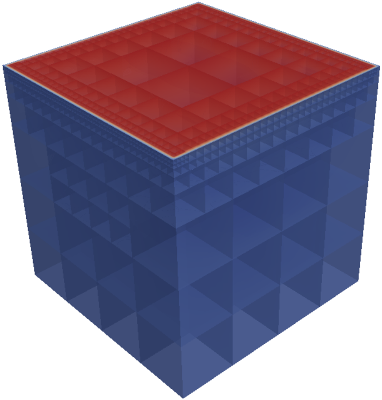
Starting with a $2 \times 2 \times 2$ quadratic mesh, we refine 6 times. Refinement 3.

Stokes Cavity Flow 3D



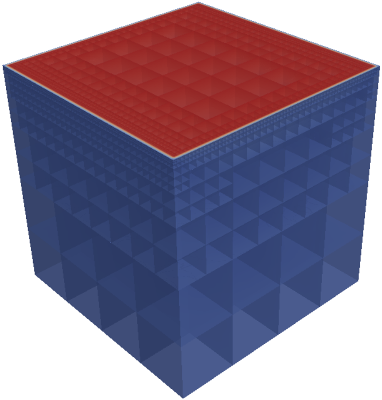
Starting with a $2 \times 2 \times 2$ quadratic mesh, we refine 6 times. Refinement 4.

Stokes Cavity Flow 3D



Starting with a $2 \times 2 \times 2$ quadratic mesh, we refine 6 times. Refinement 5.

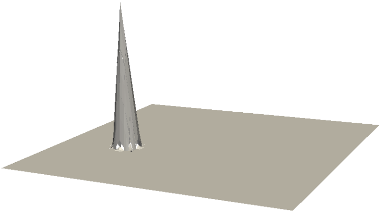
Stokes Cavity Flow 3D



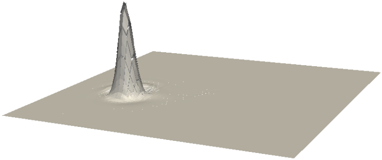
Starting with a $2 \times 2 \times 2$ quadratic mesh, we refine 6 times. Refinement 6.

Convecting Cone Problem

Beginning with 2D data in the range $[0, 1]$ in the shape of a cone as initial condition, we convect it in a circle, and examine the range of the final solution. Want to assess the spatial method, so we use Crank-Nicolson with $\Delta t = \frac{2\pi}{2000}$ where the time for one revolution is 2π.



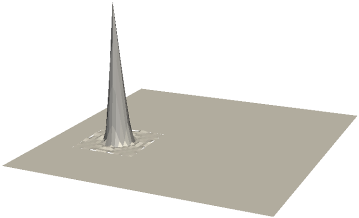
$k = 1, 64 \times 64$ mesh, initial value.



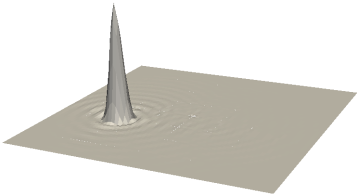
$k = 1, 64 \times 64$ mesh, final value.
Range: $[-0.050, 0.66]$.

Convecting Cone Problem

Beginning with 2D data (in the shape of a cone) as initial condition in the range $[0, 1]$, we convect it in a circle, and examine the range of the final solution.

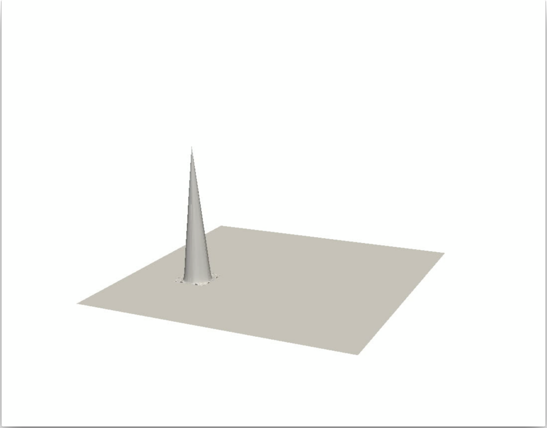


$k = 8, 8 \times 8$ mesh, initial value.



