# Discontinuous Galerkin Method for Stokes Equation (P)

## Seminar in Computational Fluid Dynamics

Purusharth Saxena
Universität Heidelberg

January 7, 2024

# Overview

Informal introduction to discontinuous Galerkin Methods

# Brief Introduction

- Discontinuous Galerkin (DG) method uses discontinuous basis functions.

# Brief Introduction

- Discontinuous Galerkin (DG) method uses discontinuous basis functions.
- First dG method was introduced by reed and Hill (1973) for numerically solving the neutron transport equation.

# Brief Introduction

- Discontinuous Galerkin (DG) method uses discontinuous basis functions.
- First dG method was introduced by reed and Hill (1973) for numerically solving the neutron transport equation.
- These method give discontinuous approximations defined by a Galerkin method *element by element*.

# Brief Introduction

- Discontinuous Galerkin (DG) method uses discontinuous basis functions.
- First dG method was introduced by reed and Hill (1973) for numerically solving the neutron transport equation.
- These method give discontinuous approximations defined by a Galerkin method *element by element*.
- Since these methods employ discontinuous approximations, they can easily handle elements of arbitrary shapes.

# Brief Introduction

- Discontinuous Galerkin (DG) method uses discontinuous basis functions.
- First dG method was introduced by reed and Hill (1973) for numerically solving the neutron transport equation.
- These method give discontinuous approximations defined by a Galerkin method *element by element*.
- Since these methods employ discontinuous approximations, they can easily handle elements of arbitrary shapes.

# Continuous Galerkin (cG)

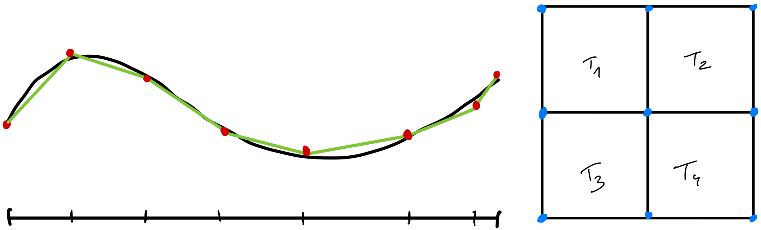



Figure: Continuous galerkin method. The nodes are shared by the corresponding elements.

# Discontinuous Galerkin (dG)

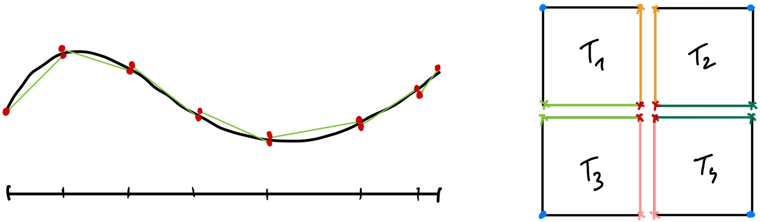



Figure: In discontinuous galerkin method, the elements do not share the nodes. In the two dimensional case, the boundary with same colors are shared amongst the elements, however, with dG, they are treated as "separate" elements

$$V_h = P^k(\mathcal{T}) = \{v_h \in L^2(\Omega) : v_h|_T \in P^k(\mathcal{T}) \forall T \in \mathcal{T}_h\}$$

# Averages and Jumps

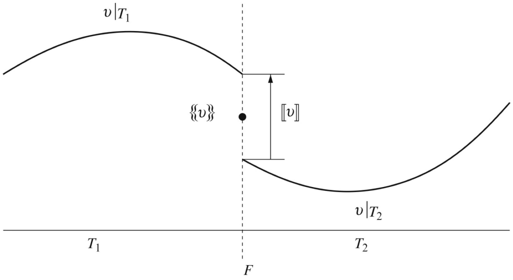



Figure: One-dimensional example of average and jump operators

# Averages and Jumps

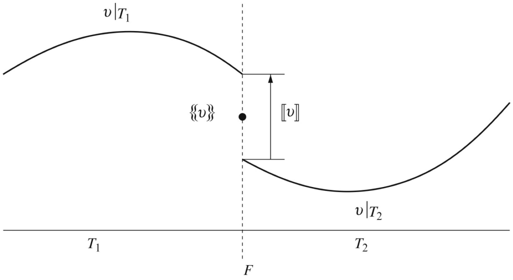



Figure: One-dimensional example of average and jump operators

$$\text{Jump:} \quad [\![v]\!]_F(x) = v|_{T_1}(x) - v|_{T_2}(x)$$

# Averages and Jumps

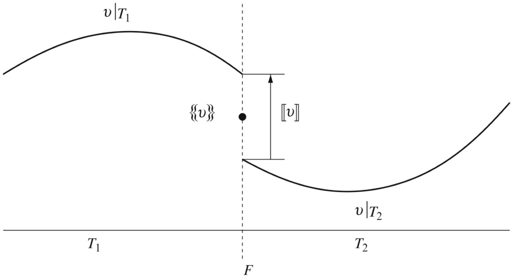



Figure: One-dimensional example of average and jump operators

$$\text{Jump:} \quad [\![v]\!]_F(x) = v\big|_{T_1}(x) - v\big|_{T_2}(x)$$

$$\text{Average:} \quad \{\{v\}\}_F(x) := \frac{1}{2}\left[v\big|_{T_1}(x) - v\big|_{T_2}(x)\right]$$

# Averages and Jumps

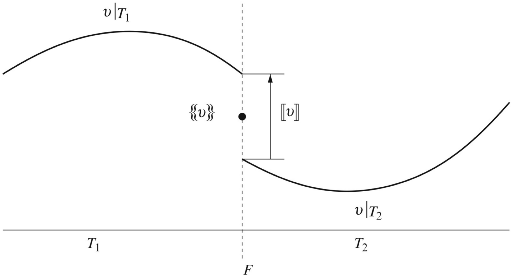



Figure: One-dimensional example of average and jump operators

$$\text{Jump:} \quad [\![v]\!]_F(x) = v|_{T_1}(x) - v|_{T_2}(x)$$

$$\text{Average:} \quad \{\{v\}\}_F(x) := \frac{1}{2}\Big[v|_{T_1}(x) - v|_{T_2}(x)\Big]$$

At the boundary of the domain: $\{\{v\}\}_F(x) = [\![v]\!]_F(x) := v|_T(x)$

# Penalty and numerical Fluxes

**Numerical Flux**

Introduces coupling between different sub problems.

