

# Stable and robust projection methods for the incompressible Navier–Stokes equations using high-order hybridizable discontinuous Galerkin schemes

Corbin Foucart<sup>a</sup>, Pierre F.J. Lermusiaux<sup>a,\*</sup>

<sup>a</sup>*Department of Mechanical Engineering, Computational Science and Engineering, Massachusetts Institute of Technology, 77 Massachusetts Avenue, Cambridge, 02139, MA, USA*

---

## Abstract

abstract.tex

*Keywords:* Nonhydrostatic modeling, Finite element methods, Discontinuous Galerkin methods

---

## 1. Introduction

In [4], the authors investigate the stability and robustness of DG discretizations of several projection methods. They compare fully-implicit, high-order dual splitting [8], and pressure-correction schemes, but weirdly discretize the advection term implicitly for the pressure-correction, making the system nonlinear. So not exactly a fair comparison, since solving a nonlinear problem defeats the purpose of a projection method.

The numerical experiments in [4] suggest that the high-order, dual splitting scheme

### 1.1. Incompressible Navier–Stokes

$$\begin{aligned} \frac{\partial \mathbf{u}}{\partial t} + \nabla \cdot (\mathbf{u} \otimes \mathbf{u}) - \nabla \cdot (\nu \nabla \mathbf{u}) + \nabla p &= \mathbf{f} & \text{on } \Omega \times [0, T] \\ \nabla \cdot \mathbf{u} &= 0 & \text{on } \Omega \times [0, T] \end{aligned} \quad (1)$$

The incompressible Navier–Stokes equations are subject to the initial condition

$$\mathbf{u}(\mathbf{x}, t = 0) = \mathbf{u}_0 \text{ on } \Omega, \quad (2)$$

where  $\mathbf{u}_0$  is divergence-free. We denote the outward-facing unit normal vector as  $\mathbf{n}$ . On the boundary  $\Gamma$ , we prescribe Dirichlet and Neumann conditions on  $\Gamma_D$  and  $\Gamma_N$ , respectively, such that  $\Gamma_D \cup \Gamma_N = \Gamma$ .

$$\mathbf{u} = \mathbf{g}_D^u \quad \text{on } \Gamma_D \times [0, T] \quad (3)$$

$$(\mathbf{F}_v(\mathbf{u}) - p\mathbf{I}) \cdot \mathbf{n} = \mathbf{g}_N^{\text{stress}} \quad \text{on } \Gamma_D \times [0, T] \quad (4)$$

Where  $\mathbf{F}_v(\mathbf{u})$  is a representation of the viscous flux, usually given as  $\mathbf{F}_v(\mathbf{u}) = \nu \nabla \mathbf{u}$ . The operator splitting associated with projection methods will necessitate splitting the boundary condition as well, so we decompose the stress condition into viscous and pressure components  $\mathbf{g}_N^{\text{stress}} = \mathbf{g}_N^u - g_N^p \mathbf{n}$  and prescribe them separately, following [4],

$$\mathbf{F}_v(\mathbf{u}) \cdot \mathbf{n} = \mathbf{g}_N^u \quad \text{on } \Gamma_N, \quad (5)$$

$$p = g_N^p \quad \text{on } \Gamma_N. \quad (6)$$

We use the Rothe method, handling the temporal discretization and operator splitting before spatial discretization. For time integration, we apply backward differentiation formula (BDF) for all schemes in this paper.

---

\*Corresponding author,

Email addresses: [foucartc@mit.edu](mailto:foucartc@mit.edu) (Corbin Foucart), [pierrel@mit.edu](mailto:pierrel@mit.edu) (Pierre F.J. Lermusiaux)

## 2. Projection methods

An overview of the temporal discretization and operator splitting for pressure-correction schemes is given in [5]. Results from [5]: (1) under certain smoothness requirements on the solution, the nonlinear advection term in the Navier–Stokes equations does not affect the convergence rates of the splitting errors, and they treat it explicitly.

### 2.0.1. Velocity predictor step

An intermediate predictor velocity  $\bar{\mathbf{u}}$  is calculated by solving the momentum equation with an explicit extrapolation of the pressure gradient term and explicit treatment of the advection term

$$\frac{\beta_s \bar{\mathbf{u}} - \sum_{i=0}^{s_u-1} (\beta_i \mathbf{u}^{n-i})}{\Delta t} - \nabla \cdot (\nu \nabla \bar{\mathbf{u}}) = - \sum_{i=0}^{s_p-1} (\gamma_i \nabla p^{n-i}) - \nabla \cdot (\mathbf{u}^n \otimes \mathbf{u}^n) + \mathbf{f}(t_{n+1}), \quad (7)$$

where the boundary conditions for the predictor velocity are

$$\begin{aligned} \bar{\mathbf{u}} &= \mathbf{g}_D^u(t_{n+1}) & \text{on } \Gamma_D, \\ (\nu \nabla \bar{\mathbf{u}}) \cdot \mathbf{n} &= \mathbf{g}_N^u(t_{n+1}) & \text{on } \Gamma_N. \end{aligned} \quad (8)$$

### 2.0.2. Pressure corrector step

The second step involves computing a correction  $\delta p^{k+1}$  to the pressure by solving

$$-\nabla^2 \delta p^{k+1} = -\frac{\beta_s}{\Delta t} \nabla \cdot \bar{\mathbf{u}}, \quad (9)$$

subject to the boundary conditions

$$\nabla \delta p^{n+1} \cdot \mathbf{n} = 0 \quad \text{on } \Gamma_D, \quad (10)$$

$$\delta p^{n+1} = g_p(t_{n+1}) - \sum_{i=0}^{s_p-1} (\beta_i g_p(t_{n-i})) \quad \text{on } \Gamma_N. \quad (11)$$

The pressure Poisson equation (9) is obtained by writing the intermediate velocity  $\bar{\mathbf{u}}$  in terms of a Helmholtz decomposition consisting of a solenoidal component  $\mathbf{u}$  (since  $\nabla \cdot \mathbf{u} = 0$ ) and irrotational component  $\nabla \delta p^{n+1}$ ,

$$\frac{\beta_s}{\Delta t} \mathbf{u}^{n+1} + \nabla \delta p^{n+1} = \frac{\beta_s}{\Delta t} \bar{\mathbf{u}}, \quad (12)$$

$$\delta p^{n+1} = p^{n+1} - \sum_{i=0}^{s_p-1} (\beta_i p^{n-i}) + \chi \nu \nabla \cdot \bar{\mathbf{u}}, \quad (13)$$

then taking the divergence of equation (12), making use of the divergence-free constraint  $\nabla \cdot \mathbf{u}^{n+1} = 0$ . Taking  $\chi = 0$  corresponds to the standard formulation and  $\chi = 1$  the rotational formulation of the method, respectively [5]. We consider the rotational formulation hereafter.

