Comparison of high-order continuous and hybridizable discontinuous Galerkin methods for incompressible fluid flow problems

Mahendra Paipuri[a,b], Sonia Fernández–Méndez[b], Carlos Tiago[a,*]

[a] *CERIS, ICIST, Instituto Superior Técnico, Universidade de Lisboa, Av. Rovisco Pais, 1049-001 Lisboa, Portugal*
[b] *Laboratori de Càlcul Numèric (LaCàN), Universitat Politècnica de Catalunya (UPC-BarcelonaTech), 08034 Barcelona, Spain*

Abstract

The computational efficiency and the stability of Continuous Galerkin (CG) methods, with Taylor–Hood approximations, and Hybridizable Discontinuous Galerkin (HDG) methods are compared for the solution of the incompressible Stokes and Navier–Stokes equations at low Reynolds numbers using direct solvers. A thorough comparison in terms of CPU time and accuracy for both discretization methods is made, under the same platform, for steady state problems, with triangular and quadrilateral elements of degree $k = 2 - 9$. Various results are presented such as error *vs.* CPU time of the direct solver, error *vs.* ratio of CPU times of HDG to CG, *etc.* CG can outperform HDG when the CPU time, for a given degree and mesh, is considered. However, for high degree of approximation, HDG is computationally more efficient than CG, for a given level of accuracy, as HDG produces lesser error than CG for a given mesh and degree. Finally, stability of HDG and CG is studied using a manufactured solution that produces a sharp boundary layer, confirming that HDG provides smooth converged solutions for Reynolds numbers higher than CG, in the presence of sharp fronts.

*Corresponding author
Email addresses: mahendra.paipuri@tecnico.ulisboa.pt (Mahendra Paipuri), sonia.fernandez@upc.edu (Sonia Fernández–Méndez), carlos.tiago@tecnico.ulisboa.pt (Carlos Tiago)

February 27, 2018

Keywords: efficiency, Hybridizable Discontinuous Galerkin, Continuous Galerkin; stability, Navier–Stokes, Taylor–Hood finite elements
2010 MSC: 00-01, 99-00

1. Introduction

Incompressible fluid flow problems are encountered in everyday life and have utmost practical importance. The applications range from meteorology and weather forecasting, to aerodynamics and magneto-hydrodynamics. Over the past decades, there have been a lot of developments in Computational Fluid Dynamics (CFD) to obtain the numerical solutions of a wide variety of problems. Finite volume and Finite element methods are often used in this context. Over the years, Continuous Galerkin (CG) finite elements [1] underwent a lot of development and their usage for incompressible Navier–Stokes equations was greatly improved. The major drawback of CG methods is the stability issues they pose in convection dominated problems.

Discontinuous Galerkin (DG) finite elements are a class of finite elements, which can offer both local conservation and stability properties with appropriate weak formulations. The original DG method was introduced to solve linear hyperbolic equations [2], and later developed for the solution of the Navier–Stokes equations [3]. Ever since, many DG methods have been proposed for discretizing second-order self-adjoint operators. Some of them are Interior Penalty Methods (IPM) [4], Local Discontinuous Galerkin (LDG) methods [5] and Compact Discontinuous Galerkin (CDG) methods [6]. In spite of being more stable and the ability to handle adaptive algorithms with little implementation overhead, DG methods are often criticized for having a higher number of Degrees of Freedom (DOFs) compared to the CG counterpart. The linear systems arising from DG methods are considerably larger and less sparse. Moreover, at high order approximation, static condensation improves the computational efficiency of CG methods. The applicability of a technique similar to static condensation to those DG methods is crippled due to the coupling of interior nodes of an element with

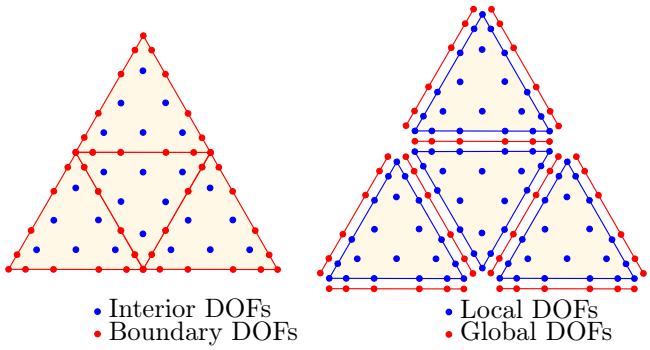


Figure 1: Comparison of velocity DOFs between CG and HDG.

the neighbouring element nodes.

With the introduction of a numerical technique called hybridization [7], the shortcomings of the DG methods were addressed. The Hybridizable Discontinuous Galerkin (HDG) method was proposed in [8] in the framework of second order elliptic problems. Thereafter, it was extended to other physical phenomenon [9, 10, 11, 12, 13, 14, 15, 16]. The hybridization in HDG allows the use of a technique similar to static condensation in CG, both of them leading to a significant reduction in the number of DOFs in the final system for high-order computations.

Figure 1 highlights the fundamental difference between the velocity DOFs of CG and HDG methods: the blue nodes correspond to interior nodes in CG and to local elemental DOFs in HDG, whereas the red nodes represent boundary DOFs in CG, and the so-called global DOFs in HDG. For the pressure field, the CG approximation has again interior and boundary DOFs, with the same structure as figure 1 but with a lower degree, whereas in HDG global DOFs correspond to a single scalar DOF per element. The hybridization in HDG (static condensation in CG) enables to express local (interior) DOFs in terms of global (boundary) DOFs, leaving only the last ones to be solved. Hence, in the final condensed system of equations, HDG has more DOFs than CG for velocity, due to the duplication of vertexes in 2D (edge nodes in 3D), but it has less DOFs for pressure. Compared to other DG methods such as IPM or CDG, which do not

allow static condensation, HDG has much less DOFs. Reference [12] highlights four distinctive features of HDG methods over other DG methods namely, (i) reduced DOF count, (ii) optimal convergence, (iii) superconvergence and (iv) unified treatment of boundary conditions. The reduction of the number of DOFs is due to the hybridization, as already discussed. Second, for fluid flow problems, HDG method provides an approximate velocity, pressure and gradient of velocity converging with optimal order of $k+1$ in $\mathcal{L}_2$-norm for a smooth solution, where k is the polynomial order of approximation used to represent the components of the solution. Since, HDG shows optimal convergence for gradient of velocity, in addition to the property that the mean of the velocity inside each element converges with order $k+2$, it is possible to do an element-by-element post-processing, to obtain a new approximation of velocity which converges with an order of $k+2$. The computational overhead for post-processing is very small as it is done at the elemental level. However, this property of superconvergence can only be noticed in the diffusion regime. In convection-dominated problems, superconvergence cannot be guaranteed, even though numerical experiments show that the post-processed solution provides improved accuracy in most cases.

Other DG methods have also been recently developed, like Multi-scale Discontinuous Galerkin (MDG) method [17] and Embedded Discontinuous Galerkin (EDG) method [18], aiming to reduce the number of DOFs in a DG discretization. Unlike HDG method, neither MDG nor EDG method has superconvergence properties.

In spite of HDG having multiple desirable properties, its performance compared to CG and other DG methods is still under study. A comparison for steady state convection–diffusion equation in diffusive regime can be found at [19]. Reference [20] presents a first comparison study between CG and HDG for a two-dimensional elliptic problem. In that work, CPU times for solving the linear system are compared for polynomials of degree up to fourteen, for triangular and quadrilateral elements. Later, [21] extended the work to 3D and studied the performance with direct and iterative solvers. A comparative study between space-time DG and space-time HDG methods for incompressible Navier–Stokes

problem is presented in [22]. A scalability study of HDG in compressible flows is made in [23], while a theoretical floating point operations (FLOPS) count for CG, CDG and HDG for second order elliptic problem is given in [24]. Reference [25] compares CG and HDG for linear elasticity problems, and concludes that HDG of degree k is as efficient as CG of degree $k+1$. A comparison between CG, CDG and HDG for wave problems can be found at [26]. A target based adaptive formulation is compared between hybridised and standard DG methods in [27]. A comparison was made between HDG and DG methods on DOFs count and number of non-zeros in the linear system arising from Poisson problem in [28].

The present work focuses on the computational performance between CG, with Taylor-Hood elements, and HDG for Stokes and Navier–Stokes incompressible flows. Previous work of [21] concluded that the effective pre-conditioning strategies has to be developed for HDG discretization to have a competitive iterative solver performance compared to CG. Hence, only direct solvers are considered in the present work. The comparison study of the CPU times is carried out at low Reynolds regimes, to avoid the need of stabilization techniques, specially in the case of CG, that may affect the convergence and accuracy. The study made in [29] using different formulations of HDG for Stokes concluded that velocity-pressure-gradient formulation provides the best approximation for the same computational complexity and hence, the same is used in the present work. The CG and HDG formulations considered are described in section 3. Some details of the implementation, and a theoretical count of the number of DOFs in 2D and 3D, are presented in sections 4 and 5, respectively. 2D numerical tests are used to assess the computational efficiency of HDG and CG, in terms of accuracy and CPU time for the solution of the linear system, in section 6. The variation of the condition number of the matrix with the degree of approximation is also studied. Finally, in section 7, the stability of CG and HDG for the solution of the Navier-Stokes equations at high Reynolds number, in the presence of sharp fronts, is compared with a manufactured numerical test. Note that in this section, only robustness of the discretizations are compared and not the CPU times. The aim of the stability study is to compare the regimes

where stabilised formulations are necessary for both CG and HDG.