$k = 8, 8 \times 8$ mesh, final value.
Range: $[-0.016, 0.89]$.

Convecting Cone Problem

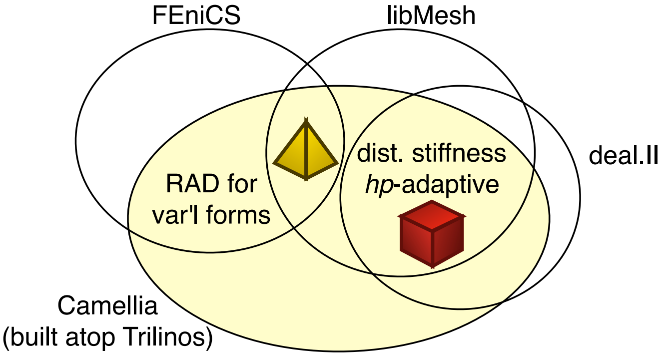


Convecting cone solution for $k = 8, 8 \times 8$ mesh: one revolution.

Outline

Camellia[1]

Design Goal: make DPG research and experimentation as simple as possible, while maintaining computational efficiency and scalability.



[1] Nathan V. Roberts. Camellia: A software framework for discontinuous Petrov-Galerkin methods. *Computers & Mathematics with Applications*, 2014.

Camellia: Rapid Specification of Inner Products

Suppose we wish to specify a test space norm

$$||(v, \boldsymbol{q})||_V^2 = ||v||^2 + ||\boldsymbol{q}||^2 + \left|\left|\frac{\partial v}{\partial x} - \frac{\partial v}{\partial y} + \nabla \cdot \boldsymbol{q}\right|\right|^2.$$

To specify this in Camellia, simply do:

```
VarFactory varFactory;
VarPtr v = varFactory.testVar("v", HGRAD);
VarPtr q = varFactory.testVar("q", HDIV);
IPPtr ip = IP::ip();
ip->addTerm(v);
ip->addTerm(q);
ip->addTerm(v->dx() - v->dy() + q->div());
```

Camellia: Rapid Specification of Inner Products

What if you wanted a test norm that varied in space? Maybe something like:

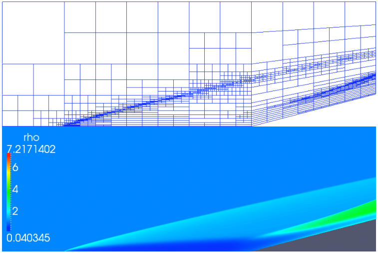
$$||(v)||_V^2 = ||v||^2 + \left|\left|\nabla v \cdot \begin{pmatrix} 1-x \\ y \end{pmatrix}\right|\right|^2.$$

We can handle that, too.

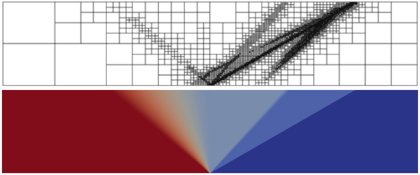
```
VarFactory varFactory;
VarPtr v = varFactory.testVar("v", HGRAD);
IPPtr ip = IP::ip();
ip->addTerm(v);
FunctionPtr x = Function::xn(1);
FunctionPtr y = Function::yn(1);
FunctionPtr weight = Function::vectorize(1-x, y);
ip->addTerm(weight * v->grad());
```

It is also simple to specify your own custom functions by subclassing `Function`.