Allows us to recover global solution (flux needs to be conservative between elements).

Local solution on a particular element is dependent upon the local solution from nearby elements through the numerical flux function.

# Penalty and numerical Fluxes

**Numerical Flux**

Introduces coupling between different sub problems.

Allows us to recover global solution (flux needs to be conservative between elements).

Local solution on a particular element is dependent upon the local solution from nearby elements through the numerical flux function.

**Weakly enforce**:

Continuity of the flux: $[\![\nabla u]\!] = 0$ over all facets

Continuity of the solution: $[\![u]\!] = 0$ over all facets

Stability

Short Recap to Poisson equation

# Short Example: Diffusion Equation

Diffusion / poission equation in weak form

$$
\begin{aligned}
-\Delta u = f \quad & in \quad \Omega \\
u = 0 \quad & on \quad \partial\Omega
\end{aligned}
$$

# Short Example: Diffusion Equation

Diffusion / poission equation in weak form

$$-\Delta u = f \quad in \quad \Omega$$
$$u = 0 \quad on \quad \partial\Omega$$

the weak formulation is given by: (Proof by general agreement)[1]:

$$u \in V \quad s.t. \quad a(u,v) = \int_\Omega fv \quad \forall v \in V, V = H_0^1(\Omega)$$

[1] All in favour..?

# Short Example: Diffusion Equation

Diffusion / poission equation in weak form

$$
\begin{aligned}
-\Delta u &= f \quad in \quad \Omega \\
u &= 0 \quad on \quad \partial\Omega
\end{aligned}
$$

the weak formulation is given by: (Proof by general agreement)[1]:

$$u \in V \quad s.t. \quad a(u,v) = \int_\Omega fv \quad \forall v \in V, V = H_0^1(\Omega)$$

$$\ni a(u,v) := \int_\Omega \nabla u \cdot \nabla v$$

[1] All in favour..?

# Poisson Equation: SIP Formulation

$a_h(u_h, v_h) :=$

$\sum_{T\in\mathcal{T}} \int_T \nabla u_h \cdot \nabla v_h dx$

# Poisson Equation: SIP Formulation

$$a_h(u_h, v_h) :=$$

$$\sum_{T\in\mathcal{T}} \int_T \nabla u_h \cdot \nabla v_h dx - \overbrace{\sum_{F\in\mathcal{F}} \int_F \{\{\nabla u_h\}\} \cdot \mathbf{n} [\![ v_h ]\!] dS}^{\text{Consistency}}$$

# Poisson Equation: SIP Formulation

$$a_h(u_h, v_h) :=$$

$$\sum_{T\in\mathcal{T}} \int_T \nabla u_h \cdot \nabla v_h dx - \overbrace{\sum_{F\in\mathcal{F}} \int_F \{\{\nabla u_h\}\} \cdot \mathbf{n}[\![v_h]\!] dS}^{\text{Consistency}} - \overbrace{\sum_{F\in\mathcal{F}} \int_F \{\{\nabla v_h\}\} \cdot \mathbf{n}[\![u_h]\!] dS}^{\text{Symmetry}}$$

# Poisson Equation: SIP Formulation

$$a_h(u_h, v_h) :=$$

$$\textstyle\sum_{T\in\mathcal{T}} \int_T \nabla u_h \cdot \nabla v_h dx - \overbrace{\sum_{F\in\mathcal{F}} \int_F \{\{\nabla u_h\}\} \cdot \mathbf{n}[\![v_h]\!] dS}^{\text{Consistency}} - \overbrace{\sum_{F\in\mathcal{F}} \int_F \{\{\nabla v_h\}\} \cdot \mathbf{n}[\![u_h]\!] dS}^{\text{Symmetry}}$$

$$+ \underbrace{\sum_{F\in\mathcal{F}} \frac{\eta}{h} \int_F [\![u_h]\!][\![v_h]\!] dS}_{\text{Penalty}}$$

# Poisson Equation: SIP Formulation

$$
\mathrm{a}_h(u_h, v_h) :=
$$

$$
\sum_{T\in\mathcal{T}} \int_T \nabla u_h \cdot \nabla v_h dx - \overbrace{\sum_{F\in\mathcal{F}} \int_F \{\{\nabla u_h\}\} \cdot \mathbf{n}[\![v_h]\!] dS}^{\text{Consistency}} - \overbrace{\sum_{F\in\mathcal{F}} \int_F \{\{\nabla v_h\}\} \cdot \mathbf{n}[\![u_h]\!] dS}^{\text{Symmetry}}
$$

$$
+ \underbrace{\sum_{F\in\mathcal{F}} \frac{\eta}{h} \int_F [\![u_h]\!][\![v_h]\!] dS}_{\text{Penalty}}
$$

## Integration domain

Unlike cG, integration in this case is not done over $\Omega$, rather it is limited to each element $T \in \mathcal{T}_h$.

# cG vs dG

**cG**

$$a_h(u_h, v_h) = \int_\Omega f v_h \forall v_h \in U_h$$

* Considers entire domain.
* DoF on the edges are shared.