2. Notation

Most of the algebra presented in this text is expressed in symbolic (also frequently referred to as direct, intrinsic or absolute) notation, very similar to the one employed in [30]. The usual matrix and indicial notation are sometimes employed in specific cases.

Throughout the text, italic Latin or Greek lowercase letters ($a, b, \ldots \alpha, \beta, \ldots$) denote scalar quantities, bold italic Latin or Greek lowercase letters ($\boldsymbol{a}, \boldsymbol{b}, \ldots \boldsymbol{\alpha}, \boldsymbol{\beta}, \ldots$) denote vectors and bold italic Latin or Greek capital letters ($\boldsymbol{A}, \boldsymbol{B}, \ldots$) denote second-order tensors in a Euclidean space.

Rectangular and single-column matrices built of tensor components on orthogonal Cartesian frames are expressed by boldface upright Latin or Greek letters ($\mathbf{A}, \mathbf{B}, \ldots \mathbf{a}, \mathbf{b} \ldots \boldsymbol{\rho}, \boldsymbol{\lambda} \ldots$). The scalar products used in the present paper are $(\cdot,\cdot)_D$ and $\langle \cdot,\cdot \rangle_B$, which represent the $\mathcal{L}_2$ scalar product in any domain D, and the $\mathcal{L}_2$ scalar product over any boundary B, respectively.

In the following, the domain Ω is assumed to be divided into n_{el} elements, Ω_e, with the boundaries $\partial\Omega_e$,

$$\bar{\Omega} = \bigcup_{e=1}^{n_{el}} \bar{\Omega}_e, \quad \Omega_e \cap \Omega_k = \emptyset \quad \text{for} \quad e \neq k.$$

The union of the n_{fc} faces, Γ_f, is denoted as

$$\Gamma = \bigcup_{e=1}^{n_{el}} \partial\Omega_e = \bigcup_{f=1}^{n_{fc}} \Gamma_f.$$

Along the text P_k and Q_k denote the finite element spaces of degree k for triangles and quadrilaterals, respectively. In particular, Taylor-Hood elements consider degree k for velocity and degree $k-1$ for pressure and are denoted by P_k/P_{k-1} and Q_k/Q_{k-1}.

3. The CG and HDG discretization of the incompressible Navier–Stokes equations

Let $\Omega \subset \mathbb{R}^d$ be the domain with boundary $\partial\Omega$ divided into Dirichlet, $\partial\Omega_D$, and Neumann, $\partial\Omega_N$, boundaries and d the dimension of the space. The steady state incompressible Navier–Stokes equations can be written as

$$\operatorname{div}(\boldsymbol{u}\otimes\boldsymbol{u}) - \operatorname{div}(-p\boldsymbol{I} + \nu \operatorname{grad}\boldsymbol{u}) = \boldsymbol{f} \quad \text{in } \Omega, \tag{1a}$$

$$\operatorname{div}\boldsymbol{u} = 0 \quad \text{in } \Omega, \tag{1b}$$

$$\boldsymbol{u} = \boldsymbol{u}_D \quad \text{on } \partial\Omega_D,$$

$$(-p\boldsymbol{I} + \nu \operatorname{grad}\boldsymbol{u})\,\boldsymbol{n} = \boldsymbol{t} \quad \text{on } \partial\Omega_N, \tag{1c}$$

where $\boldsymbol{u}$ is the velocity, p is the kinematic pressure, ν is the kinematic viscosity, $\boldsymbol{f}$ is the body force, $\boldsymbol{u}_D$ is the prescribed velocity on the Dirichlet boundary, $\partial\Omega_D$, and $\boldsymbol{t}$ is the prescribed pseudo traction on the Neumann boundary, $\partial\Omega_N$.

The next subsections recall the basics on the CG and the HDG discretizations considered in this work.

3.1. CG formulation

The CG weak form of the equilibrium equations, (1a) and (1c), and the incompressibility condition, (1b), is: find $\boldsymbol{u}_h \in [\mathcal{V}_c^h(\Omega)]^d$ and $p_h \in \mathcal{V}^h(\Omega)$ such that $\boldsymbol{u}_h = \boldsymbol{u}_D$ on $\partial\Omega_D$ and

$$(\delta\boldsymbol{u}, (\operatorname{grad}\boldsymbol{u}_h)\,\boldsymbol{u}_h)_\Omega + (\operatorname{grad}\delta\boldsymbol{u}, -p_h\boldsymbol{I} + \nu\operatorname{grad}\boldsymbol{u}_h)_\Omega - (\delta\boldsymbol{u}, \boldsymbol{f}_h)_\Omega - \langle\delta\boldsymbol{u}, \boldsymbol{t}_h\rangle_{\partial\Omega_N} = 0, \tag{2a}$$

$$-(\delta p, \operatorname{div}\boldsymbol{u}_h)_\Omega = 0,$$

for all $\delta\boldsymbol{u} \in [\mathcal{V}_c^h(\Omega)]^d$, such that $\delta\boldsymbol{u} = \boldsymbol{0}$ on $\partial\Omega_D$, and for all $\delta p \in \mathcal{V}^h(\Omega)$, where discrete spaces are defined as

$$\mathcal{V}_c^h := \left\{v \in \mathcal{H}^1(\Omega)\ ;\ v|_{\Omega_e} \in \mathcal{P}_k(\Omega_e)\right\}, \quad \mathcal{V}^h := \left\{v \in \mathcal{L}_2(\Omega)\ ;\ v|_{\Omega_e} \in \mathcal{P}_k(\Omega_e)\right\}.$$

Equation (2a) is obtained after using the identity $\operatorname{div}(\boldsymbol{u}\otimes\boldsymbol{u}) = \boldsymbol{u}(\operatorname{div}\boldsymbol{u}) + (\operatorname{grad}\boldsymbol{u})\,\boldsymbol{u}$ and setting $\operatorname{div}\boldsymbol{u} = 0$.

If the problem is a pure Dirichlet one, that is $\partial\Omega = \partial\Omega_D$ and $\partial\Omega_N = \emptyset$, the pressure is determined up to a constant. In this case, the mean of the pressure in Ω is set to a prescribed value to ensure uniqueness of the solution.

The weak form is discretized with mixed Taylor–Hood approximations [31], with degree k for the velocities, $\boldsymbol{u}$ and $\delta\boldsymbol{u}$, and degree $k-1$ for the pressures, p and δp, satisfying the so-called LBB condition [32] for stability. The residual of the Navier–Stokes equations, after spatial discretization, can be expressed as follows,

$$\mathbf{r}\left(\mathbf{u},\mathbf{p}\right) \equiv \begin{bmatrix} \mathbf{K}+\mathbf{C}(\mathbf{u}) & \mathbf{G} \\ \mathbf{G}^T & \mathbf{0} \end{bmatrix} \begin{Bmatrix} \mathbf{u} \\ \mathbf{p} \end{Bmatrix} - \begin{Bmatrix} \mathbf{f}+\mathbf{t} \\ \mathbf{0} \end{Bmatrix} = \mathbf{0}. \tag{3}$$

In the equation (3), $\mathbf{K}$, $\mathbf{G}$ and $\mathbf{C}(\mathbf{u})$ represent the viscosity, discrete gradient operator and convective matrices, respectively, while $\mathbf{f}$ contains the body force vector. The nodal values of velocity and pressure are represented by $\mathbf{u}$ and $\mathbf{p}$, respectively. The elemental matrices used to compute the global system are presented in Appendix A.

The CG formulation, with Taylor–Hood approximations of order k for velocity and $k-1$ for pressure, leads to errors of order k in $\mathcal{H}^1$ norm for velocity and in $\mathcal{L}_2$ norm for pressure and, consequently, errors of order $k+1$ in $\mathcal{L}_2$ norm are expected for the velocity solution.

Computational efficiency in terms of CPU times is studied with numerical examples that do not present sharp fronts, aiming for a fair comparison of the accuracy. In the stability study, no stabilised formulations are considered for CG, as the aim is to compare the robustness without any stabilisation techniques.