### 2.0.3. Projection step

In the third step, the velocity  $\mathbf{u}^{n+1}$  and pressure  $p^{n+1}$  at time  $t_{n+1}$  are obtained by performing updates amounting to the projection of  $\bar{\mathbf{u}}$  onto the space of divergence-free vector fields

$$\mathbf{u}^{n+1} = \bar{\mathbf{u}} - \frac{\Delta t}{\beta_s} \nabla \delta p^{n+1}, \quad (14)$$

$$p^{n+1} = \sum_{i=0}^{s_p-1} (\beta_i p^{n-i}) + \delta p^{n+1} - \nu \nabla \cdot \bar{\mathbf{u}}, \quad (15)$$

Theoretical rates of convergence for the pressure-corrector schemes are given in [5]. The authors suggest that schemes are only conditionally stable for  $s_p \geq 2$ . To ensure unconditional stability, we therefore select  $s_u = 2$  and  $s_p = 1$ . With these choices, the pressure-corrector scheme using the rotational formulation can be expected to be  $\Delta t^2$  accurate in the  $L^2$ -norm of the velocity and  $\Delta t^{3/2}$  accurate in the  $L^2$ -norm of the pressure.

### 3. Spatial discretization

#### 3.1. Notation

Boilerplate

#### 3.2. Finite element spaces

boilerplate

#### 3.3. HDG Pressure-correction scheme

The choice of spatial discretization for the pressure gradient terms and the velocity divergence term is of central importance to the stability and robustness of the projection schemes in [4]. Previous work in [13] conducted limited investigation of the discretization of these terms as they appeared in HDG discretizations of projection methods. This motivates the present work in determining whether the discretization of these two terms is similarly important to the robustness of HDG schemes.

The point of departure for HDG schemes is to write each semi-discretized PDE as a first-order system, which we do for the velocity predictor equation (7) and pressure correction equation (11). The projection step does not require an implicit formulation and is computed directly.

##### 3.3.1. HDG formulation of explicit operators

*Pressure gradient terms.* The pressure gradient term can be integrated by parts and replace  $p_h$  on the element boundary  $\partial K$  with a central numerical flux  $p_h^* = \{\{p_h\}\}$ . Applying the mean operator definitions on the interior and boundary interfaces separately,

$$\text{pg}_h(\mathbf{v}, p_h, g_N^p) = -(\nabla \cdot \mathbf{v}, p_h)_{\mathcal{T}_h} + \langle \mathbf{v}, \{\{p_h\}\} \mathbf{n} \rangle_{\partial \mathcal{T}_h \setminus \Gamma} + \langle \mathbf{v}, g_N^p \mathbf{n} \rangle_{\Gamma_N} + \langle \mathbf{v}, p_h \mathbf{n} \rangle_{\Gamma_D} \quad (16)$$

As in Fehn et al., we consider also an alternate reference formulation used in [6, 13], which does not integrate the pressure gradient by parts,

$$\text{pg}_{h,\text{ref}}(\mathbf{v}, p_h) = (\mathbf{v}, \nabla p_h)_{\mathcal{T}_h} \quad (17)$$

*Advection term.* The advection term

$$F_a(\mathbf{v}, \mathbf{u}_h) \quad (18)$$

*Velocity divergence term.* The HDG formulation of the velocity divergence term pertains to the discretization of terms of the form  $(w, \nabla \cdot \bar{\mathbf{u}}_h)_K$  over an element  $K$ . Just as for the pressure gradient term, we integrate by parts and replace  $\mathbf{u}_h$  on the element boundary  $\partial K$  with a central numerical flux  $\bar{\mathbf{u}}_h^* = \{\{\bar{\mathbf{u}}_h\}\}$ . Applying the mean operator definitions on the interior and boundary interfaces separately,

$$\text{vd}_h(w, \bar{\mathbf{u}}_h, \mathbf{g}_D^u) = (\nabla w, \bar{\mathbf{u}}_h)_{\mathcal{T}_h} + \langle w, \{\{\bar{\mathbf{u}}_h\}\} \cdot \mathbf{n} \rangle_{\partial \mathcal{T}_h \setminus \Gamma} + \langle w, \bar{\mathbf{u}}_h \cdot \mathbf{n} \rangle_{\Gamma_N} + \langle w, \mathbf{g}_D^u \cdot \mathbf{n} \rangle_{\Gamma_D}, \quad (19)$$

where we could have taken  $\mathbf{g}_D^u = \bar{\mathbf{u}}_h$  instead, since this is enforced in equation (8). Just as for the pressure gradient term, we consider also an alternate reference formulation given in [6],

$$\text{vd}_{h,\text{ref}}(w, \bar{\mathbf{u}}_h) = (w, \nabla \cdot \bar{\mathbf{u}}_h)_{\mathcal{T}_h}, \quad (20)$$

which does not perform integration by parts.

note: in [13], We integrate by parts another time (equivalent) but take  $\widehat{\mathbf{u}}_h^{k+1}$  as the HDG flux from the predictor solve, which may be unsound.

In addition to the form of the terms themselves, HDG methods provide different choices of numerical flux which we have to investigate. Using the HDG fluxes themselves as numerical fluxes for these terms turns out to require quite complicated accounting in order to make sure the scheme stays consistent. Since the HDG flux is in some sense an intermediate quantity designed to allow for static condensation and reduction of globally-coupled degrees of freedom, it makes more sense to avoid using these fluxes elsewhere in the time discretization. Further, storing the fluxes incurs additional memory costs and requires correction [13].

### 3.3.2. Velocity predictor

Rewritten as a first-order system, equation (7) takes the form

$$\begin{aligned} \bar{\mathbf{L}} - \nabla \mathbf{u} &= 0 \\ \frac{\beta_s \bar{\mathbf{u}} - \sum_{i=0}^{s_u-1} (\beta_i \mathbf{u}^{n-i})}{\Delta t} - \nabla \cdot (\nu \bar{\mathbf{L}}) &= - \sum_{i=0}^{s_p-1} (\gamma_i \nabla p^{n-i}) - \nabla \cdot (\mathbf{u}^n \otimes \mathbf{u}^n) + \mathbf{f}(t_{n+1}), \end{aligned} \quad (21)$$

where the first equation defines a new tensor-valued unknown  $\bar{\mathbf{L}}$  approximating the velocity gradient  $\nabla \bar{\mathbf{u}}$ , and the second is equation (7) written in terms of  $\bar{\mathbf{L}}$ . Taking the numerical flux definition  $(-\nu \hat{\bar{\mathbf{L}}}_h) \mathbf{n} \equiv (-\nu \bar{\mathbf{L}}_h) \mathbf{n} + \tau(\bar{\mathbf{u}}_h - \hat{\mathbf{u}}_h)$  and adding an equation to weakly enforce the continuity of its normal component on the space  $M_h^p$  [10, 11], we arrive at the following weak form