**dG**

$$a_h(u_h, v_h) + \mathrm{SIP} = \int_\Omega f v_h \forall v_h \in U_h$$

* Local to the generating element.
* Numerical flux is used to derive a uniquely defined value of the quantities of interest
* Comparatively more degrees of freedom.

DG (in a weak way) does a better job at conserving fluxes at a same computational cost (it is more useful to use a dG method than to use a finer mesh).

# cG vs dG

**cG**

$$a_h(u_h, v_h) = \int_\Omega f v_h \forall v_h \in U_h$$

* Considers entire domain.
* DoF on the edges are shared.

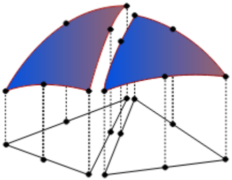
**dG**

$$a_h(u_h, v_h) + \text{SIP} = \int_\Omega f v_h \forall v_h \in U_h$$

* Local to the generating element.
* Numerical flux is used to derive a uniquely defined value of the quantities of interest
* Comparatively more degrees of freedom.

DG (in a weak way) does a better job at conserving fluxes at a same computational cost (it is more useful to use a dG method than to use a finer mesh).

# dG: Poission Equation



## Stokes Equation

# Stokes Equation

$$
\begin{aligned}
-\triangle u + \nabla p &= f \quad in \quad \Omega \\
\nabla \cdot u &= 0 \quad in \quad \Omega \\
u &= 0 \quad on \quad \partial\Omega
\end{aligned}
$$

# Stokes Equation

$$
\begin{aligned}
-\triangle u + \nabla p = f \quad in \quad \Omega \\
\nabla \cdot u = 0 \quad in \quad \Omega \\
u = 0 \quad on \quad \partial\Omega
\end{aligned}
$$

The corresponding discrete bi-linear form is given as:

# Stokes Equation

$$
\begin{aligned}
-\triangle u + \nabla p = f \quad in \quad \Omega \\
\nabla \cdot u = 0 \quad in \quad \Omega \\
u = 0 \quad on \quad \partial\Omega
\end{aligned}
$$

The corresponding discrete bi-linear form is given as:

$$
\begin{aligned}
a_h(u_h, v_h) + b_h(v_h, p_h) = \int_\Omega f \cdot v_h \quad \forall v_h \in U_h \\
-b_h(u_h, q_h) = 0 \quad \forall q_h \in P_h
\end{aligned}
$$

# Stokes Equation

$$
\begin{aligned}
-\triangle u + \nabla p = f \quad &in \quad \Omega \\
\nabla \cdot u = 0 \quad &in \quad \Omega \\
u = 0 \quad &on \quad \partial\Omega
\end{aligned}
$$

The corresponding discrete bi-linear form is given as:

$$
\begin{aligned}
a_h(u_h, v_h) + b_h(v_h, p_h) &= \int_\Omega f \cdot v_h \quad \forall v_h \in U_h \\
-b_h(u_h, q_h) &= 0 \quad \forall q_h \in P_h
\end{aligned}
$$

Wherein, $f \in [L^2(\Omega)]^d, u \in U := [H_0^1(\Omega)]^d, p \in P := [L_0^2(\Omega)]$, we look for solution in $X := U \times P$.

# Stokes Equation

$$
\begin{aligned}
-\triangle u + \nabla p = f \quad & in \quad \Omega \\
\nabla \cdot u = 0 \quad & in \quad \Omega \\
u = 0 \quad & on \quad \partial\Omega
\end{aligned}
$$

The corresponding discrete bi-linear form is given as:

$$
\begin{gathered}
a_h(u_h, v_h) + b_h(v_h, p_h) = \int_\Omega f \cdot v_h \quad \forall v_h \in U_h \\
-b_h(u_h, q_h) = 0 \quad \forall q_h \in P_h
\end{gathered}
$$

Wherein, $f \in [L^2(\Omega)]^d, u \in U := [H_0^1(\Omega)]^d, p \in P := [L_0^2(\Omega)]$, we look for solution in $X := U \times P$.
$u_h, p_h$ belong to the respective finite element spaces of U and P respectively.

# Discreete Spaces for dG

The solution for discontinous gelarkin method lies in the in $X_h := U_h \times P_h$ where

$$U_h := [\mathbb{P}_d^k(\mathcal{T}_h)]^d$$

and

$$P_h := [\mathbb{P}_{d,0}^k(\mathcal{T}_h)]^d$$

such that $[\mathbb{P}_d^k(\mathcal{T}_h)]^d$ is the broken polynomial space.

# Stokes Equation (SIP formulation)

We consider the equal order discontinuous velocity and pressure.

$$
\begin{aligned}
a_h^{sip}(u_h, v_h) + b_h(v_h, p_h) &= \int_\Omega f \cdot v_h \quad \forall v_h \in U_h \\
-b_h(u_h, q_h) + s_h(p_h, q_h) &= 0 \quad \forall q_h \in P_h
\end{aligned}
$$

# Stokes Equation (SIP formulation)

We consider the equal order discontinuous velocity and pressure.

$$\begin{aligned} a_h^{sip}(u_h, v_h) + b_h(v_h, p_h) &= \int_\Omega f \cdot v_h \quad \forall v_h \in U_h \\ -b_h(u_h, q_h) + s_h(p_h, q_h) &= 0 \quad \forall q_h \in P_h \end{aligned}$$

Where $s_h(p_h, q_h) := \sum_{F \in \mathcal{F}_h^i} h_F \int_F [\![q_h]\!][\![p_h]\!]$ is the stablization meant to control pressure jumps across interfaces.