3.2. HDG formulation

The HDG formulation of the Navier-Stokes equations (1) — see [33] for a detailed derivation — is: find $\boldsymbol{u}_h \in [\mathcal{V}^h]^d$, $p_h \in \mathcal{V}^h$, $\boldsymbol{L}_h \in [\mathcal{V}^h]^{d\times d}$, $\hat{\boldsymbol{u}}_h \in [\Lambda^h]^d$

and $\boldsymbol{\rho} \in \mathbb{R}^{n_{el}}$ satisfying the *local problem* in every element Ω_e,

$$(\delta \boldsymbol{L}, \boldsymbol{L}_h)_{\Omega_e} + (\operatorname{div} \delta \boldsymbol{L}, \boldsymbol{u}_h)_{\Omega_e} - \langle \delta \boldsymbol{L} \boldsymbol{n}, \hat{\boldsymbol{u}}_h \rangle_{\partial\Omega_e} = 0, \tag{4a}$$

$$\begin{aligned} -(\operatorname{grad} \delta \boldsymbol{u}, \boldsymbol{u}_h \otimes \boldsymbol{u}_h)_{\Omega_e} + (\delta \boldsymbol{u}, \operatorname{div}\, (-\nu \boldsymbol{L}_h + p_h \boldsymbol{I}))_{\Omega_e} \\ + \langle \delta \boldsymbol{u}, (\hat{\boldsymbol{u}}_h \otimes \hat{\boldsymbol{u}}_h)\, \boldsymbol{n} + \tau\, (\boldsymbol{u}_h - \hat{\boldsymbol{u}}_h) \rangle_{\partial\Omega_e} - (\delta \boldsymbol{u}, \boldsymbol{f})_{\Omega_e} = 0, \end{aligned} \tag{4b}$$

$$-(\operatorname{grad} \delta p, \boldsymbol{u}_h)_{\Omega_e} + \langle \delta p, \hat{\boldsymbol{u}}_h \cdot \boldsymbol{n} \rangle_{\partial\Omega_e} = 0, \tag{4c}$$

$$\frac{1}{|\partial\Omega_e|} \langle p_h, 1 \rangle_{\partial\Omega_e} = \rho_e, \tag{4d}$$

for $e = 1, \ldots, n_{el}$, and the *global problem*

$$\sum_{e=1}^{n_{el}} \langle \delta \hat{\boldsymbol{u}}, (-p_h \boldsymbol{I} + \nu \boldsymbol{L}_h)\, \boldsymbol{n} + \tau\, (\hat{\boldsymbol{u}}_h - \boldsymbol{u}_h) \rangle_{\partial\Omega_e} = \langle \delta \hat{\boldsymbol{u}}, \boldsymbol{t} \rangle_{\partial\Omega_N}, \tag{5a}$$

$$\langle \hat{\boldsymbol{u}}_h \cdot \boldsymbol{n}, 1 \rangle_{\partial\Omega_e} = 0 \;\text{ for } e = 1, \ldots, n_{el}, \tag{5b}$$

$$\hat{\boldsymbol{u}}_h = \mathbb{P}_2(\boldsymbol{u}_D) \;\text{ on } \partial\Omega_D,$$

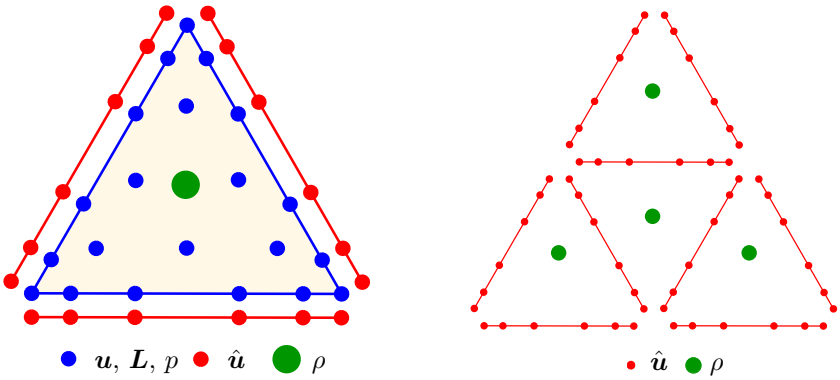
for all $\delta \boldsymbol{u} \in [\mathcal{V}^h]^d$, $\delta p \in \mathcal{V}^h$, $\delta \boldsymbol{L} \in [\mathcal{V}^h]^{d \times d}$ and $\delta \hat{\boldsymbol{u}} \in [\Lambda^h]^d$, such that $\delta \hat{\boldsymbol{u}} = \boldsymbol{0}$ on $\partial\Omega_D$, where the discrete space, Λ^h, is defined as

$$\Lambda^h := \big\{ \hat{v} \in \mathcal{L}_2(\Gamma)\ ;\ \hat{v}|_{\Gamma_i} \in \mathcal{P}_k(\Gamma_i) \big\},$$

$\mathcal{P}_k$ is the space of polynomials of degree less or equal to k, and $\mathbb{P}_2(\boldsymbol{u}_D)$ is the $\mathcal{L}_2$ projection of the Dirichlet data into the approximation space on $\partial\Omega_D$. Equations (4a) and (4b) include the variation and the discretized form of the newly introduced variable $\boldsymbol{L}$, which independently approximates $\operatorname{grad} \boldsymbol{u}$. Following [12], τ is a positive parameter, and it is usually taken as

$$\tau \approx \frac{\nu}{h} + |\boldsymbol{u}|,$$

where h is the element characteristic length. Even though the so-called stabilization parameter has some influence on the accuracy of the HDG solution, the method is very robust to variations of τ, see [9, 34]. Nevertheless, as will be seen in the numerical tests in section 7, this parameter may have an important effect in the stability properties of the HDG method, to alleviate or remove numerical oscillations in the presence of sharp fronts.



(a) Local problem (b) Global problem

Figure 2: Representation of the nodes for the discretization of the HDG.

The salient feature of HDG is the introduction of an independent approximation for the trace of the velocity, $\hat{\boldsymbol{u}}$, on the mesh skeleton, Γ. The introduction of this trace velocity defines two types of problems, namely local and global. The local problem corresponds to the solution of Navier–Stokes equations inside each element with $\hat{\boldsymbol{u}}$ as Dirichlet boundary condition. However, a Navier–Stokes problem with Dirichlet boundary condition on all the boundary is not solvable unless a condition on pressure is imposed. Hence, a new variable, $\rho_e \in \mathbb{R}$, is introduced as the mean of the pressure on the boundary of each element.

Figure 2a shows a representation of the nodes for the discretization of the local variables, in blue, the nodes on the skeleton of the mesh Γ for trace variables, in red, and the mean pressure, represented by a green dot. Note that the mean of the pressure, ρ_e, is not a physical node and it is just a scalar value defined for each element, Ω_e.

The local problem for each element (4) allows expressing the velocity, $\boldsymbol{u}_h$, the gradient of the velocity, $\boldsymbol{L}_h$, and the pressure, p_h, in the element, in terms of the trace of the velocity, $\hat{\boldsymbol{u}}_h$, on the mesh skeleton and the mean of the pressure at the element, ρ_e. Therefore, $\hat{\boldsymbol{u}}_h$ and $\boldsymbol{\rho}$ can be regarded as actual unknowns of the problem, that are determined with the global problem (5). The global problem corresponds to the discretization of the so-called conservativity

condition, that is, the conservation of the numerical pseudo-tractions across interior faces, together with the Neumann and Dirichlet boundary conditions, and the solvability condition (5b) for the Dirichlet data in the local problems. In the case of a pure Dirichlet problem, that is $\partial\Omega = \partial\Omega_D$, the mean of the pressure ρ_e is set to a constant in, for instance, a single element, closing the problem with an unique solution.

The discretization of local and global problems (4)-(5) leads to a discrete residual of the form

$$\mathbf{r} = \left[\begin{array}{cc:cccc} \mathbf{A}_{\hat{u}\hat{u}} & \mathbf{0} & \mathbf{A}_{\hat{u}u} & \mathbf{A}_{\hat{u}L} & \mathbf{A}_{\hat{u}p} & \mathbf{0} \\ \mathbf{A}_{\rho\hat{u}} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \hdashline \mathbf{A}_{u\hat{u}} + \mathbf{C}_{u\hat{u}}(\hat{\mathbf{u}}) & \mathbf{0} & \mathbf{A}_{uu} + \mathbf{C}_{uu}(\mathbf{u}) & \mathbf{A}_{uL} & \mathbf{A}_{up} & \mathbf{0} \\ \mathbf{A}_{L\hat{u}} & \mathbf{0} & \mathbf{A}_{Lu} & \mathbf{A}_{LL} & \mathbf{0} & \mathbf{0} \\ \mathbf{A}_{p\hat{u}} & \mathbf{0} & \mathbf{A}_{pu} & \mathbf{0} & \mathbf{0} & \mathbf{A}_{\rho p}^T \\ \mathbf{0} & -\mathbf{1} & \mathbf{0} & \mathbf{0} & \mathbf{A}_{\rho p} & 0 \end{array}\right] \left\{\begin{array}{c} \hat{\mathbf{u}} \\ \boldsymbol{\rho} \\ \hdashline \mathbf{u} \\ \mathbf{L} \\ \mathbf{p} \\ \boldsymbol{\lambda} \end{array}\right\} - \left\{\begin{array}{c} \mathbf{t} \\ \mathbf{0} \\ \hdashline \mathbf{f} \\ \mathbf{0} \\ \mathbf{0} \\ \mathbf{0} \end{array}\right\} = \mathbf{0}. \tag{6}$$

The nodal values of $\hat{\boldsymbol{u}}_h$, $\boldsymbol{u}_h$, $\boldsymbol{L}_h$ and p_h are represented by $\hat{\mathbf{u}}$, $\mathbf{u}$, $\mathbf{L}$ and $\mathbf{p}$, respectively. The constraints (4d) are applied using the Lagrangian multiplier $\boldsymbol{\lambda}$. The dashed lines inside the matrix separates the global and local problems. The elemental matrices used to compute the system are again presented in Appendix A.