The velocity predictor  $\bar{\mathbf{u}}_h$

$$\begin{aligned} (\mathbf{G}, \bar{\mathbf{L}}_h)_{\mathcal{T}_h} + (\nabla \cdot \mathbf{G}, \bar{\mathbf{u}}_h)_{\mathcal{T}_h} - \langle \mathbf{G} \cdot \mathbf{n}, \hat{\mathbf{u}}_h \rangle_{\partial \mathcal{T}_h} &= 0 \\ \left( \mathbf{v}, \frac{\beta_s}{\Delta t} \bar{\mathbf{u}} \right)_{\mathcal{T}_h} - (\mathbf{v}, \nabla \cdot (\nu \bar{\mathbf{L}}_h))_{\mathcal{T}_h} + \langle \mathbf{v}, \tau(\bar{\mathbf{u}}_h - \hat{\mathbf{u}}_h) \rangle_{\partial \mathcal{T}_h} &= \left( \mathbf{v}, \sum_{i=0}^{s_u-1} \left( \frac{\beta_i}{\Delta t} \mathbf{u}^{n-i} \right) \right)_{\mathcal{T}_h} \\ &\quad - \sum_{i=0}^{s_p-1} (\gamma_i \text{pg}_h(\mathbf{v}, p_h^{n-i}, g_N^p(t_{n-i}))) + F_a(\mathbf{v}, \mathbf{u}_h) + (\mathbf{v}, \mathbf{f})_{\mathcal{T}_h} \\ \langle \boldsymbol{\mu}, (-\nu \bar{\mathbf{L}}_h) \mathbf{n} + \tau(\bar{\mathbf{u}}_h - \hat{\mathbf{u}}_h) \rangle_{\partial \mathcal{T}_h \setminus \Gamma_D} + \langle \boldsymbol{\mu}, \hat{\mathbf{b}}_h \rangle_{\Gamma_N} &= 0 \end{aligned} \quad (22)$$

This differs from the scheme in [13], which takes the jump of the pressure in the numerical flux.

This admits a matrix-discretization over each element:

$$\begin{bmatrix} A & B & -C \\ B^T & -D & E \\ -N & G & -H \end{bmatrix} \begin{bmatrix} L_h \\ U_h \\ \hat{U}_h \end{bmatrix} = \begin{bmatrix} 0 \\ -(F - P) \\ -L \end{bmatrix}. \quad (23)$$

In the case where the velocity components are not coupled on the boundary of the mesh due to the boundary conditions, for example, when a zero Dirichlet no-slip condition is applied to the velocity along a non-trivial bathymetry, and the vertical sides of the domain are axis-aligned, the linear system arising from equation (??) is decoupled, and the predictor velocity can be solved component-by-component.

Let  $\bar{\phi}$  represent any component of the velocity  $\bar{\mathbf{u}}$ . If we were to write the usual form of the HDG problem where  $\mathbf{q} = \nu \nabla \bar{\phi}$  represents the complete gradient of the component  $\bar{\phi}$ , then we want to find  $(\bar{\phi}, \mathbf{q}) \in W_h \times \mathbf{V}_h$  such that

$$\begin{aligned} (\mathbf{v}, \nu^{-1} \mathbf{q})_{\mathcal{T}_h} + (\nabla \cdot \mathbf{v}, \bar{\phi})_{\mathcal{T}_h} - \langle \mathbf{v} \cdot \mathbf{n}, \hat{\phi} \rangle_{\partial \mathcal{T}_h} &= 0 \\ \left( w, \frac{\bar{\phi}}{a \Delta t} \right)_{\mathcal{T}_h} + (w, \nabla \cdot (\mathbf{q}))_{\mathcal{T}_h} - \langle w, \tau(\bar{\phi} - \hat{\phi}) \rangle_{\partial \mathcal{T}_h} &= \text{rh}_h(w, \mathbf{u}^k, p_h^k, \mathbf{F})_K \\ \langle \boldsymbol{\mu}, \hat{\mathbf{q}} \cdot \mathbf{n} \rangle_{\partial \mathcal{T}_h} &= \langle \boldsymbol{\mu}, g_N \rangle_{\Gamma_N} \end{aligned} \quad (24)$$

where the operator  $\text{rh}_h(\cdot)$  contains all the component-wise right-hand side terms of the momentum equation. This formulation is consistent with the full 3D velocity predictor if  $\nu_{xy} = \nu_z$ . This also couples the domain together on the interfaces. The velocity predictor  $\bar{\mathbf{u}}_h^{k+1}$  is the vector  $(\bar{\phi}_u, \bar{\phi}_v, \bar{\phi}_w)$  resulting from each of the HDG solves.

In the decoupled case, the formulation of the DG right-hand side operators can be obtained by considering only the relevant component  $w$  of the test function  $\mathbf{v} \in \mathbf{V}_h^p$  as appears in equation (??). For the advective term, we have

$$a_h(w, \bar{\phi}, \mathbf{u}_h^k, \mathbf{g}_D) = -(w, \nabla \cdot (\bar{\phi} \mathbf{u}_h^k))_K = -(\nabla w, \bar{\phi} \mathbf{u}_h^k)_K + \langle w, (\bar{\phi} \mathbf{u}_h^k)^* \cdot \mathbf{n} \rangle_{\partial K}, \quad (25)$$

where the operator  $a_h(\cdot)$  can be treated explicitly by taking  $\bar{\phi} = u_{h,j}^k$  where the integer  $j$  denotes the spatial dimension corresponding to the component of the predictor velocity sought, or it can be treated

semi-implicitly (*cf.* [9]) as written; in the latter case, the weak form is no longer symmetric. In this paper, we consider a completely explicit treatment. [need to consider boundary treatment for the advective flux—compare table in Fehn with naive approach!](#) The numerical flux in this work  $(\bar{\phi} \mathbf{u}_h^k)^*$  is taken to be an upwind flux. Since  $\mathbf{u}_h^k$  is multiply-valued, either the average value  $\{\{\mathbf{u}_h\}\}$  or the HDG flux  $\hat{\mathbf{u}}_h$  can be used.

Similarly, the pressure gradient term for the component  $i$  of  $\bar{\mathbf{u}}_h^{k+1}$ ,  $\bar{\phi}_i$ , is

$$\text{pg}_h(w, p_h^k) = - \left( \frac{\partial w}{\partial x_i}, p_h^{',k} \right) + \left\langle w, \left( p_h^{',k} \right)^* n_i \right\rangle \quad (26)$$

where  $n_i$  denotes the  $i^{\text{th}}$  component of the outward unit normal  $\mathbf{n}$ , and where the numerical flux is again chosen to be the central flux,  $\left( p_h^{',k} \right)^* = \left\{ \left\{ p_h^{',k} \right\} \right\}$ .