# Stokes Equation (SIP formulation)

We consider the equal order discontinuous velocity and pressure.

$$a_h^{sip}(u_h, v_h) + b_h(v_h, p_h) = \int_\Omega f \cdot v_h \quad \forall v_h \in U_h$$
$$-b_h(u_h, q_h) + s_h(p_h, q_h) = 0 \quad \forall q_h \in P_h$$

Where $s_h(p_h, q_h) := \sum_{F \in \mathcal{F}_h^i} h_F \int_F [\![q_h]\!][\![p_h]\!]$ is the stablization meant to control pressure jumps across interfaces.

# Stokes Equation (SIP formulation)

and the discreetization of pressure velocity copuling is given as

$$b_h(q_h, v_h) = \int_\Omega v_h \cdot \nabla_h q_h + \sum_{F \in \mathcal{F}_h} \int_F \{\{v_h\}\} \cdot n_F [\![v_h]\!]$$

Hence, the discreetization of Stokes equation is given as:

$$c_h((u_h, p_h), (v_h, p_h)) := a_h(u_h, v_h) + b_h(v_h, p_h) - b_h(u_h, q_h) + s_h(p_h, q_h) \tag{1}$$

It can be shown that the given formulation is Discrete inf-sup stable (Lemma 6.13) in [1].

# Implementation

# General Implementation of FE solver

**Require:** Create / import mesh.

**Ensure:** Prepare system (initialize polynomial space, linear solver etc.).

Set up boundary conditions.<br>
**while** $t \leq T$ or !mesh_refinement **do**<br>
    Set up boundary conditions.<br>
    call the local assembler (assemble_system)<br>
    Solve System<br>
    Compute Error<br>
**end while**<br>
Post Processing

# General Implementation of FE solver

**Require:** Create / import mesh.
**Ensure:** Prepare system (initialize polynomial space, linear solver etc.).

```
Set up boundary conditions.
while t ≤ T or !mesh_refinement do
    Set up boundary conditions.
    call the local assembler (assemble_system)
    Solve System
    Compute Error
end while
Post Processing
```

Weak formulation is defined in the local assembler.

```
code: local_assembler.h
```

# General Implementation of FE solver

```
for (int q = 0; q < num_q; ++q){
  ...
  // loop over test DOFs <-> test function v
  for (int i = 0; i < num_dof; ++i){
    // loop over trrial DOFs <-> trial function u
    for (int j = 0; j < num_dof; ++j) {
      lm(i, j) += wq * (dot(phi_j, phi_i) * dJ;
    }
  }
}
```

# Implementation

- Compared to the cG method (shown in previous slide), implementation of dG method has some extra steps.
- The construction of local assembler is the same.
- SIP formulation is added within the local assembler.
- Jumps and Averages are calculated based on the interface (boundary/interior).

# Implementation

- Compared to the cG method (shown in previous slide), implementation of dG method has some extra steps.
- The construction of local assembler is the same.
- SIP formulation is added within the local assembler.
- Jumps and Averages are calculated based on the interface (boundary/interior).

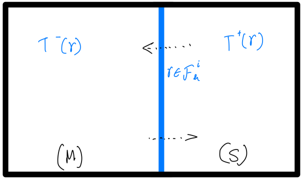



Figure: hiFlow employs *master/slave* paradigm $n_m$, $n_s$ is the unit normal for master and slave element respectively.

# Implementation SIP

```
for "interior interfaces" e:
    for "trial_functions" in (m, s):
        for "test_functions" in (m, s):
            compute: integral_over_interface
```

where **intergral_over_interface** $= \int_e [\![\phi_i^{test}]\!] \cdot [\![\phi_j^{trial}]\!]$

$$\phi_i^{test}|_m = \begin{cases} \phi_i^{test} & \text{test} = \text{m} \\ 0 & \text{test} = \text{s} \end{cases} \qquad \phi_i^{test}|_s = \begin{cases} 0 & \text{test} = \text{m} \\ \phi_i^{test} & \text{test} = \text{s} \end{cases}$$

# Implementation - Boundary Conditions

To distinguish between boundary and interior facets, hiFlow3 uses material numbers. The normal acts differently on interior facets and boundary facests,

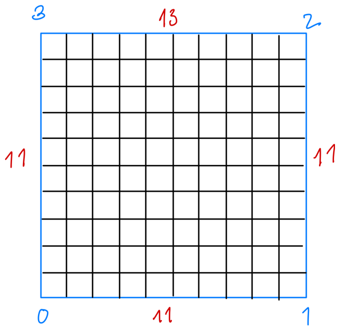



Figure: Material number of the boundary is given in red. The black boxes inside the square have material number less than 11

# Implementation - Boundary Conditions (cont.)

To distinguish between boundary and interior facets, hiFlow3 uses material numbers. The normal acts differently on interior facets and boundary facests,

```
int get_if_type(const Entity &face) const{
    const int mat_num = face.get_material_number();
    if (mat_num >= 11){
        return 1; // dirichlet
    }
    return 0;   // interior
}
```

# Implementation - Assembly

```
code dg_CavityStokes.h
```

*In theory, theory and practice are the same; in practice, they're not.*

# Poission Equation

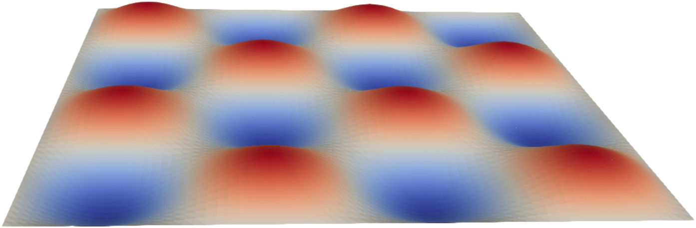


Figure: Poission Equation with Discontinous Galerkin Method

# Stokes dG

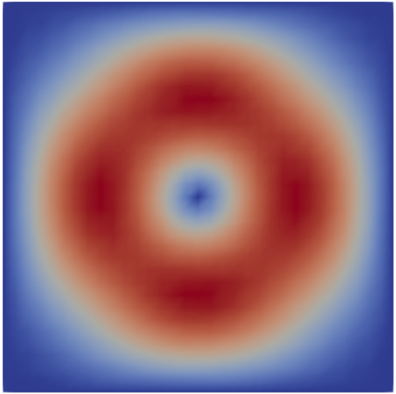


Figure: Stokes Equation with dirichlet boundary $= 0$ on all four sides.