The equations below the dashed line in (6) correspond to the discretization of the local problems (4). In the linear case, for the Stokes equations, these can be solved element-by-element to define the so-called local solver, *i.e.*, the expression of the local variables $\mathbf{u}^{(e)}$, $\mathbf{L}^{(e)}$ and $\mathbf{p}^{(e)}$ (where (e) denotes the nodal values for element Ω_e) in terms of global variables $\hat{\mathbf{u}}$ and ρ_e. Replacement of the local solver for each element in the global equations (5), *i.e.*, in the equations above the dashed line in (6), leads to a system of equations involving only the global variables $\hat{\mathbf{u}}$ and $\boldsymbol{\rho}$. Figure 2b represents the nodes corresponding to the actual DOFs of the global problem of HDG. After the global problem is solved, the solution inside each element, $\mathbf{u}^{(e)}$, $\mathbf{L}^{(e)}$ and $\mathbf{p}^{(e)}$, can be computed with the local solver.

Remark 1. *Equation* (4d) *must be scaled by the perimeter of the element,* $|\partial\Omega_e|$*, to get a symmetric system, in the case of Stokes problem, after condensation of the local variables. Another possibility is choosing the average of the pressure inside each element, which must be scaled by the area of the element.*

In the non-linear case, for the Navier-Stokes equations, the solution of the non-linear system with Newton-Raphson leads to a linear system of equations to be solved in each iteration. This linear system can be solved analogously to the Stokes solution, that is, the local equations are solved element-by-element, to express the local variables in terms of global variables, leading to a smaller linear system of equations involving only trace variables and the mean of the pressure in the elements.

The HDG formulation provides a numerical solution with optimal convergence of order $k+1$ in $\mathcal{L}_2$ norm for the velocity, $\boldsymbol{u}_h$, the pressure, p_h, and also for the approximation of the gradient, $\boldsymbol{L}_h$. In addition, the mean of the velocity is each element, $(\boldsymbol{u}_h, 1)_{\Omega_e}$, is super-convergent with errors of order $k+2$. Hence, a new super-convergent approximation of velocity, $\boldsymbol{u}_h^*$, can be computed by solving a new problem in each element: find $\boldsymbol{u}_h^* \in \left[\mathcal{V}^{h^*}\right]^d$ such that

$$
\begin{aligned}
&(\operatorname{grad}\delta\boldsymbol{u}^*, \operatorname{grad}\boldsymbol{u}_h^*)_{\Omega_e} = (\operatorname{grad}\delta\boldsymbol{u}^*, \boldsymbol{L}_h)_{\Omega_e}\,,\\
&(\boldsymbol{u}_h^*, 1)_{\Omega_e} = (\boldsymbol{u}_h, 1)_{\Omega_e}\,,
\end{aligned}
$$

for all $\delta\boldsymbol{u}^* \in \left[\mathcal{V}^{h^*}\right]^d$ and $e = 1, \ldots, n_{el}$, where $\mathcal{V}^{h^*}$is a richer space with one polynomial degree more than $\mathcal{V}^h$, *i.e.* $\mathcal{V}^{h^*} = \left\{\boldsymbol{v} \in \mathcal{L}_2(\Omega)\ ;\ \boldsymbol{v}|_{\Omega_e} \in \mathcal{P}^{k+1}(\Omega_e),\ \text{for } e = 1, \ldots, n_{el}\right\}$. The super-convergent velocity, $\boldsymbol{u}^*$, converges asymptotically with a rate of $k+2$ in the $\mathcal{L}_2$ norm for a mesh with uniform degree, k, see [35]. Convergence properties for the non-uniform degree are discussed in detail in [36, 37]. The computational overhead in computing the super-convergent solution, $\boldsymbol{u}_h^*$, is small. This solution can be used to define a reliable and inexpensive error estimator for HDG velocity approximation, $\boldsymbol{u}_h$, see [33].

4. Implementation

In all the results presented in subsequent section, the shape functions that are used to approximate the variables inside each element are generated using Fekete nodal distributions [38] for triangular elements and Gauss–Lobatto points [39, p. 888] in the case of quadrilateral elements. The shape functions are computed using Jacobi polynomials discussed in detail in [40]. All the meshes are generated using EZ4U [41, 42, 43], which is a high order mesh generator, and Gmsh [44] is used to post process the results.

The non-linear system of equations is linearised using Newton–Raphson method. Relative incremental and residual norms are used as convergence criteria with a tolerance of 10^{-12}.

The in-house code is implemented in FORTRAN. Only direct solvers are considered in the present work. Harwell Subroutine Library (HSL) [45] routines MA57d and MA41d [46] are used for solving symmetric and unsymmetric systems, respectively. Both solvers use Approximate Minimum Degree (AMD) [47] reordering algorithm to reorder the linear system of equations. MC75d [48] is used to estimate the condition number of the tangent stiffness matrices, $\kappa(\mathbf{A})$.

Let $\boldsymbol{w}$ and z be generic vector and scalar fields, respectively, defined over Ω. Their error norms are computed as follows,

$$||e_{\boldsymbol{w}}||_{\mathcal{L}_2(\Omega)} = \left[\int_\Omega (\boldsymbol{w}_{ex} - \boldsymbol{w}_{num}) \cdot (\boldsymbol{w}_{ex} - \boldsymbol{w}_{num})\, \mathrm{d}\Omega\right]^{1/2},$$

$$||e_z||_{\mathcal{L}_2(\Omega)} = \left[\int_\Omega (z_{ex} - z_{num})^2\, \mathrm{d}\Omega\right]^{1/2},$$

where suffixes ex and num stand for exact and numerical values.

All tests were performed on machine equipped with 24 Intel(R) Xeon(R) E5-2620 v2 2.10-2.60 GHz processors and 64 GB of RAM running OpenSUSE 13.1 (x86_64) using a serial implementation. The code was compiled using gfortran 4.8.1.

4.1. Static condensation

Static condensation is used in both CG and HDG methods in the present work: the DOFs that are not shared by neighbouring elements can be expressed in terms of remaining DOFs of the element, hence reducing the global DOFs of the system. In the case of CG, interior nodes are not shared by other elements and, therefore, they can be expressed in terms of boundary nodes of the element. In the case of HDG, all the local DOFs are approximated independently inside each element, consequently, they can be expressed in terms of global DOFs.

Let $\delta\mathbf{x}_m$ and $\delta\mathbf{x}_s$ be the incremental master and slave DOFs in a linearised system. In the case of CG, master and slave DOFs correspond to boundary and interior DOFs respectively, while in HDG, they represent local and global DOFs, respectively. A typical linearised system of equations can be written in the form,

$$\begin{bmatrix} \mathbf{A}_{mm} & \mathbf{A}_{ms} \\ \mathbf{A}_{sm} & \mathbf{A}_{ss} \end{bmatrix} \begin{Bmatrix} \delta\mathbf{x}_m \\ \delta\mathbf{x}_s \end{Bmatrix} = - \begin{Bmatrix} \mathbf{r}_m \\ \mathbf{r}_s \end{Bmatrix}. \tag{7}$$

The matrix $\mathbf{A}_{ss}$ is block diagonal and its inverse is well defined. Hence, $\delta\mathbf{x}_s$ can be expressed in terms of $\delta\mathbf{x}_m$ in an element-by-element fashion as,

$$\delta\mathbf{x}_s^{(e)} = \left[-\mathbf{A}_{ss}^{(e)}\right]^{-1} \mathbf{A}_{sm}^{(e)} \delta\mathbf{x}_m - \left[\mathbf{A}_{ss}^{(e)}\right]^{-1} \mathbf{r}_s^{(e)}. \tag{8}$$

Note that (e) represents the elemental matrices. Replacing $\delta\mathbf{x}_s$ from equation (8) into the first of the (7) equations and assembling into the global system results in the following,

$$\left(\mathbf{A}_{mm} - \mathbf{A}_{ms}\mathbf{A}_{ss}^{-1}\mathbf{A}_{sm}\right) \delta\mathbf{x}_m = -\mathbf{r}_m + \mathbf{A}_{ms}\mathbf{A}_{ss}^{-1}\mathbf{r}_s.$$

As the matrices are condensed on elemental basis, the computational overhead is negligible. In the case of a problem with very high DOF count, this numerical technique can save significant CPU time in solving the system of equations, specially for high degree approximations.