### 3.3.3. Pressure corrector

The weak form for the pressure corrector can be expressed as

$$\begin{aligned} (\mathbf{v}, \mathbf{q}_{\delta p}^{k+1})_{\mathcal{T}_h} + (\nabla \cdot \mathbf{v}, \delta p^{k+1})_{\mathcal{T}_h} - \langle \mathbf{v} \cdot \mathbf{n}, \hat{\delta p} \rangle_{\partial \mathcal{T}_h} &= 0 \\ -(w, \nabla \cdot \mathbf{q}_{\delta p}^{k+1})_{\mathcal{T}_h} + \langle w, \tau_p \delta p^{k+1} \rangle_{\partial \mathcal{T}_h} - \langle w, \tau_p \hat{\delta p} \rangle_{\partial \mathcal{T}_h} &= -\frac{\beta_s}{\Delta t} \text{vd}_h(w, \bar{\mathbf{u}}_h, \mathbf{g}_D^u(t_{n+1})) \\ \left\langle \mu, \mathbf{q}_{\delta p}^{k+1} \cdot \mathbf{n} + \tau_p (\delta p^{k+1} - \hat{\delta p}) \right\rangle_{\partial \mathcal{T}_h \setminus \Gamma_D} &= 0 \end{aligned} \quad (27)$$

### 3.4. Inhomogeneous boundary condition treatment

### 3.5. Numerical flux definitions

## 4. Implementation

The finite element forward models were implemented in C++ and make use of the finite element library `deal.ii` [? ].

All surface and volume integral operators are discretized with Gaussian quadrature using  $p_{\text{order}} + 1$  one-dimensional (1D) quadrature points in each spatial direction, where  $p_{\text{order}}$  is the polynomial order of the finite element space (space). The experiments in this work are run on a compute blade containing two 32-core AMD EPYC 7543 processors running at 2.80 GHz.

Convergence results are reported using the relative  $L^2$ -errors

$$\mathbf{e}_{\mathbf{u}_h}(t) = \frac{\|\mathbf{u}(\mathbf{x}, t) - \mathbf{u}_h(\mathbf{x}, t)\|_{L^2(\Omega_h)}}{\|\mathbf{u}(\mathbf{x}, t)\|_{L^2(\Omega_h)}}, \quad \mathbf{e}_{p_h}(t) = \frac{\|p(\mathbf{x}, t) - p_h(\mathbf{x}, t)\|_{L^2(\Omega_h)}}{\|p(\mathbf{x}, t)\|_{L^2(\Omega_h)}}, \quad (28)$$

for vector- and scalar-valued quantities, respectively. Each of the  $L^2$ -norms over the computational domain  $\Omega_h$  is calculated using Gaussian quadrature over the element volumes using  $p_{\mathbf{u}} + 3$  for errors in the velocity and  $p_p + 3$  for errors in the pressure variables. Unless explicitly specified, the abbreviations  $\mathbf{e}_{\mathbf{u}_h}$  and  $\mathbf{e}_{p_h}$  refer to the velocity and pressure errors at the final time  $T$ .

Similarly, unless otherwise specified, all iterative solver tolerances are specified to a relative error of  $10^{-12}$  and an absolute error of

The iterative solvers used in the subsequent test cases use a relative solver tolerance of  $10^{-8}$  and an absolute solver tolerance of  $10^{-12}$  as tolerance criteria for all system of equations to be solved, unless otherwise specified.

### 4.1. Computational considerations

HDG methods offer several well-known advantages over standard DG-FEM methods. In addition to favorable convergence properties, hybridization of the system by introducing the space `Mph` can lead to a substantial reduction in globally-coupled degrees of freedom as compared to classical DG methods. However, the complexities associated with the static condensation procedure introduce a set of non-typical computational performance considerations as compared to other high-order methods. In this section, we provide a brief discussion of practical computational considerations specific to HDG schemes in the context of parallelization and matrix-free solvers. These topics are often overlooked or unaddressed in the literature (to DG/HDG etc ref).

#### 4.2. Parallelization

A commonly cited performance advantage of HDG is the embarrassingly parallel nature of the assembly as well as the local reconstruction of the numerical solution QH UH from the trace quantity UHAT. Means that people brush this step off computationally, as a parallel implementation should make it trivial. However there's more to the story than that.

- argument: that assembly and reconstruction start to matter at large problem sizes
- parallelization needed to beat down assembly and reconstruction time
- multi-threaded parallel applications highly dependent on polynomial order

though it may seem counter-intuitive, multi-threaded programs can sometimes run slower than their single-threaded or serial counterparts due to several reasons:

Overhead: Threads require resources for creation, termination, and synchronization. The overhead of managing multiple threads can slow down the program, especially if the computation performed by each thread is not significant compared to this overhead. Contention: When multiple threads try to access a shared resource simultaneously, contention can occur. This could lead to a situation called a lock, where one thread has to wait for another to release a resource. In extreme cases, this can lead to a slowdown known as lock contention, where the threads spend more time waiting to access the resource than doing useful work. False sharing: Even if different threads are working on different data, if that data is close enough in memory (i.e., in the same cache line), then the hardware can treat it as if it were shared. When one thread updates its data, the hardware will think the entire cache line has been modified and will have to update it across all the cores, leading to a significant slowdown. This is known as false sharing. Load imbalance: If the work is not evenly distributed across threads, some threads may finish their work early and sit idle, while others are still working. This is known as load imbalance. It can result in the execution time of the program being dominated by the slowest thread. Non-parallelizable tasks: Some tasks simply can't be parallelized effectively due to their nature, they are inherently serial (or "sequential"). This principle is encapsulated by Amdahl's Law, which states that the maximum improvement to a system's performance is limited by the portion of the system that can't be parallelized. Memory limits: Every thread created uses some memory for its stack. If the number of threads is too high, it can cause significant memory usage, potentially leading to swapping if the system runs out of physical memory, which can drastically decrease performance. It's worth noting that developing an efficient multi-threaded program requires careful design to minimize these issues. This can involve techniques such as fine-tuning the number of threads, avoiding contention and false sharing, balancing the load properly, and choosing appropriate data structures and synchronization primitives. But even with all these considerations, there's no guarantee that a multi-threaded solution will always outperform a well-optimized single-threaded one. It highly depends on the nature of the problem and the specific hardware the software is running on.

In this case the synchronization work necessary to ensure no race condition while copying elemental contributions into the global linear system is highly sensitive to problem dimension and polynomial order. The necessary work on the cell

In the following, we discretize the problem at different polynomial orders such that the global number of primal degrees of freedom are held constant. We evaluate the assembly times using both serial and parallelized assembly.

- need to give the computer enough work (i.e., large enough problem size)

It turns out that for one-dimensional problems using any polynomial order less than or equal to  $p = 10$ , it is far more efficient to perform assembly in serial as opposed to in parallel using multi-threading. This is because the problem dimension leads to element-local systems small enough that the synchronization work between the threads to prevent a race condition copying into the global linear system is much larger than the actual work on the cell.