# Error Norms

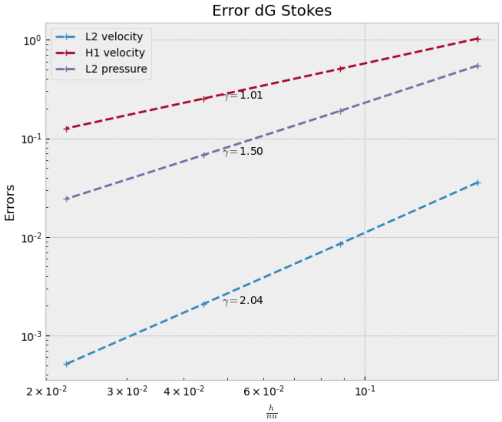



Figure: Stokes equation with 0 Dirichlet boundary condition and $\eta = 10$

# Error Norms: cG vs dG

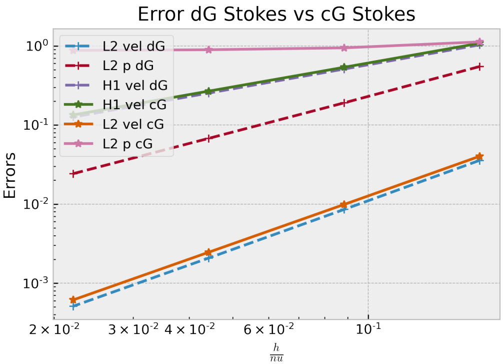
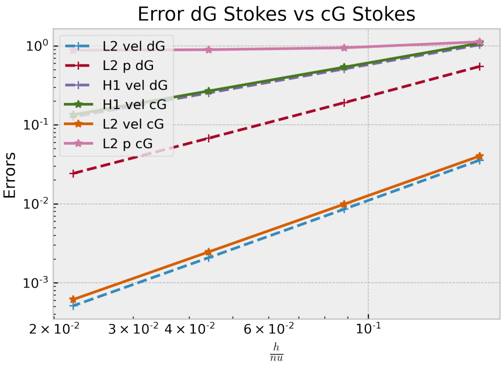



Figure: Error norm for cG and dG methods for stokes equation with $u = 0 \quad \forall u \in \partial\Omega$. The dashed lines represent errors with dG, while the straight lines represent errors with cG method.

# Error Norms: cG vs dG




| $\gamma$ | L2 Pressure | L2 Velocity | H1 Velocity |
|---|---|---|---|
| cG | 0.118 | 2.010 | 1.003 |
| dG | 1.499 | 2.041 | 1.009 |

Figure: Error norm for cG and dG methods for stokes equation with $u = 0 \quad \forall u \in \partial\Omega$. The dashed lines represent errors with dG, while the straight lines represent errors with cG method.

# Lid Driven Cavity

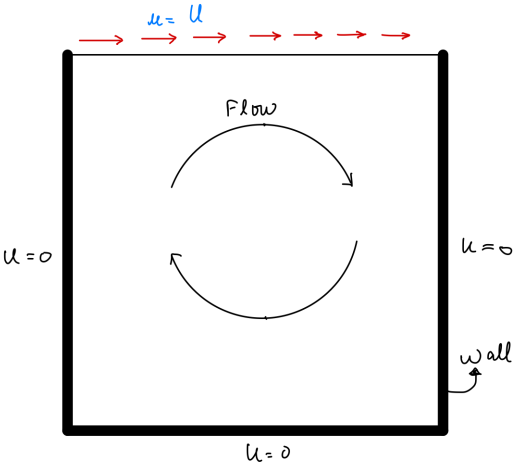



Figure: Lid driven cavity

# Lid Driven Cavity

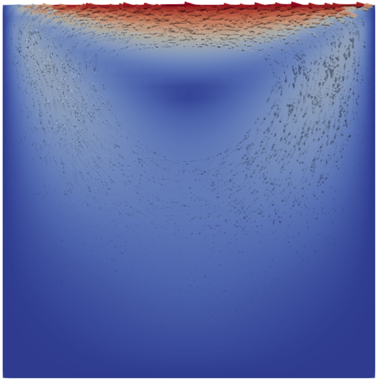


Figure: Velocity

Figure: Pressure

# The conundrum of $\eta$

The inf-sup stability of discrete formulation is given by Lemma 6.13 in [1] , which assumes that the penalty parameter $\eta > \underline{\eta}$ where $\underline{\eta} := C_{tr}^2 N_\partial$ (Lemma 4.12), $C_{tr} = \sqrt{k(k+d)}$

# The conundrum of $\eta$

The inf-sup stability of discrete formulation is given by Lemma 6.13 in [1] , which assumes that the penalty parameter $\eta > \underline{\eta}$ where $\underline{\eta} := C_{tr}^2 N_\partial$ (Lemma 4.12), $C_{tr} = \sqrt{k(k+d)}$
Essentially it says that if the penalty parameter $\eta$ is large enough, the SIP bilinear form is coercive on $V_h$

# The conundrum of $\eta$

The inf-sup stability of discrete formulation is given by Lemma 6.13 in [1] , which assumes that the penalty parameter $\eta > \underline{\eta}$ where $\underline{\eta} := C_{tr}^2 N_\partial$ (Lemma 4.12), $C_{tr} = \sqrt{k(k+d)}$ Essentially it says that if the penalty parameter $\eta$ is large enough, the SIP bilinear form is coercive on $V_h$

However, in the *implementation* of stokes equations above, this is not strictly true.[2]

[2]Q: What's a dilemma? A: a lemma that produces two results.