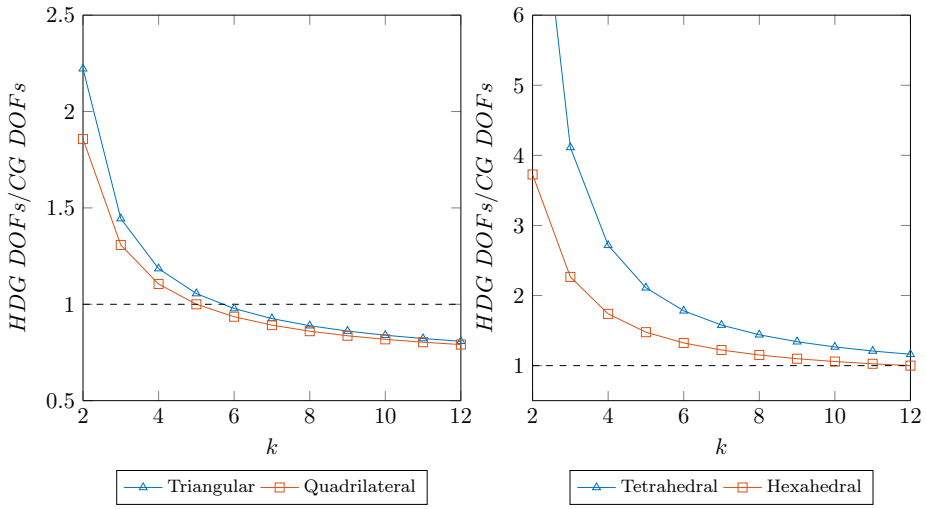
Table 1: Analytical expressions for number of DOFs in HDG and CG.

Method	2D	
	Triangular	Quadrilateral
HDG(P_k/Q_k)	$(3k+4)\,n_{el}$	$(4k+5)\,n_{el}$
CG(P_kP_{k-1}/Q_kQ_{k-1})	$\left(\frac{9k-9}{2}\right)n_{el}$	$(6k-5)\,n_{el}$
	3D	
	Tetrahedral	Hexahedral
HDG(P_k/Q_k)	$\left(3k^2+9k+7\right)n_{el}$	$\left(9k^2+18k+10\right)n_{el}$
CG(P_kP_{k-1}/Q_kQ_{k-1})	$\left(4k^2-\frac{28k}{3}+\frac{41}{6}\right)n_{el}$	$\left(12k^2-18k+10\right)n_{el}$

5. Count of DOFs

A theoretical count of the approximate number of DOFs, for both HDG (P_k or Q_k) and CG (P_kP_{k-1} or Q_kQ_{k-1}), is presented. Table 1 presents the number of DOFs in terms of the number of elements, n_{el}. These formulas have been derived using the relations proposed in the appendix of [24] to express the number of nodes and faces in terms of the number of elements. The main assumption used in this analysis is uniform structured meshes with very large number of elements. The ratio of number of DOFs of HDG to CG is plotted against the degree of approximation, k, in figure 3 in 2D and 3D spaces. In both cases only the number of DOFs after static condensation are taken into account.

It is evident from the figure 3a that HDG has less DOFs than CG for both triangular and quadrilateral elements when $k > 5$ in the case of 2D. This is due to the fact that, even though HDG has more DOFs for velocity, the pressure DOFs are condensed to a single scalar unknown per element, while in CG only interior pressure DOFs can be condensed out. Hence, at high-order approxima-

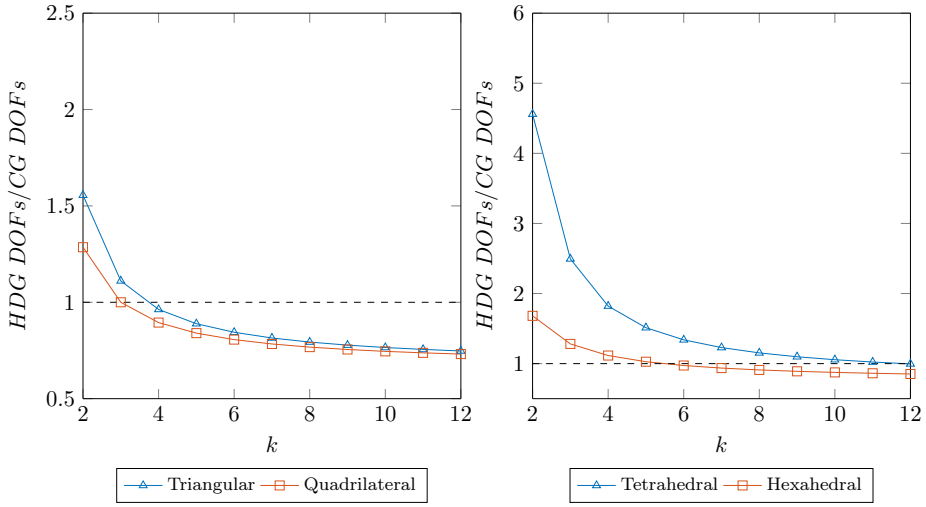


(a) 2D. (b) 3D.

Figure 3: Comparison of DOFs between HDG (P_k/Q_k) and CG (P_kP_{k-1}/Q_kQ_{k-1}).

tions, HDG tends to have lesser DOFs than CG. On the other hand, for 3D, HDG has less DOFs than CG only for very high degrees: $k = 12$ for hexahedral and $k = 19$ for tetrahedral meshes of elements. In this case, both types of elements have more nodes along the edges than vertices in the corresponding 2D case, hence more velocity DOFs are repeated in HDG. However, the post-processed solution of HDG (P_{k-1}/Q_{k-1}) with degree $k-1$ has the same order of convergence as CG (P_kP_{k-1}/Q_kQ_{k-1}) solution of degree k for velocity. Accordingly, a plot is presented in figure 4, comparing the ratio of number of DOFs of HDG (P_{k-1}/Q_{k-1}) to CG (P_kP_{k-1}/Q_kQ_{k-1}). The ratio favours HDG when $k > 4$ in the case of 2D and, in the case of 3D, $k > 6$ and $k > 12$ for hexahedral and tetrahedral elements, respectively.

Table 2 shows the various quantities of interest regarding the linear system of equations for Stokes problem. The number of DOFs of the system is denoted by `ndof` and number of non-zeros of the global stiffness matrix matrix and its factor by $\texttt{nnz}(\mathbf{A})$ and $\texttt{nnz}(\mathbf{L})$, respectively. Since, Stokes problems leads to a symmetric matrix, only lower triangular part of the matrix is stored. All



(a) 2D. (b) 3D.

Figure 4: Comparison of DOFs between HDG (P_{k-1}/Q_{k-1}) and CG (P_kP_{k-1}/Q_kQ_{k-1}).

the numbers are provided only for free DOFs excluding DOFs corresponding to Dirichlet boundary. From the table, it can be verified that the number of DOFs of HDG is less than CG for $k > 5$, as deduced from theoretical count in fig. 3a. Nevertheless, the number of non-zeros in the global matrix and its factor are very similar for $k = 4$ and perhaps, HDG system leads to fewer entries from $k \geq 5$. At higher degrees the entries in the factors of CG systems are almost 1.5 times more than HDG systems. This can favour the HDG systems when using the direct solvers.

It is also worth noting that HDG matrices have a regular block sparsity pattern that is beneficial for the direct solver, see [20] for the Laplace equation. Consider a mesh with triangular elements: each face has contributions from 4 other faces, as shown in figure 2b. Hence, each row in the final system of HDG has 5 blocks of equal size for velocity DOFs. In figure 5, the sparsity pattern of matrices of HDG and CG are shown for a regular mesh with 32 triangular elements. The blue DOFs correspond to velocity-velocity, the black DOFs correspond to pressure-pressure, in the case of CG, and the red ones are

Table 2: Comparison of DOFs (`ndof`), number of non-zeros (`nnz`) in global stiffness matrix ($\mathbf{A}$) and its factor ($\mathbf{L}$) for Stokes problem with $n_{el} = 2048$ triangular elements.

	k	CG(P_kP_{k-1})	HDG(P_k)	k	CG(P_kP_{k-1})	HDG(P_k)
`ndof`	2	9 029	20 095	6	45 634	44 159
`nnz`($\mathbf{A}$)		69 641	313 267		2 007 486	1 556 131
`nnz`($\mathbf{L}$)		133 866	1 484 136		9 389 209	7 225 163
`ndof`	3	18 178	26 111	7	54 786	50 175
`nnz`($\mathbf{A}$)		400 191	535 279		2 812 691	2 014 687
`nnz`($\mathbf{L}$)		1 955 880	2 437 997		12 741 010	9 000 060
`ndof`	4	27 320	32 127	8	65 938	56 191
`nnz`($\mathbf{A}$)		801 236	816 427		3 752 616	2 532 379
`nnz`($\mathbf{L}$)		3 755 471	3 751 615		17 006 063	11 344 166
`ndof`	5	36 482	38 143	9	73 090	62 207
`nnz`($\mathbf{A}$)		1 337 001	1 156 711		7 827 261	3 109 207
`nnz`($\mathbf{L}$)		6 276 872	5 347 337		21 918 136	13 959 296

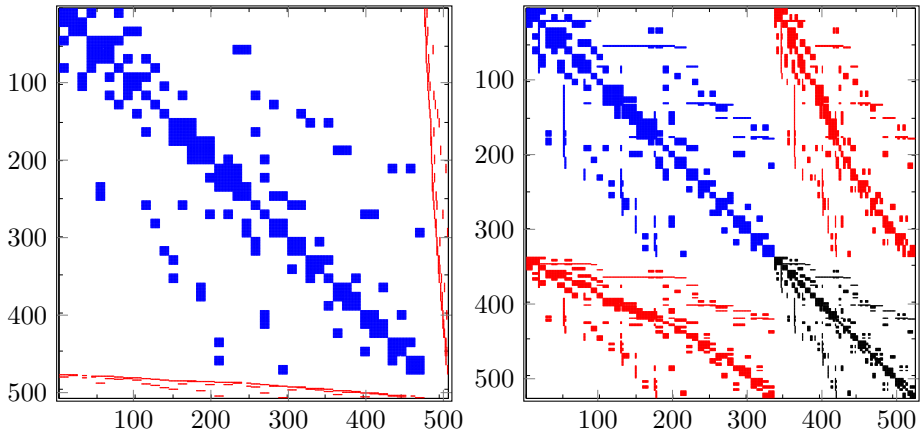
velocity-pressure, in the case of CG, and velocity-mean pressure, in the case of HDG.