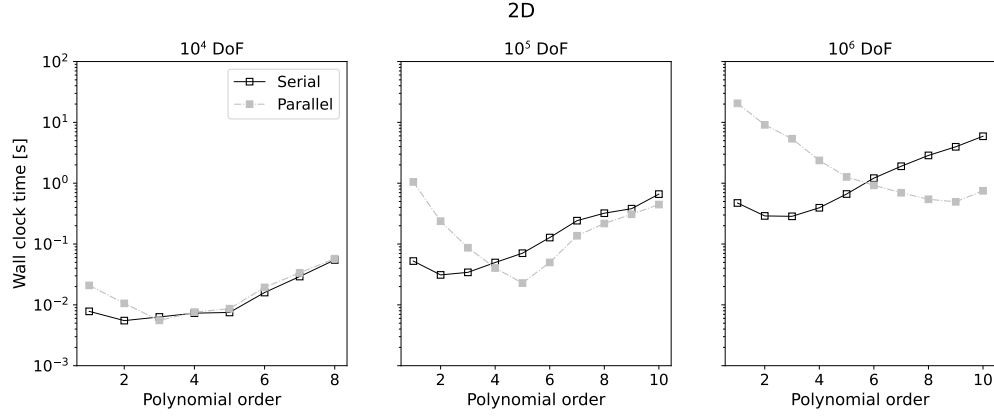


Figure 1: Wall clock times for 2D HDG assembly at constant problem size, serial vs. parallel

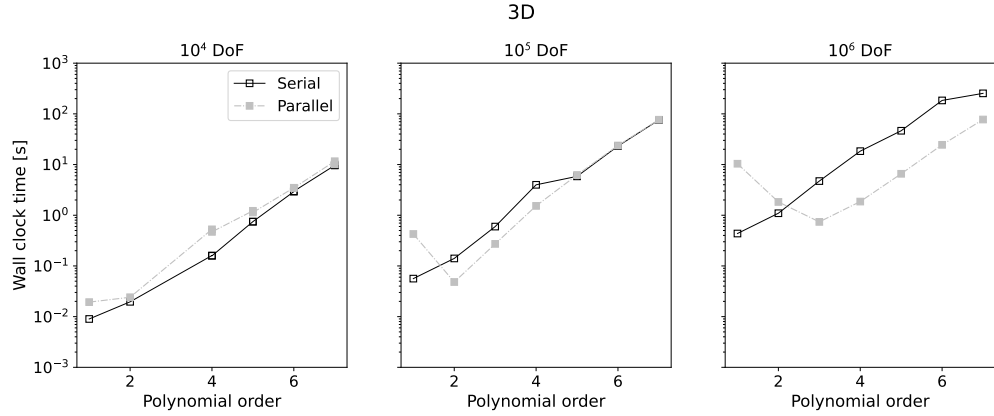


Figure 2: Wall clock times for 3D HDG assembly at constant problem sizes, serial vs. parallel

#### 4.3. Matrix-free solvers

#### 4.4. On the solution of the pure Neumann problem

In the case of pure Dirichlet boundaries on the velocity predictor, the pressure level is undefined. To see this, note that if the scalar field  $\delta p_h$  is a solution of (9) with  $\Gamma_N = \partial\Omega$ , then  $\delta p_h + c$  for any  $c \in \mathbf{R}$  is also a solution. Discretely, the null space of the linear system arising due to the discretization of (9) is spanned by the set of constant vectors, implying a singular linear system.

Although crucial to the performance of pressure correction schemes, the issue of finding a solution to the rank-one deficient singular system in such cases receives little attention in the literature, outside of specialized literature for iterative solvers [1, 7].

In [2], the authors rigorously describe strategies for addressing the singular system in a continuous finite element context. Here, we extend that discussion to the DG-FEM setting and illustrate computational trade-offs associated with different candidate approaches.

This is complicated by the fact that some numerical linear algebra software varies widely in terms of implementation; while some libraries will not solve a singular system (numpy), other direct procedures will do so when they can identify a zero-pivot. Similar for iterative solvers.

#### section TD

- lit review
- three approaches
- manufactured solution

##### 4.4.1. Proposed solutions

**Subspace projection.** As a first approach, we apply a subspace projection using a Krylov-solver [14], making use of the fact that iterative solvers solve singular systems provided that the right-hand side is in the orthogonal complement of the null-space. This modification is ok because we're only removing the null space contribution to the linear system

**Penalty method.** As a second approach, we apply a penalty-based method which can be interpreted as a regularization. This method, while simple and independent of linear solver type, involves the specification of a hyperparameter  $\gamma$ . As a third approach, we impose a mean-value constraint

$$\int_{\Omega} \delta p \, d\Omega = 0 \quad (29)$$

into the linear system directly, avoiding the saddle point system that arises as a result of applying the constraint as a Lagrange multiplier [2].

**Point constraint.** An approach favored by many practitioners is to manually specify the value of the candidate solution at a single point by removing an equation from the discrete system and applying a Dirichlet constraint fixing the value of the solution at that point to an arbitrary constant, eliminating the null space and allowing solution of the linear system using a conventional direct solver. However, in a finite-element context, the function  $\delta_p$  is often represented in the Sobolev space  $H^1(\Omega)$  or  $L^2(\Omega)$ , spaces in which point evaluations do not make sense. In these cases, imposing such a constraint can render the variational problem ill-posed. A typical manner of fixing this is to specify the value of the solution along a measurable subset of the domain; typically an element face; while this constraint is well-posed, it's also undesirable, because specifying that the value of the numerical solution along the face will change the character of the numerical solution and thus the physics.

##### 4.4.2. Computational benchmarks

In order to investigate the performance of each proposed method to solve the pure Neumann problem, we assess the convergence, accuracy, and efficiency of each method over a wide range of problem sizes and polynomial degrees  $p$ . We consider the analytical solution

$$\delta_p^* = \frac{1}{3} \sin\left(-\frac{\pi}{2}x\right) \cos(2\pi y), \quad (30)$$



(Figure 3, left) with boundary conditions and forcing function deduced from the exact solution over the computational domain  $\Omega = [-1, 1]^2$ . By symmetry, we have that the exact solution  $\delta_p^*$  is analytically zero-mean in the sense of equation (29). Additionally, the value of the boundary condition  $g_N = \nabla \delta_p^* \cdot \mathbf{n}$  vanishes algebraically over the boundary  $\partial\Omega$ , imitating the boundary conditions imposed in the pressure corrector step. We consider a uniform Cartesian grid consisting of quadrilateral elements of length  $L/2^\ell$  in each spatial