# The conundrum of $\eta$

The inf-sup stability of discrete formulation is given by Lemma 6.13 in [1] , which assumes that the penalty parameter $\eta > \underline{\eta}$ where $\underline{\eta} := C_{tr}^2 N_\partial$ (Lemma 4.12), $C_{tr} = \sqrt{k(k+d)}$ Essentially it says that if the penalty parameter $\eta$ is large enough, the SIP bilinear form is coercive on $V_h$

However, in the *implementation* of stokes equations above, this is not strictly true.[2]


[2] Q: What's a dilemma? A: a lemma that produces two results.

# Lid Driven Cavity - Varying $\eta$

Calculation for eta for k = 1, d = 2

# Lid Driven Cavity - Varying $\eta$

Calculation for eta for k = 1, d = 2

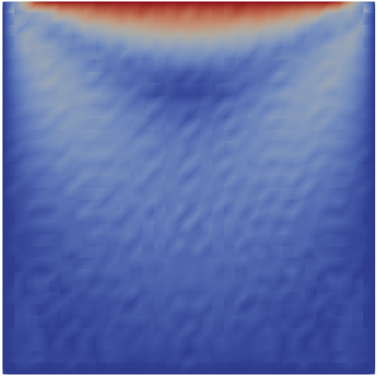


Figure: $\eta = -10$

# Lid Driven Cavity - Varying $\eta$

Calculation for eta for k = 1, d = 2

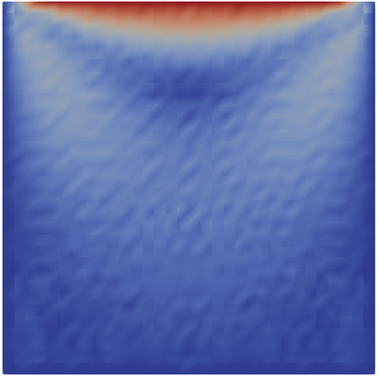


Figure: $\eta = -10$

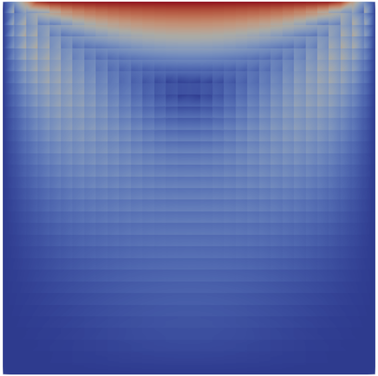


Figure: $\eta = 1$

# Lid Driven Cavity - Varying $\eta$

Figure: $\eta = -100$

# Lid Driven Cavity - Varying $\eta$

Figure: $\eta = -100$

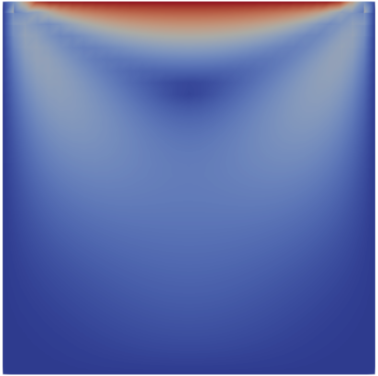


Figure: $\eta = 2$

# Linear Solvers: $\eta$

GMRES residuals for different $\eta$ (Lid Driven Cavity)

# Linear Solvers: $\eta$

GMRES residuals for different $\eta$ (Lid Driven Cavity)

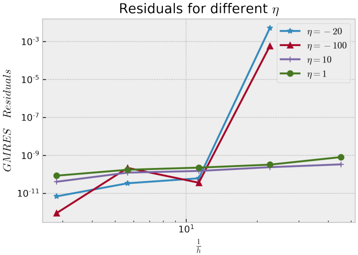



Figure: GMRES residuals for different $\eta$ (Lid Driven Cavity)

Table: Iterations. First column (bold) represents *eta* values

| **-100** | **-20** | **1** | **10** |
|---|---|---|---|
| 72 | 67 | 34 | 51 |
| 167 | 139 | 38 | 67 |
| 389 | 347 | 52 | 107 |
| - | - | 58 | 131 |
| - | - | 70 | 190 |
| - | - | - | - |
| - | - | - | - |

# Linear Solvers: cG vs dG

GMRES Residuals for cG vs dG

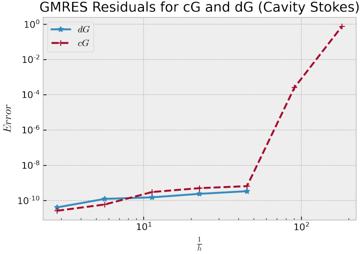



Figure: cG dG GMRES residuals (stokes, dbc = 0)

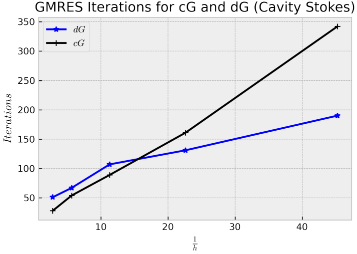



Figure: cG dG GMRES iterations (stokes, dbc = 0)

# Linear Solvers: Equal Order Convergence

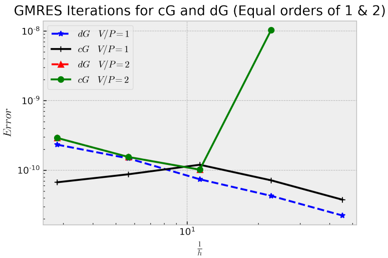



Figure: dG cG equal order (k=1,2) residuals

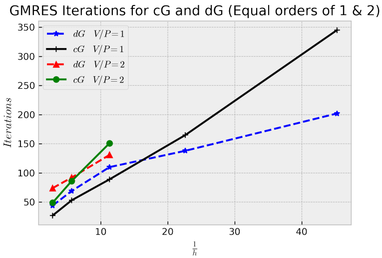



Figure: cG dG equal order (k=1,2) iterations

# The curious case of Domain Decomposition

Stokes simulation on 48 cores.