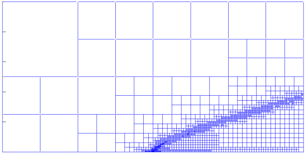
Example Camellia Applications



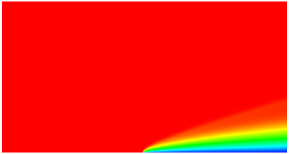
(a) Ramp Shock



(b) Sod Shock Tube (space-time)



(c) Carter Mesh



(d) Carter u_1

Camellia Features

Important features of Camellia:

- mechanisms for rapid specification of DPG variational forms, inner products, etc.
- distributed computation of stiffness matrix
- distributed representation of the solution
- 2D: curvilinear elements
- 2D: meshes of triangles and/or quads
- 3D: nonconforming hexahedra

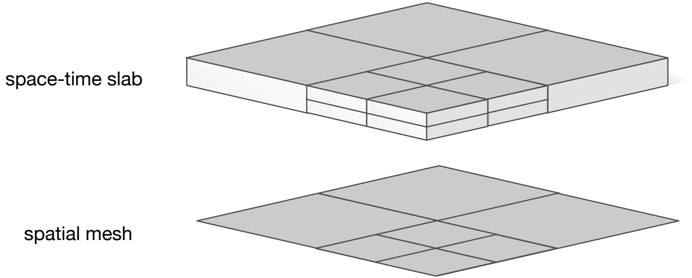
Adaptivity features:

- h- and p-refinements (anisotropic in h in 2D)
- arbitrarily irregular meshes
- modular interface: simple to implement new adaptive strategies

Camellia Features (Coming Soon)

Features under development:

- robust iterative solver for the global solve (nearly there)
- space-time elements (partway there)
- distributed mesh representation (not yet there)



Outline

Camellia and Trilinos

Camellia relies heavily on several Trilinos packages:

- Epetra—distributed matrices and vectors
- Intrepid—basis functions, pullbacks, FieldContainer
- Shards—cell topologies
- Teuchos—RCP, CLI options parsing, LAPACK interface
- Amesos—direct solver interface (MUMPS, SuperLUDist, KLU)
- AztecOO—CG and GMRES iterative solvers
- IfPack—additive Schwarz preconditioners
- Zoltan—mesh partitioning

Outline

Suitability of DPG for HPC

DPG has several attractive features for HPC:

- locality: optimal test functions embarrassingly parallel
- intensity: high-order computations take advantage of “free” flops
- automaticity: robust adaptivity means less human involvement

DPG for HPC: Software Checklist

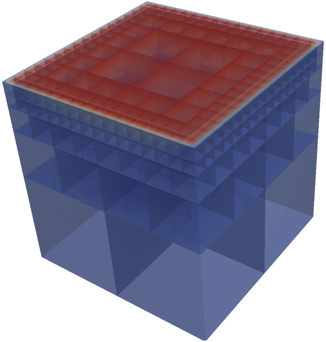
Feature	Camellia
parallel optimal test function determination distributed stiffness matrix	
high-order basis functions static condensation	
hp-adaptivity	
robust iterative solver (multigrid-preconditioned CG)	
distributed solution representation distributed mesh representation	
support for curvilinear geometry 3D support time-domain support (space-time)	

DPG for HPC: Software Checklist

Feature	Camellia
parallel optimal test function determination	✓
distributed stiffness matrix	✓
high-order basis functions	✓
static condensation	✓
hp-adaptivity	✓
robust iterative solver (multigrid-preconditioned CG)	to-do
distributed solution representation	✓
distributed mesh representation	to-do
support for curvilinear geometry	✓ (2D)
3D support	✓
time-domain support (space-time)	to-do

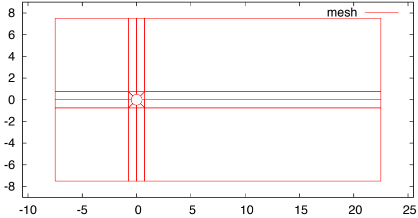
Thank you for your attention!

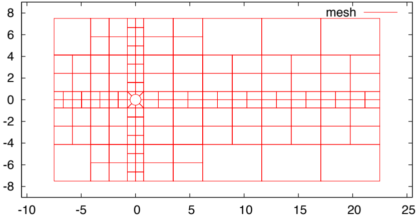
Questions?



For more info:
nvroberts@alcf.anl.gov

Flow Past a Cylinder: Initial Mesh





Starting with a minimal mesh on a domain 30×15 cylinder diameters, we refine anisotropically to make the (cubic) mesh approximately isotropic and 1-irregular. The mesh on the right has 256 elements and 23,488 degrees of freedom.