6. Comparison of computational efficiency

6.1. Kovasznay flow

The benchmark problem Kovasznay flow is considered for the comparison of CG and HDG for the solution of Stokes and Navier–Stokes problems. Kovasznay flow is an analytical solution of Navier–Stokes equations in a domain $[0, 2] \times [-0.5, 1.5]$, see [49].

$$\boldsymbol{u} = \begin{bmatrix} 1 - \exp(\lambda x_1)\cos(2\pi x_2) \\ \frac{\lambda}{2\pi}\exp(\lambda x_1)\sin(2\pi x_2) \end{bmatrix},$$
$$p = -\frac{1}{2}\exp(2\lambda x_1) + C,$$



(a) Sparsity pattern of HDG, DOFs = 511, $nnz = 1830$.

(b) Sparsity pattern of CG, DOFs = 530, $nnz = 3580$.

Figure 5: Sparsity pattern of HDG (P_k) and CG ($P_k P_{k-1}$) matrices for $k = 5$ and $h = 0.5$. Variable nnz represents number of non-zeros.

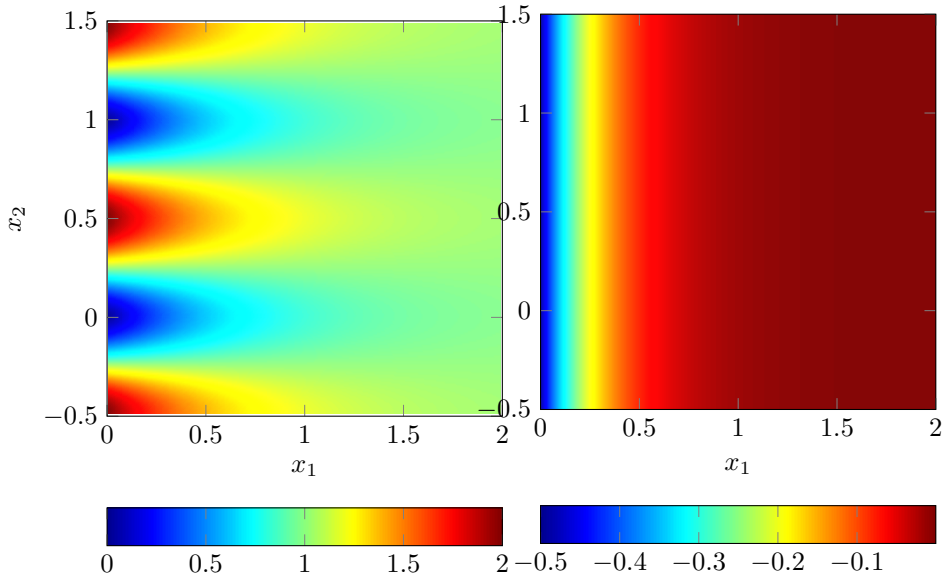
where $\lambda = \frac{Re}{2} - \sqrt{\frac{Re^2}{4} + 4\pi^2}$ and $Re = \frac{1}{\nu}$ is the Reynolds number. The analytical velocity and pressure are shown in figure 6. Dirichlet boundary conditions are prescribed for the velocity on all the boundary. In the case of Stokes problem, a body force equal to the convective term, $\mathrm{div}\,(\boldsymbol{u} \otimes \boldsymbol{u})$, is set using the exact solution at $Re = 20$.

Meshes are obtained by splitting a regular $n \times n$ Cartesian grid into either $2n^2$ triangles or n^2 quadrilaterals, which gives an uniform element size, $h = 2/n$. All the results are presented considering the stability parameter $\tau = 1$ on all faces of each element.

In this section, results are presented for the Stokes problem, and just commented for the Navier–Stokes case.

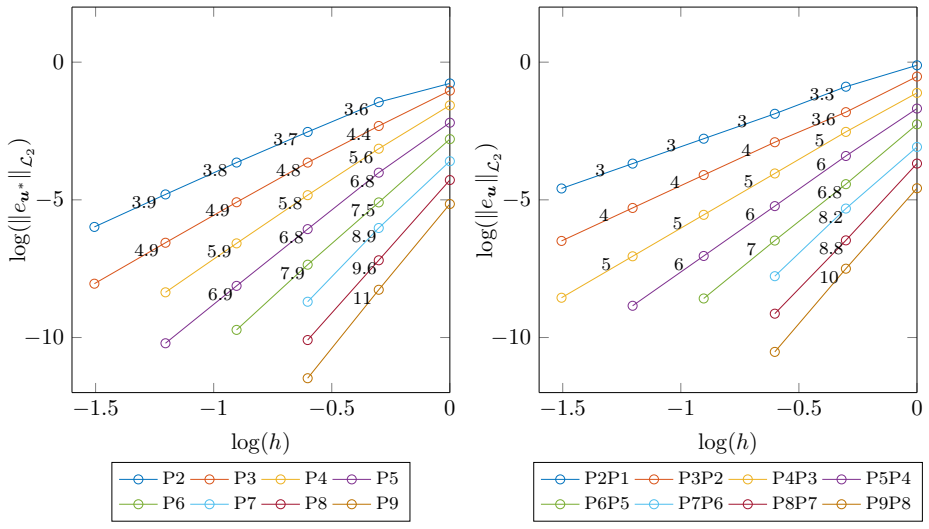
Convergence

The $\mathcal{L}_2$ norm of the error in the post-processed velocity of HDG is compared to the $\mathcal{L}_2$ norm of error in velocity of CG. Similarly, the $\mathcal{L}_2$ norm of the error in pressure is also compared between HDG and CG.



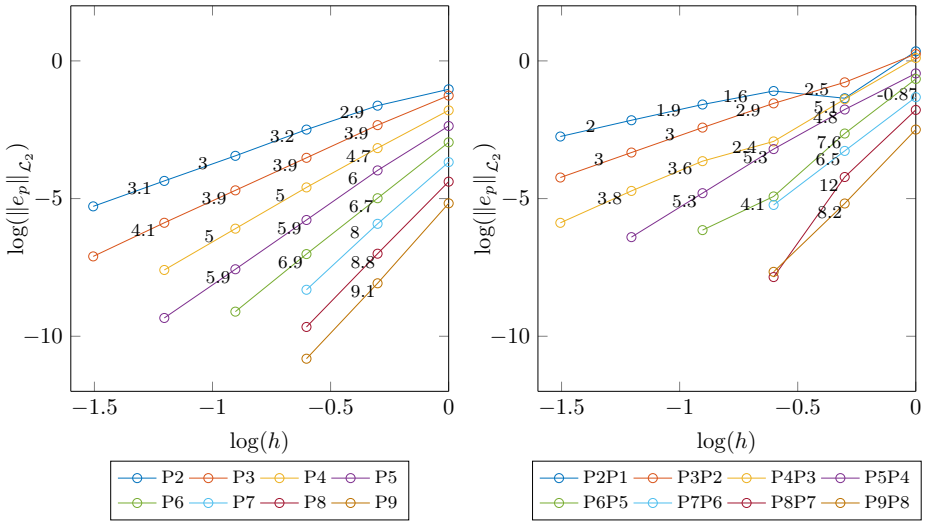
(a) Velocity. (b) Pressure.

Figure 6: Kovasznay flow: analytical solution.



(a) HDG: $\boldsymbol{u}^*$. (b) CG: $\boldsymbol{u}$.

Figure 7: Kovasznay flow: convergence of velocity.



(a) HDG: p.

(b) CG: p.

Figure 8: Kovasznay flow: convergence of pressure.

Figure 7 shows the convergence plots, for triangular elements, of velocity for HDG and CG, while figure 8 has the convergence results of pressure. In HDG, both velocity and pressure are approximated with the same degree of approximation, k, while in the case of CG, degree k is used for velocity and $k-1$ for pressure. The HDG post-processed velocity converges with order $k+2$, compared to $k+1$ for CG. Similarly, HDG has $k+1$ rate for pressure compared to the rate of k for pressure of CG.

Although not presented, optimal theoretical convergence with similar accuracy is observed for Navier–Stokes problem in both HDG and CG.

It is noticed that the CPU times for the Kovaszany flow problem are too small for the considered mesh sizes to make a reliable comparison. Hence, a new test case is proposed in the following section in order to compare the computational efficiency between CG and HDG.