# The curious case of Domain Decomposition

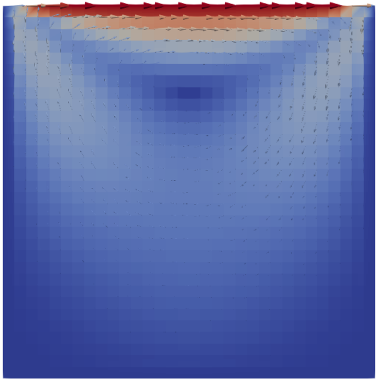
Stokes simulation on 48 cores.

| dG | | |
|---|---|---|
| Refinement | Iterations | Residual |
| 4 | 429 | 1.00E-10 |
| 5 | 454 | 1.57E-08 |
| 6 | 469 | 2.66E-11 |

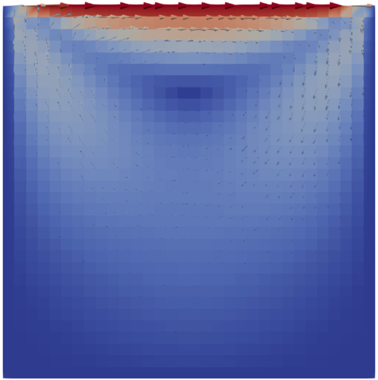
| cG | | |
|---|---|---|
| Refinenment | Iterations | Residual |
| 4 | 1000 | 2.94E-07 |
| 5 | 1000 | 0.237797 |
| 6 | 1000 | 0.00347334 |

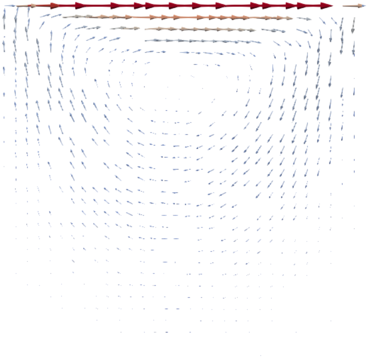
# Non Symmetric Penalty, $\eta = 1$

# Non Symmetric Penalty, $\eta = 1$



# Non Symmetric Penalty, $\eta = 1$





# Conclusion

1.

# Further Reading

1. Ocellaris: mass conserving dG solver.

# Further Reading

1. Ocellaris: mass conserving dG solver.
2. Nodal Discontinuous Galerkin Methods (Hesthaven, Warburton)

# Further Reading

1. Ocellaris: mass conserving dG solver.
2. Nodal Discontinuous Galerkin Methods (Hesthaven, Warburton)

# Thank You

**Appendix**

# Convergence wars: Return of the pressure stabilization

| cG | | |
|---|---|---|
| Refinements | Iterations | Residuals |
| 2 | 16 | 6.54E-11 |
| 3 | 21 | 4.73E-11 |
| 4 | 27 | 2.52E-10 |
| 5 | 35 | 2.86E-10 |
| 6 | 56 | 5.04E-10 |
| 7 | 143 | 9.42E-10 |

| dG | | |
|---|---|---|
| Refinements | Iterations | Residuals |
| 2 | 39 | 2.17E-11 |
| 3 | 52 | 1.10E-10 |
| 4 | 85 | 1.63E-10 |
| 5 | 116 | 2.46E-10 |
| 6 | 170 | 3.39E-10 |
| 7 | 295 | 4.65E-10 |

Figure: dG method with a different pressure stabilization scheme converges even at higher refinement level. cG Performs preferably better than dG

# NS Formulation

The weak formulation for Navier stokes builds up on stokes equation.

# NS Formulation

The weak formulation for Navier stokes builds up on stokes equation. We formulated stokes equation as:

$$c((u,p),(v,q)) = \int_\Omega f \cdot v$$

# NS Formulation

The weak formulation for Navier stokes builds up on stokes equation. We formulated stokes equation as:

$$c((u, p), (v, q)) = \int_\Omega f \cdot v$$

upon this we add the trilinear form of the convection term:

$$c((u, p), (v, q)) + t(u, u, v) = \int_\Omega f \cdot v \tag{2}$$

# NS Formulation

The weak formulation for Navier stokes builds up on stokes equation. We formulated stokes equation as:

$$c((u,p),(v,q)) = \int_\Omega f \cdot v$$

upon this we add the trilinear form of the convection term:

$$c((u,p),(v,q)) + t(u,u,v) = \int_\Omega f \cdot v \tag{2}$$

Where $t(u,u,v) := \int_\Omega (w \cdot \nabla u) \cdot v = (u \cdot \nabla) u$

# NS Formulation

The weak formulation for Navier stokes builds up on stokes equation. We formulated stokes equation as:

$$c((u,p),(v,q)) = \int_\Omega f \cdot v$$

upon this we add the trilinear form of the convection term:

$$c((u,p),(v,q)) + t(u,u,v) = \int_\Omega f \cdot v \tag{2}$$

Where $t(u,u,v) := \int_\Omega (w \cdot \nabla u) \cdot v = (u \cdot \nabla) u$
We can show that the given trilinear form is bounded by a domain parameter $t_\Omega$ (Lemma 6.32 in [1]) and it's existance and uniqueness by Thm 6.36 in [1].