Deriving DPG

Recall Babuška's Theorem:
Suppose we have a bilinear form $b(u, v)$ defined on Hilbert spaces U and V, and suppose that

$$|b(u, v)| \leq M \, ||u||_U \, ||v||_V$$

for continuity constant M,

Deriving DPG

Recall Babuška's Theorem:
Suppose we have a bilinear form $b(u,v)$ defined on Hilbert spaces U and V, and suppose that

$$|b(u,v)| \leq M \, ||u||_U \, ||v||_V$$

for continuity constant M, and suppose that the inf-sup condition holds:

$$\inf_{||u||_U=1} \sup_{||v||_V=1} b(u,v) \geq \gamma.$$

Deriving DPG

Now, consider a discretization $U_h \subset U$, $V_h \subset V$, where $\dim U_h = \dim V_h$. If we have the discrete inf-sup condition

$$\inf_{||u||_{U_h}=1} \sup_{||v||_{V_h}=1} b(u,v) \geq \gamma_h > 0,$$

then Babuška's Theorem tells us that the continuous and discrete problems have unique solutions u and u_h, and that

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

Originating Idea

Can we choose our test space V_h so that the discrete inf-sup condition automatically holds?

Originating Idea

Can we choose our test space V_h so that the discrete inf-sup condition automatically holds?

For each basis function $e_i \in U_h$, find $v_{e_i} \in V$ that realizes the supremum:

$$\sup_{||v||_V=1} b(e_i, v) = b(e_i, v_{e_i}).$$

We can find this by solving $(v_{e_i}, v)_V = b(e_i, v)\ \forall v \in V$.

Originating Idea

Can we choose our test space V_h so that the discrete inf-sup condition automatically holds?

For each basis function $e_i \in U_h$, find $v_{e_i} \in V$ that realizes the supremum:

$$\sup_{||v||_V=1} b(e_i, v) = b(e_i, v_{e_i}).$$

We can find this by solving $(v_{e_i}, v)_V = b(e_i, v)\ \forall v \in V$. If we solve this

exactly and use the v_{e_i} as our (optimal) test space, we can show that Babuška's Theorem gives us

$$||u - u_h||_E = \inf_{w_h \in U_h} ||u - w_h||_E \,,$$

where

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

The Abstract Problem and Minimization of the Residual

Take U, V Hilbert.

We seek $u \in U$ such that

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

where b is bilinear and and l is linear in v. Writing in operator form

$$Bu = l,$$

and fixing discrete space $U_h \subset U$, we seek to minimize the residual

$$Bu_h - l.$$

Specifically, we seek

$$u_h = \underset{w_h \in U_h}{\arg\min} \; \frac{1}{2} \left\| Bw_h - l \right\|_{V'}^2 .$$

The Abstract Problem and Minimization of the Residual

$$u_h = \underset{w_h \in U_h}{\arg\min} \; \frac{1}{2} \left|\left| B w_h - l \right|\right|_{V'}^2 .$$

Now, the dual space V' is not especially easy to work with; we would prefer to work with V itself. Recalling that the Riesz operator $R_V : V \to V'$ defined by

$$R_V v = (v, \cdot)_V ,$$

is an *isometry*—$\left|\left|R_V v\right|\right|_{V'} = \left|\left|v\right|\right|_V$—we can rewrite the term we want to minimize as a norm in V:

$$\begin{aligned} \frac{1}{2} \left|\left| B w_h - l \right|\right|_{V'}^2 &= \frac{1}{2} \left|\left| R_V^{-1} \left(B w_h - l \right) \right|\right|_V^2 \\ &= \frac{1}{2} \left(R_V^{-1} \left(B w_h - l \right), R_V^{-1} \left(B w_h - l \right) \right)_V . \end{aligned}$$

The Abstract Problem and Minimization of the Residual

We seek to minimize

$$\frac{1}{2}\left(R_V^{-1}\left(Bw_h - l\right), R_V^{-1}\left(Bw_h - l\right)\right)_V.$$