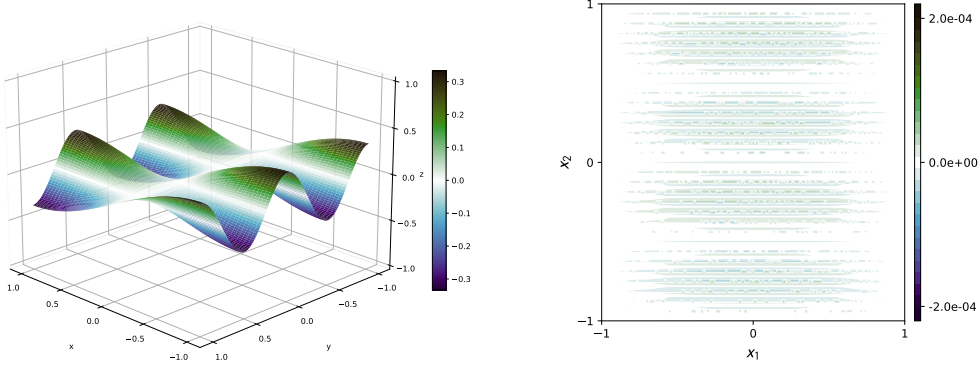


Figure 3: (Left) Analytical solution of the pure Neumann problem. (Right) Distribution of numerical errors in the gradient  $\nabla \delta_p^* - \mathbf{q}_{\delta_{p,h}}$  over the computational domain for a mesh of  $32 \times 32$  elements at polynomial order  $p = 3$ .

dimension, where the integer  $\ell = 0, 1, \dots$  denotes the level of refinement. To assess the convergence of the primal unknown  $\delta_p$ , the integral mean level of the numerical solution is subtracted as post-processing to agree with the analytical solution. All iterative solves are performed using conjugate gradient (CG) iterations to solve for the trace unknowns  $\hat{\delta}_{p,h}$ , from which the primal solution  $\delta_{p,h}$  and its gradient  $\mathbf{q}_{\delta_{p,h}}$  are reconstructed.

All three methods achieve optimal  $p + 1$  order convergence to the exact solution in both the primal  $\delta_{p,h}$  and gradient  $\mathbf{q}_{\delta_{p,h}}$  unknowns, respectively, before the iterative solver tolerance and finite-precision effects begin to pollute the convergence order. Figure 4 illustrates the convergence history of the primal unknown  $\delta_{p,h}$  for the projection approach, both from the perspective of mesh resolution and in globally-coupled trace unknowns. The errors and orders of convergence in the  $L^2$ -norm are presented in tabular form in Appendix A.1. In what follows, we demonstrate the subspace projection approach to be the computationally superior method; therefore, we present the convergence results for the projection method and omit the others for brevity.

An examination of the spatial distribution of the errors in the numerical gradient<sup>1</sup> over the computational domain (Figure 3, right) illustrates that they are neither disproportionately large on the domain boundary, nor are they localized around a single point, problems that have been known to plague the penalization and point constraint methods, which depend on a specific choice of degree of freedom for their implementation [2, 5].

Given the comparable order of convergence to the analytical solution for all three approaches, we compare their relative computational advantages. Reporting a condition number for the resultant linear system solved with each method is not feasible. On one hand, the problem sizes reported in this work are large enough to make direct computation of the eigenvalue spectrum intractable. Moreover, as the subspace projection transformation is applied at each conjugate gradient iteration, there is not a single resultant linear system for which to compute a condition number. Therefore, as a practical indication of the conditioning of each algorithm, we examine the efficiency of the iterative solver. Figure 5a shows the number of conjugate gradient iterations required to achieve convergence to the specified solver tolerance (without preconditioning) over a wide range of problem sizes. We observe that the point constraint and penalization methods require many more iterations to converge than the subspace projection approach across polynomial orders, typically between a factor of two and a factor of ten. Since the subspace projection requires removal of the mean at

<sup>1</sup>We compute the directional error  $\nabla \delta_p^* - \mathbf{q}_{\delta_{p,h}}$  over each element and plot the  $x_1$ -component. The results are similar for the gradient in the  $x_2$  direction. We consider the gradient rather than the primal variable, because the gradient is the quantity that is used in the projection method detailed in §2.

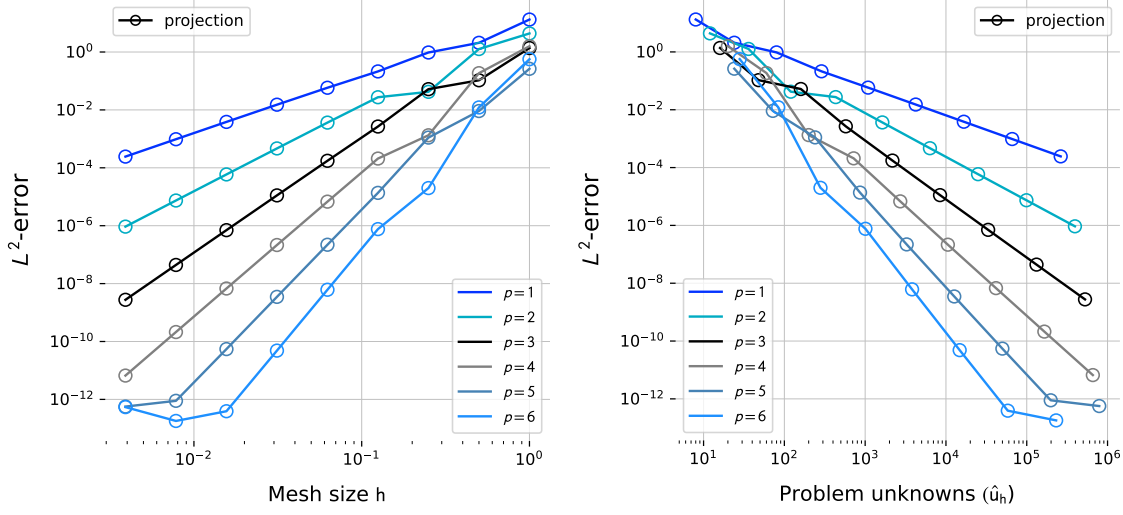


Figure 4: Errors in the primal unknown  $\|\delta_p^* - \delta_{p,h}\|_{L^2(\Omega)}$  for the singular Neumann problem (2D) treated with subspace projection. Convergence is shown with respect to mesh size as well as number of globally-coupled unknowns. Optimal convergence is achieved for all polynomial orders. Convergence history data is tabulated in [Appendix A.1](#)

every conjugate gradient iteration and the other two methods do not, we also benchmark the performance of each algorithm in terms of the overall  $L^2$ -error in the primal solution variable as a function of the wall clock time to solution of the linear system for the trace unknown  $\delta_{p,h}$  in Figure 5b. We observe that the subspace projection approach equals or outperforms the other two methods over a wide range of problem sizes, with the advantage increasing as a function of polynomial order.