# NS Formulation

Moreover, We consider a modified form of the trilinear form $t$

$$t'(w,u,v) = t(w,u,v) + \frac{1}{2}\int_\Omega (\nabla \cdot w) u \cdot v = \int_\Omega (w \cdot \nabla u) \cdot v + \frac{1}{2}\int_\Omega (\nabla \cdot w) u \cdot v$$

to satisfy the skew symmetry of triliner form when dealing with dG approximations.

# NS Formulation

Moreover, We consider a modified form of the trilinear form $t$

$$t'(w,u,v) = t(w,u,v) + \frac{1}{2}\int_\Omega (\nabla\cdot w)u\cdot v = \int_\Omega (w\cdot\nabla u)\cdot v + \frac{1}{2}\int_\Omega (\nabla\cdot w)u\cdot v$$

to satisfy the skew symmetry of triliner form when dealing with dG approximations. The INS formulation is then given by:

$$c((u,p),(v,q)) + t'(u,u,v) = \int_\Omega f\cdot v$$

(3)

# NS Formulation - Discrete Setting

The discrete trilinear form is defined as:

$$t_h(w_h, u_h, v_h) := \int_\Omega (w_h \cdot \nabla_h u_h) + \frac{1}{2}\int_\Omega (\nabla_h \cdot w_h) \cdot (u_h \cdot v_h)$$
$$- \sum_{F \in \mathcal{F}_h^i} \int_F \{\{w_h\}\} \cdot n_F [\![u_h]\!] \cdot \{\{v_h\}\} - \frac{1}{2} \sum_{F \in \mathcal{F}_h} \int_F [\![w_h]\!] \cdot n_F \{\{u_h \cdot v_h\}\}$$

The discrete trilinear form is bounded. [1]

# NS Discrete Formulation

Finally, we have the dG discrete formulation as:

$$\mathcal{V} a_h(u_h, v_h) + t_h(u_h, u_h, v_h) + b_h(v_h, p_h) = \int_\Omega f \cdot v_h \quad \forall v_h \in U_h$$

$$-b_h(u_h, q_h) + \mathcal{V}^{-1} s_h(p_h, q_h) = 0 \quad \forall q_h \in P_h$$

The solution for the above formulation exists, and is unique [1] (Lemma 6.41). It can also be shown that the discrete formulation show above converges to the unique solution of 2 [1] (Thm 6.47)

# dG notations

- $\mathcal{F}_h := \mathcal{F}_h^i \cup \mathcal{F}_h^b$, where $\mathcal{F}_h^i$ are the interior interfaces
- $\mathcal{F}_T := \{\mathcal{F} \in \mathcal{F}_h | F \subset \partial T\}$
- $\mathcal{T}_F := \{T \in \mathcal{T}_h | F \subset \partial T\}$
- $n_F$ is the face normal, orientation of $n_F$ is arbitrary depending on the choice of $T_1$ and $T_2$.

# dG notations

- $\mathcal{F}_h := \mathcal{F}_h^i \cup \mathcal{F}_h^b$, where $\mathcal{F}_h^i$ are the interior interfaces
- $\mathcal{F}_T := \{\mathcal{F} \in \mathcal{F}_h | F \subset \partial T\}$
- $\mathcal{T}_F := \{T \in \mathcal{T}_h | F \subset \partial T\}$
- $n_F$ is the face normal, orientation of $n_F$ is arbitrary depending on the choice of $T_1$ and $T_2$.
- $\mathbb{P}_d^k(\mathcal{T}_h)$ is the broken polynomial space restricted to each element $T$

# dG notations

- $\mathcal{F}_h := \mathcal{F}_h^i \cup \mathcal{F}_h^b$, where $\mathcal{F}_h^i$ are the interior interfaces
- $\mathcal{F}_T := \{\mathcal{F} \in \mathcal{F}_h | F \subset \partial T\}$
- $\mathcal{T}_F := \{T \in \mathcal{T}_h | F \subset \partial T\}$
- $n_F$ is the face normal, orientation of $n_F$ is arbitrary depending on the choice of $T_1$ and $T_2$.
- $\mathbb{P}_d^k(\mathcal{T}_h)$ is the broken polynomial space restricted to each element $T$
- Similarly, broken gradient, $H(div; \Omega)$, and Sobolev space are restricted to each element $T$.

# dG notations

- $\mathcal{F}_h := \mathcal{F}_h^i \cup \mathcal{F}_h^b$, where $\mathcal{F}_h^i$ are the interior interfaces
- $\mathcal{F}_T := \{\mathcal{F} \in \mathcal{F}_h | F \subset \partial T\}$
- $\mathcal{T}_F := \{T \in \mathcal{T}_h | F \subset \partial T\}$
- $n_F$ is the face normal, orientation of $n_F$ is arbitrary depending on the choice of $T_1$ and $T_2$.
- $\mathbb{P}_d^k(\mathcal{T}_h)$ is the broken polynomial space restricted to each element $T$
- Similarly, broken gradient, $H(div; \Omega)$, and Sobolev space are restricted to each element $T$.

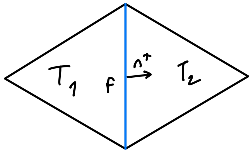



Figure: Notation of an interface

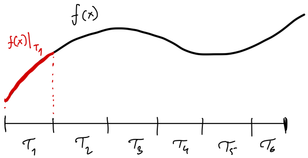



Figure: Restriction of $f(x)$ in $T_1$ is highlighted by the red region.

# References

Di Pietro, D. & Ern, A. Mathematical aspects of discontinuous Galerkin methods. (Springer Science Business Media,2011)

Epshteyn, Y. & Rivière, B. Estimation of penalty parameters for symmetric interior penalty Galerkin methods. *Journal Of Computational And Applied Mathematics*. **206**, 843-872 (2007), https://www.sciencedirect.com/science/article/pii/S0377042706005279