The first-order optimality condition requires that the Gâteaux derivative be equal to zero for minimizer u_h; since B is linear, we have

$$\left(R_V^{-1}\left(Bu_h - l\right), R_V^{-1}B\delta u_h\right)_V = 0, \quad \forall \delta u_h \in U_h.$$

By the definition of R_V, this is equivalent to

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

The Abstract Problem and Minimization of the Residual

We have:

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

Now, if we take e_i as a basis for U_h and identify $v_{e_i} = R_V^{-1} Be_i$ as test functions, we can rewrite this as

$$b(u_h, v_{e_i}) = l(v_{e_i}).$$

Thus, the discrete solution that minimizes the residual is exactly attained by testing the original equation with appropriate test functions. We call these optimal test functions.[2]

[2] L. Demkowicz and J. Gopalakrishnan. A class of discontinuous Petrov–Galerkin methods. Part II: Optimal test functions. *Numerical Methods for Partial Differential Equations*, 27(1):70–105, 2011.

Technical Assumptions (true for VGP Stokes)

Under modest technical assumptions (true for Stokes), we have[3]

$$||Au|| \geq \gamma \, ||u|| \implies \sup_{v \in H_{A^*}} \frac{b((u,\widehat{u}),v)}{||v||_{H_{A^*}}} \geq \gamma_{\text{DPG}} \left(||u||^2 + ||\widehat{u}||^2_{\widehat{H}_A(\Gamma_h)} \right)^{1/2}$$

where $\gamma_{\text{DPG}} = O(\gamma)$ is a mesh-independent constant, and $||\cdot||_{\widehat{H}_A(\Gamma_h)}$ is the minimum energy extension norm.

In the next slides, we detail the assumptions.

[3] Nathan V. Roberts, Tan Bui-Thanh, and Leszek F. Demkowicz. The DPG method for the Stokes problem. *Computers and Mathematics with Applications*, 2014.

Technical Assumptions (true for VGP Stokes)

$$||Au|| \geq \gamma\,||u|| \implies \sup_{v \in H_{A^*}} \frac{b((u,\widehat{u}),v)}{||v||_{H_{A^*}}} \geq \gamma_{\mathrm{DPG}} \left(||u||^2 + ||\widehat{u}||^2_{\widehat{H}_A(\Gamma_h)} \right)^{1/2}$$

Define C as the operator arising from integration by parts:

$$(Au, v)_\Omega = (u, A^* v)_\Omega + \langle Cu, v \rangle.$$

We split C into C_1 and C_2 such that

$$\begin{aligned} \langle Cu, v \rangle &= \langle C_1 u, v \rangle + \langle C_2 u, v \rangle \\ &= \langle C_1 u, v \rangle + \langle u, C_2' v \rangle \end{aligned}$$

where $C_1 u = f_D$ corresponds to the Dirichlet BCs imposed.

Technical Assumptions (true for VGP Stokes)

$$||Au|| \geq \gamma\,||u|| \implies \sup_{v \in H_{A^*}} \frac{b((u,\widehat{u}),v)}{||v||_{H_{A^*}}} \geq \gamma_{\mathrm{DPG}} \left(||u||^2 + ||\widehat{u}||^2_{\widehat{H}_A(\Gamma_h)} \right)^{1/2}$$

Assumptions:

- Theorem Hypothesis: with homogeneous boundary condition $C_1 u = 0$ in place, operator A is bounded below in the L^2-orthogonal component of its null space.
- C_1 and C_2 are defined in such a way that

$$\left(\langle u, C_2' v \rangle = 0 \quad \forall u \ : C_1 u = 0 \right) \implies C_2' v = 0.$$

- A and A^* are surjective.
- Both graph spaces $H_A(\Omega)$ and $H_{A^*}(\Omega)$ admit corresponding trace spaces $\widehat{H}_A(\partial\Omega)$ and $\widehat{H}_{A^*}(\partial\Omega)$.
- The boundary term $\langle Cu, v \rangle$ arising from integration by parts is definite.