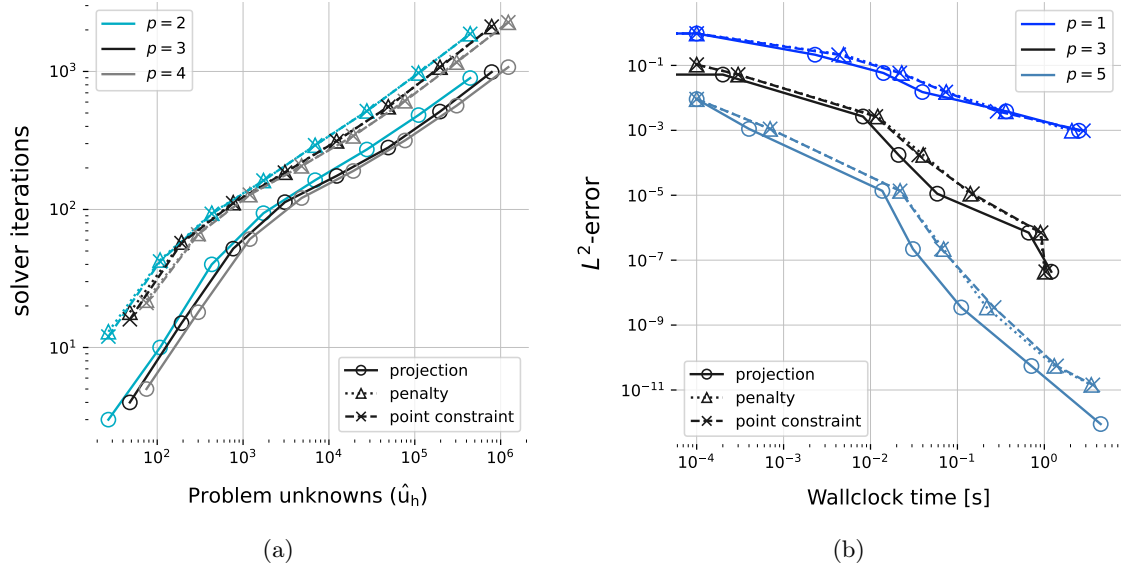


Figure 5: Performance comparisons of the different singularity treatments. (Left) Number of conjugate gradient iterations required for convergence for each method at different problem sizes and polynomial orders. (Right)  $L^2$ -error achieved in the primal unknown  $\delta_{p,h}$  as a function of wall clock time-to-solution of the global linear system for the trace unknown, shown at different problem sizes spanning from  $10^1$  to  $10^6$  trace degrees of freedom, and different polynomial orders.

In conclusion, we have found the subspace projection approach to be a stable, robust, and accurate method for numerically solving the singular system arising from the discretization of the pure Neumann problem occurring in the case of pure Dirichlet boundary conditions on the velocity. Furthermore, we have demonstrated it to be a superior method computationally, delivering better accuracy with a shorter time-to-solution than both the point-constraint and penalization approaches, and without the need to calibrate any additional hyperparameters or to single out a particular problem degree of freedom for the imposition of a

constraint.

## 5. Numerical experiments

### 5.1. Verification

#### 5.1.1. Kovasnay flow

Following [6], to provide verification of the nonlinear, pressure, and viscous spatial derivatives independent of the temporal discretization, we test the method using the Kovasnay analytical solution to the Navier–Stokes equations.

$$\mathbf{u}(\mathbf{x}) = \begin{pmatrix} 1 - e^{\lambda x} \cos(2\pi y) \\ \frac{\lambda}{2\pi} e^{\lambda x} \sin(2\pi y) \end{pmatrix}, \quad p(\mathbf{x}) = \frac{1}{2} (1 - e^{2\lambda x}), \quad (31)$$

where the parameter

$$\lambda = \frac{1}{2\nu} - \sqrt{\frac{1}{4\nu^2} + 4\pi^2}. \quad (32)$$

We consider a domain  $\Omega = [-0.5, 1] \times [-0.5, 1.5]$  with the domain boundary consisting of inflow and outflow segments  $\partial\Omega = \Gamma_{\text{in}} \cup \Gamma_{\text{out}}$  with  $\Gamma_{\text{out}} = \{\mathbf{x} \mid x_1 = 1\}$ . For the velocity field, Dirichlet boundary conditions on the velocity field at the inflow boundary  $\Gamma_{\text{in}}$  and Neumann boundary conditions at the outflow boundary  $\Gamma_{\text{out}}$  are deduced from the exact solution. Similarly, the exact solution provides the directional derivative normal to the boundary for the Neumann pressure boundary condition at the inflow boundaries  $\Gamma_{\text{in}}$  and the Dirichlet boundary conditions on the pressure at the outflow. These boundary conditions are summarized in Table 1.

	$\Gamma_{\text{in}} = \partial\Omega \setminus \Gamma_{\text{out}}$	$\Gamma_{\text{out}} = \partial\Omega \cap \{\mathbf{x} : x_1 = 1\}$
$\mathbf{u}$	$\mathbf{g}_D = \mathbf{u}(\mathbf{x})$	$\mathbf{g}_N = \nabla \mathbf{u} \cdot \mathbf{n}$
$p$	$g_N = \nabla p \cdot \mathbf{n}$	$g_D = p(\mathbf{x})$

Table 1: Summary of boundary conditions §5.1.1.

The simulation is performed for one time integration step with viscosity parameter  $\nu = 1/40$ .

### 5.2. Vortex flow

Spatial convergence test, temporal convergence test

### 5.3. Unsteady Stokes Flow

The review paper [5] solves the unsteady Stokes problem

$$\begin{aligned} \mathbf{u}(\mathbf{x}, t) &= \pi \sin t \begin{pmatrix} \sin(2\pi y) \sin^2(\pi x) \\ -\sin(2\pi x) \sin^2(\pi y) \end{pmatrix} \\ p(\mathbf{x}, t) &= \sin(t) \cos(\pi x) \sin(\pi y) \end{aligned} \quad (33)$$

where the source term  $f = u_t - \nu \nabla \cdot (\nabla \mathbf{u}) + \nabla p$ .

In [3], we have

$$\begin{aligned} \mathbf{u}(\mathbf{x}, t) &= \begin{pmatrix} \sin(x)(a \sin(ay) - \cos(a) \sinh(y)) \\ \cos(x)(\cos(ay) + \cos(a) \cosh(y)) \end{pmatrix} \exp(-\lambda t), \\ p(\mathbf{x}, t) &= \lambda \cos(a) \cos(x) \sinh(y) \exp(-\lambda t) \end{aligned} \quad (34)$$

where  $\lambda = \nu(1 + a^2)$   $\nu = 1$  and  $a = 2.883356$  on a domain of  $\Omega = [-1, 1]^2$ ; they take  $[0, T] = [0, 0.1]$  and Dirichlet boundary conditions everywhere,  $\Gamma = \Gamma_D$ .

### 5.4. Taylor-Green vortex flow

Is this the same test case as in Hesthaven?

5.5. *Pipe flow*

5.6. *Backward facing step*

5.7. *Lid-driven cavity flow*

5.8. *Lock Exchange?*

5.9. *Lab RTI flow?*

## 6. Discussion

The imposition of open boundary conditions where inflow and outflow can co-exist remains an open problem in the finite element community [12].