6.2. Test case

This test case is chosen in such a way that it facilitates the use of finer meshes and hence, larger CPU times and errors stay within the level of 10^{-10}. The analytical solutions of the velocity and pressure considered are,

$$\boldsymbol{u} = \begin{bmatrix} -\cos(70\,x_1)\sin(70\,x_2) \\ \sin(70\,x_1)\cos(70\,x_2) \end{bmatrix},$$
$$p = -\frac{1}{4}(\cos(140\,x_1) + \cos(140\,x_2)) + C,$$

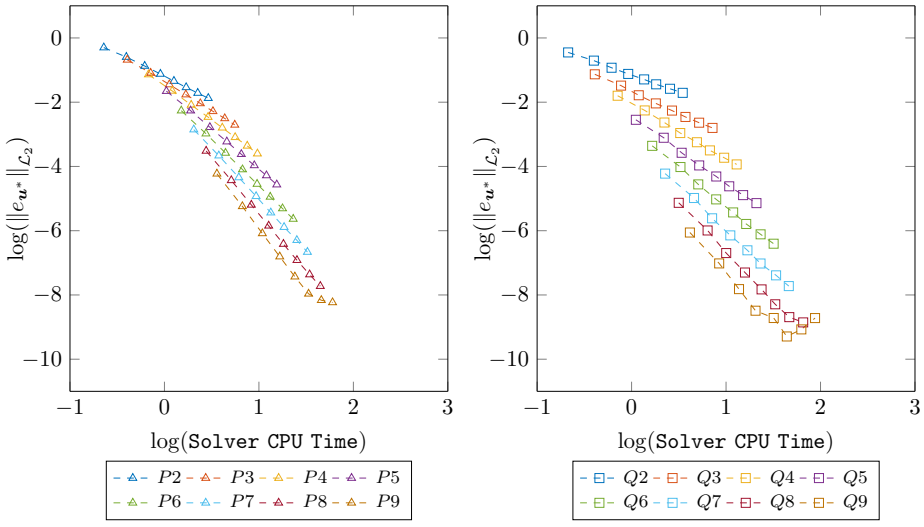
in a domain of $[0, 2] \times [0, 2]$. Dirichlet boundary conditions are applied on the boundary along with appropriate body force computed from the analytical solution. Stabilisation parameter, τ, in HDG is taken as 1 on all faces of the each element.

CPU time for direct solver

CPU times for direct solver (in seconds) are presented, for both HDG and CG, to compare their computational efficiency. The time taken for pre-processing, computation and assembly of matrices and post-processing is highly implementation-dependent and hence, not taken into account. The errors considered for comparison are the $\mathcal{L}_2$ norm of error in post-processed velocity in the case of HDG, with degree k for all variables, and the $\mathcal{L}_2$ norm of error in velocity of CG, with degree k for velocity and $k-1$ for pressure.

The results are presented for degree from 2 to 9 and the element sizes, h, used are $2/\{32, 40, 48, 56, 64, 72.80, 88\}$. The size of the elements are chosen in such a way to keep the errors within acceptable bounds.

Figures 9 and 10 show the CPU time taken by the direct solver for various degrees of approximation, for both HDG and CG, in the case of Stokes problem with triangular and quadrilateral elements. In the plots, each data point corresponds to the mesh size specified. A common trend can be observed from the plots: to achieve the same level of accuracy, it is more computationally efficient to use a high-order coarser mesh than a low-order finer mesh. This may be due to the fact that the data dependencies in a high-order mesh between elements



(a) Triangular elements.

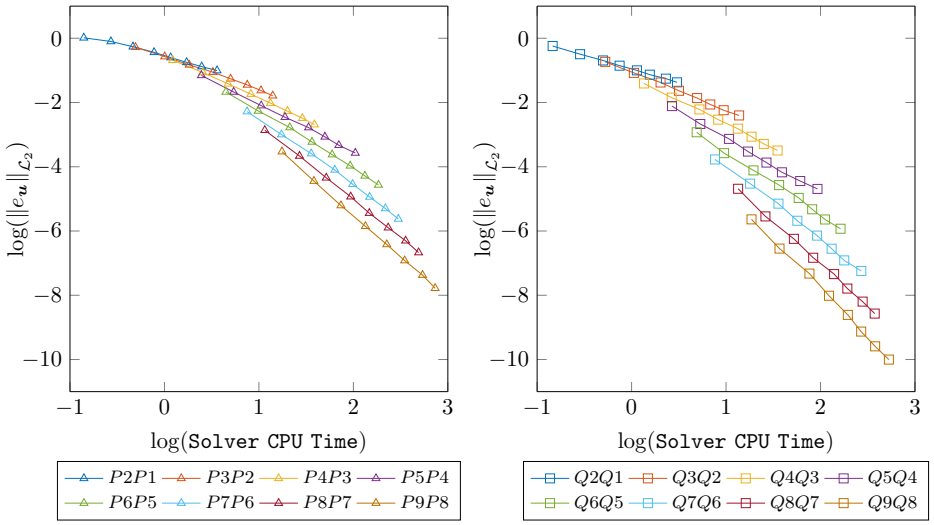
(b) Quadrilateral elements.

Figure 9: Test case: error for HDG *vs.* CPU time for direct solver with triangular and quadrilateral elements.

are lesser than in a low-order mesh, see [21]. A similar trend is observed in the case of Navier–Stokes results and hence, plots are omitted.

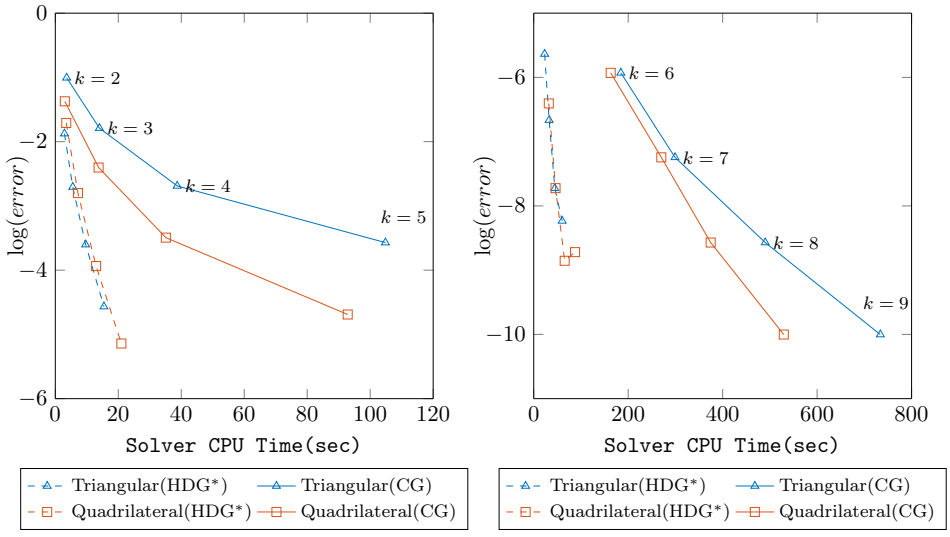
Figure 11 presents a similar plot for a given mesh, with element size $h = 2/88$ for $k = 2-9$, comparing the efficiency of HDG and CG in a single plot. Asterisk (*) on HDG denotes the post-processed solution for velocity. For a given error, HDG always outperforms CG at all the degrees of approximation presented, and the performance gap between CG and HDG increases with increasing degree of approximation. A similar trend in results is obtained for Navier–Stokes problem, but with a greater difference in the CPU times between HDG and CG at $k = 5$.

Figure 12 shows the ratio of CPU times of HDG to CG, for triangular elements and quadrilateral elements. It can be observed from these plots that, for degree $k \geq 3$, all points lie below the ratio of 1, indicating that HDG takes lesser CPU time for direct solver. Even in the case of $k = 2$, as the mesh becomes finer, HDG becomes more efficient than CG for triangular elements. In case of quadrilateral elements, HDG seems to be more efficient only for degree



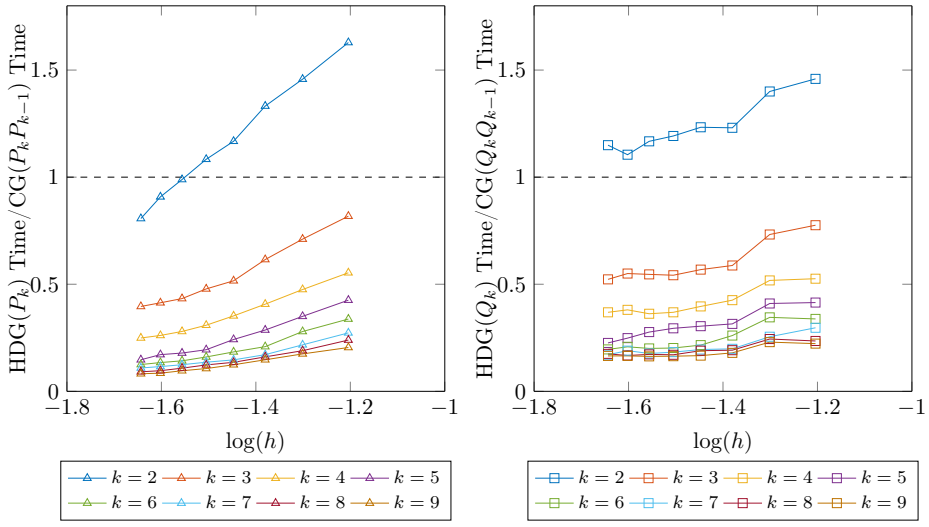
(a) Triangular elements. (b) Quadrilateral elements.