Naive Test Norm[4]

What if we don't use the graph norm, but a naive choice instead?

$$||(\boldsymbol{\tau}, \boldsymbol{v}, q)||_{\text{naive}}^2 = ||\boldsymbol{\tau}||^2 + ||\nabla \cdot \boldsymbol{\tau}||^2 + ||\boldsymbol{v}||^2 + ||\nabla \boldsymbol{v}||^2 + ||q||^2 + ||\nabla q||^2 .$$

[4] N.V. Roberts, D. Ridzal, P.N. Bochev, L. Demkowicz, K.J. Peterson, and C. M. Siefert. Application of a discontinuous Petrov-Galerkin method to the Stokes equations. In *CSRI Summer Proceedings 2010*. Sandia National Laboratories, 2010.

Naive Test Norm: u_1 convergence

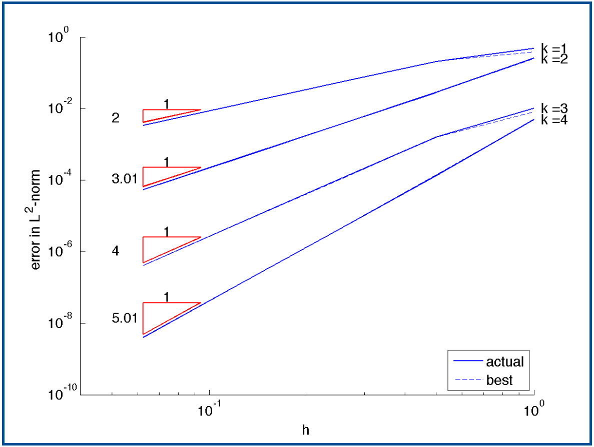


Figure : h-convergence with the naive norm: u_1. Dashed lines: best approximation error.

Naive Test Norm: u_2 convergence

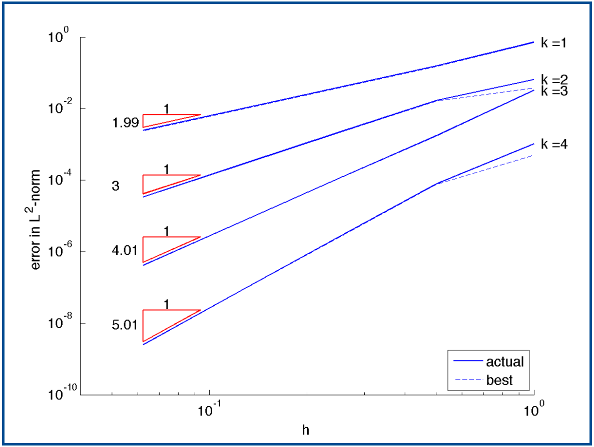


Figure : h-convergence with the naive norm: u_2. Dashed lines: best approximation error.

Naive Test Norm: p convergence

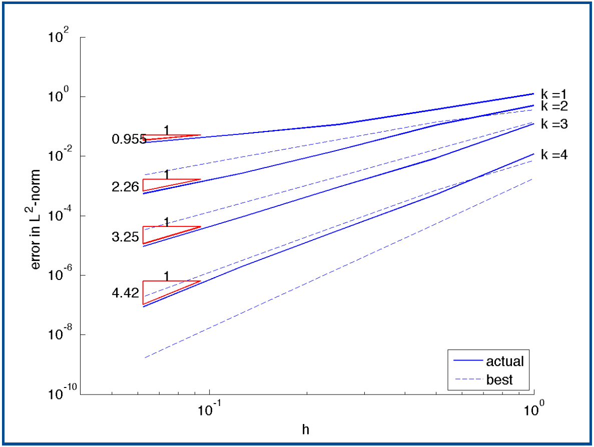


Figure : h-convergence with the naive norm: p. Dashed lines: best approximation error.