## 7. Appendix: BDF1

## Appendix A. Pure Neumann problem

### Appendix A.1. Convergence history for the pure Neumann problem

We present the error and convergence history in  $L^2$ -norm corresponding to the plots in Figure 4 in Table A.2 for the numerical solution of the pure Neumann problem using the method of subspace projection for polynomial orders  $p = 1, \dots, 6$ . All results are reported in the asymptotic regime, before errors associated with floating point precision begin to pollute the convergence order. Optimal convergence of  $p + 1$  for both  $u_h$  and  $\mathbf{q}_h$  were achieved in all test cases for all dimensions.

Degree $p$	Mesh $N_K$	$\ u - u_h\ _{L^2(\Omega)}$ error	order	$\ \mathbf{q} - \mathbf{q}_h\ _{L^2(\Omega)}$ error	order
1	1	1.323e+01	-	1.180e+01	-
	4	2.077e+00	2.67	3.348e+00	1.82
	16	9.620e-01	1.11	1.785e+00	0.91
	64	2.134e-01	2.17	3.911e-01	2.19
	256	5.828e-02	1.87	1.044e-01	1.91
	1024	1.514e-02	1.94	2.685e-02	1.96
	4096	3.851e-03	1.97	6.801e-03	1.98
	16384	9.709e-04	1.99	1.711e-03	1.99
	65536	2.437e-04	1.99	4.291e-04	2.00
2	1	4.377e+00	1.77	7.488e+00	1.76
	4	1.268e+00	1.79	2.488e+00	1.59
	16	4.212e-02	4.91	1.270e-01	4.29
	64	2.740e-02	0.62	5.035e-02	1.33
	256	3.642e-03	2.91	6.592e-03	2.93
	1024	4.668e-04	2.96	8.395e-04	2.97
	4096	5.901e-05	2.98	1.058e-04	2.99
	16384	7.415e-06	2.99	1.327e-05	2.99
	65536	9.292e-07	3.00	1.662e-06	3.00
3	1	1.370e+00	2.56	2.964e+00	2.60
	4	1.056e-01	3.70	3.524e-01	3.07
	16	5.215e-02	1.02	9.719e-02	1.86
	64	2.667e-03	4.29	4.912e-03	4.31
	256	1.748e-04	3.93	3.185e-04	3.95
	1024	1.114e-05	3.97	2.019e-05	3.98
	4096	7.019e-07	3.99	1.269e-06	3.99
	16384	4.404e-08	3.99	7.955e-08	4.00
4	1	1.584e+00	3.64	3.446e+00	3.67
	4	1.858e-01	3.09	3.587e-01	3.26
	16	1.330e-03	7.13	3.375e-03	6.73
	64	2.086e-04	2.67	3.848e-04	3.13
	256	6.778e-06	4.94	1.240e-05	4.96
	1024	2.151e-07	4.98	3.919e-07	4.98
	4096	6.767e-09	4.99	1.230e-08	4.99
	16384	2.121e-10	5.00	3.853e-10	5.00
5	1	2.630e-01	4.40	8.666e-01	4.51
	4	9.138e-03	4.85	2.661e-02	5.03
	16	1.106e-03	3.05	2.060e-03	3.69
	64	1.362e-05	6.34	2.516e-05	6.36
	256	2.200e-07	5.95	4.034e-07	5.96
	1024	3.482e-09	5.98	6.364e-09	5.99
	4096	5.470e-11	5.99	9.984e-11	5.99
6	1	5.639e-01	4.98	1.158e+00	4.84
	4	1.217e-02	5.53	2.323e-02	5.64
	16	1.995e-05	9.25	4.649e-05	8.96
	64	7.628e-07	4.71	1.411e-06	5.04
	256	6.134e-09	6.96	1.127e-08	6.97
	1024	4.846e-11	6.98	8.880e-11	6.99

Table A.2: Convergence history for the singular pure Neumann problem using subspace projection (2D).

## References

- [1] O. Axelsson. *Iterative Solution Methods*. Cambridge University Press, Mar. 1996. Google-Books-ID: hNpJg\_pUsOwC.

- [2] P. Bochev and R. B. Lehoucq. On the Finite Element Solution of the Pure Neumann Problem. *SIAM Review*, 47(1):50–66, Jan. 2005. Publisher: Society for Industrial and Applied Mathematics.
- [3] N. Fehn, W. A. Wall, and M. Kronbichler. On the stability of projection methods for the incompressible Navier–Stokes equations based on high-order discontinuous Galerkin discretizations. *Journal of Computational Physics*, 351:392–421, Dec. 2017.
- [4] N. Fehn, W. A. Wall, and M. Kronbichler. Robust and efficient discontinuous Galerkin methods for under-resolved turbulent incompressible flows. *Journal of Computational Physics*, 372:667–693, Nov. 2018.
- [5] J. L. Guermond, P. Mineev, and J. Shen. An overview of projection methods for incompressible flows. *Computer Methods in Applied Mechanics and Engineering*, 195(44):6011–6045, Sept. 2006.
- [6] J. S. Hesthaven and T. Warburton. *Nodal Discontinuous Galerkin Methods*, volume 54 of *Texts in Applied Mathematics*. Springer New York, New York, NY, 2008.
- [7] R. Iankov, M. Datcheva, S. Cherneva, and D. Stoychev. *Finite Element Simulation of Nanoindentation Process*. Jan. 2013.
- [8] G. E. Karniadakis, M. Israeli, and S. A. Orszag. High-order splitting methods for the incompressible Navier-Stokes equations. *Journal of Computational Physics*, 97(2):414–443, Dec. 1991.
- [9] N. C. Nguyen, J. Peraire, and B. Cockburn. An implicit high-order hybridizable discontinuous Galerkin method for linear convection–diffusion equations. *Journal of Computational Physics*, 228(9):3232–3254, May 2009.
- [10] N. C. Nguyen, J. Peraire, and B. Cockburn. A hybridizable discontinuous Galerkin method for Stokes flow. *Computer Methods in Applied Mechanics and Engineering*, 199(9):582–597, Jan. 2010.
- [11] N. C. Nguyen, J. Peraire, and B. Cockburn. An implicit high-order hybridizable discontinuous Galerkin method for the incompressible Navier–Stokes equations. *Journal of Computational Physics*, 230(4):1147–1170, Feb. 2011.
- [12] R. L. Sani and P. M. Gresho. Résumé and remarks on the open boundary condition minisymposium. *International Journal for Numerical Methods in Fluids*, 18(10):983–1008, 1994. \_eprint: <https://onlinelibrary.wiley.com/doi/pdf/10.1002/flid.1650181006>.
- [13] M. P. Ueckermann and P. F. J. Lermusiaux. Hybridizable discontinuous Galerkin projection methods for Navier–Stokes and Boussinesq equations. *Journal of Computational Physics*, 306:390–421, 2016.
- [14] H. A. v. d. Vorst. *Iterative Krylov Methods for Large Linear Systems*. Cambridge University Press, Apr. 2003. Google-Books-ID: wE0NrHkrqRAC.