Graph vs. Naive Test Norm

What's the difference between the two norms? Why are the results better with the graph norm?

$$
\begin{aligned}
||(\boldsymbol{\tau}, \boldsymbol{v}, q)||^2_{\text{naive}} &= ||\nabla \cdot \boldsymbol{\tau}||^2 + ||\nabla \cdot \boldsymbol{v}||^2 + ||\nabla q||^2 + ||\boldsymbol{\tau}||^2 + ||\boldsymbol{v}||^2 + ||q||^2 \\
||(\boldsymbol{\tau}, \boldsymbol{v}, q)||^2_{\text{graph}} &= ||\nabla \cdot \boldsymbol{\tau} - \nabla q||^2 + ||\nabla \cdot \boldsymbol{v}||^2 + ||\boldsymbol{\tau} + \nabla \boldsymbol{v}||^2 \\
&\quad + ||\boldsymbol{\tau}||^2 + ||\boldsymbol{v}||^2 + ||q||^2
\end{aligned}
$$

Graph vs. Naive Test Norm

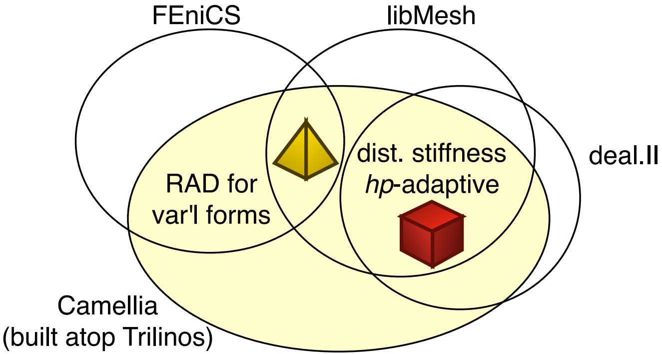
What's the difference between the two norms? Why are the results better with the graph norm?

$$\begin{aligned} ||(\boldsymbol{\tau}, \boldsymbol{v}, q)||^2_{\text{naive}} &= ||\nabla \cdot \boldsymbol{\tau}||^2 + ||\nabla \cdot \boldsymbol{v}||^2 + ||\nabla q||^2 + ||\boldsymbol{\tau}||^2 + ||\boldsymbol{v}||^2 + ||q||^2 \\ ||(\boldsymbol{\tau}, \boldsymbol{v}, q)||^2_{\text{graph}} &= ||\nabla \cdot \boldsymbol{\tau} - \nabla q||^2 + ||\nabla \cdot \boldsymbol{v}||^2 + ||\boldsymbol{\tau} + \nabla \boldsymbol{v}||^2 \\ &\quad + ||\boldsymbol{\tau}||^2 + ||\boldsymbol{v}||^2 + ||q||^2 \end{aligned}$$

The naive norm is stronger—e.g. it requires $\nabla \cdot \boldsymbol{\tau} \in L^2$ and $\nabla q \in L^2$, whereas the graph norm merely requires that $\nabla \cdot \boldsymbol{\tau} - \nabla q \in L^2$.

Camellia[5]

Design Goal: make DPG research and experimentation as simple as possible, while maintaining computational efficiency and scalability.



[5] Nathan V. Roberts. Camellia: A software framework for discontinuous Petrov-Galerkin methods. *Computers & Mathematics with Applications*, 2014.

L. Demkowicz and J. Gopalakrishnan.
A class of discontinuous Petrov–Galerkin methods. Part II: Optimal test functions.
Numerical Methods for Partial Differential Equations, 27(1):70–105, 2011.

Jiten C. Kalita and Shuvam Sen.
Triggering asymmetry for flow past circular cylinder at low Reynolds numbers.
Computers & Fluids, 59(0):44 – 60, 2012.

L. I. G. Kovasznay.
Laminar flow behind a two-dimensional grid.
Mathematical Proceedings of the Cambridge Philosophical Society, 44(01):58–62, 1948.

Nathan V. Roberts.
Camellia: A software framework for discontinuous Petrov-Galerkin methods.
Computers & Mathematics with Applications, 2014.

Nathan V. Roberts, Tan Bui-Thanh, and Leszek F. Demkowicz.
The DPG method for the Stokes problem.
Computers and Mathematics with Applications, 2014.

N.V. Roberts, D. Ridzal, P.N. Bochev, L. Demkowicz, K.J. Peterson, and C. M. Siefert.
Application of a discontinuous Petrov-Galerkin method to the Stokes equations.
In *CSRI Summer Proceedings 2010*. Sandia National Laboratories, 2010.