Figure 10: Test case: error for CG *vs.* CPU time for direct solver with triangular and quadrilateral elements.



(a) $h = 2/88$ and $k = 2 - 5$. (b) $h = 2/88$ and $k = 6 - 9$.

Figure 11: Test case: comparison of error between HDG and CG *vs.* CPU time for direct solver.



(a) Triangular elements.

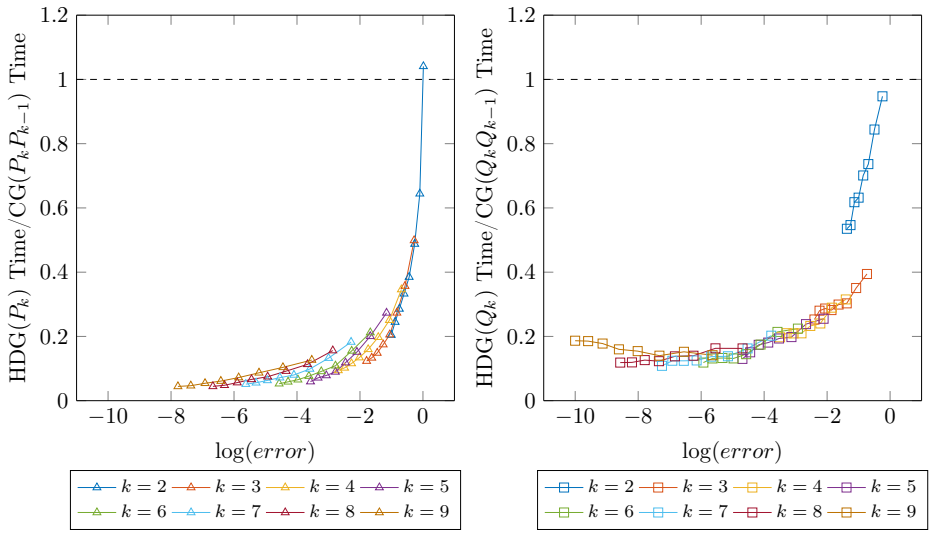
(b) Quadrilateral elements.

Figure 12: Test case: ratio of CPU times for direct solver *vs.* element size for triangular and quadrilateral elements.

$k \geq 3$. In the case of high-order, the ratio of CPU times tend to be constant for all mesh sizes, owing to the larger computational times for both triangular and quadrilateral elements.

Moreover, it is noticed from the convergence history that the HDG produces lesser error compared to CG for same mesh and same degree of approximation. Hence, a comparison of CPU time against the error is a fairer comparison.

Figure 13 shows the ratio of CPU times against error. These plots are produced using the CPU times for direct solver in figures 9 and 10. For a given degree of approximation and mesh, HDG produces lesser error than CG, thus, for every mesh in the CG plot, the CPU time of HDG is interpolated from figure 9 to determine the CPU time that would provide the same error as CG. It is evident from the plots that HDG is more efficient than CG for a given accuracy in both triangle and quadrilateral cases. It can also be noticed that the ratio of CPU times tend to a constant value for all degrees of approximation and mesh sizes in triangular elements. For the case of quadrilateral elements, same can



(a) Triangular elements. (b) Quadrilateral elements.

Figure 13: Test case: ratio of CPU times for direct solver *vs.* error for triangular and quadrilateral elements.

be concluded that most of the points produce a constant ratio of CPU times. However, the ratio for $k = 9$ tends to be increasing at the finer mesh sizes. This can be explained from the figure 9b, where last two points for $k = 9$ are within machine precision and hence, optimal convergence is lost. Nevertheless, it can be safely concluded that HDG can outperform CG for a given level of accuracy at high-order approximations.

In the case of Navier–Stokes, a similar trend is observed, with ratios below 1. Hence, the results are not presented to avoid redundancy.

Finally, the condition number of global stiffness matrix of the Stokes problem, $\kappa(\mathbf{A})$, is plotted against degree of approximation, k, for a mesh with triangular elements and $h = 0.03125$, in figure 14. Same choice of basis functions are used to compute the elemental matrices for both HDG and CG to be able to make a fair comparison of condition numbers. The condition number increases more rapidly in the case of CG than HDG: condition number in HDG increases by one order of magnitude when going from degree 2 to 9, whereas in CG it

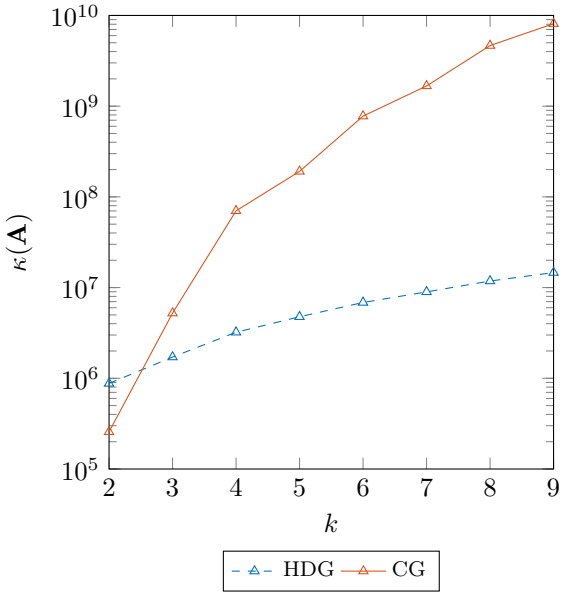


Figure 14: Test case: Condition numbers in Stokes problem with triangular elements and $h = 0.03125$.

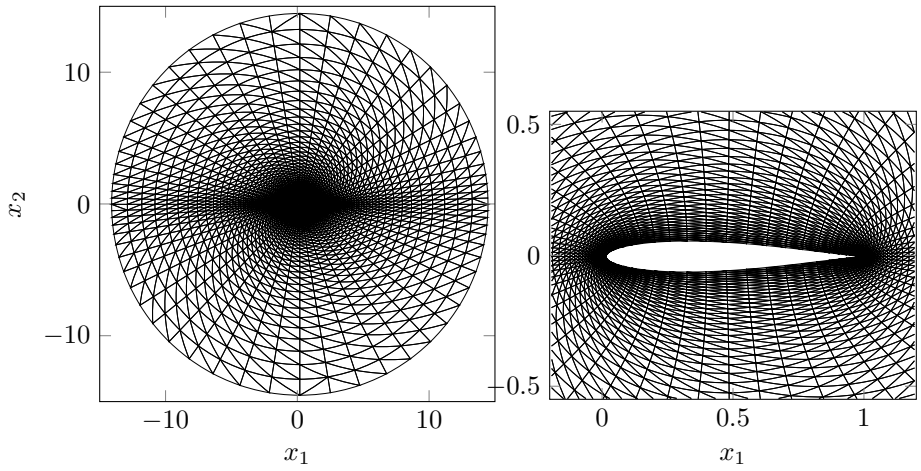
increases by approximately 5 orders.

6.3. NACA airfoil

The results presented until this point are for regular uniform meshes with a benchmark problem. Now, a more practical problem is considered in this section: the computation of the lift coefficient for the NACA0012 airfoil section at Reynolds number $Re = 5000$ and angle of attack $\alpha = 2°$. Steady state Navier–Stokes equations are solved and the error in the lift coefficient is used for presenting the results.

The meshes are generated using an in-house code for a computational domain with a circular exterior boundary. Four different meshes are used in the computations with nested refinement. Figure 15 shows the most refined mesh used in the computations and the region around the airfoil.

The mesh is non-uniform with refinement in the vicinity of the airfoil section. Depending on the angle of attack, α, the velocity on the inflow half of the



(a) NACA computational mesh.

(b) Mesh around airfoil.

Figure 15: NACA airfoil: computational mesh (refinement level 4) and zoom.

boundary is prescribed to $(\cos(\alpha), \sin(\alpha))$. The rest of the exterior boundary is treated as outflow boundary, as it is far from the airfoil. No slip boundary condition is applied along the boundary of the airfoil.

The velocity field around the airfoil, obtained using the mesh presented in figure 15 and degree of approximation $k = 5$, is shown in figure 16. The singularity at the front tip of the airfoil can be noticed, and it can be observed that the wake region of the airfoil is steady at this Reynolds number without any vortices.

The results of computational efficiency with respect to CPU times are presented next considering the error in lift coefficient. More specifically, CPU times reported are the average CPU time for each Newton–Raphson iteration and the reference value for the lift coefficient is obtained with a CG computation with the mesh shown in figure 15 and degree $k = 8$. In both HDG and CG, the non-linear solver takes around 5-6 Newton–Raphson iterations for attaining convergence with tolerance 10^{-12}.

Figure 17 shows the ratio of the CPU times for the direct solver *vs.* element size. The element size, h, in the case of NACA test is taken as the size of the

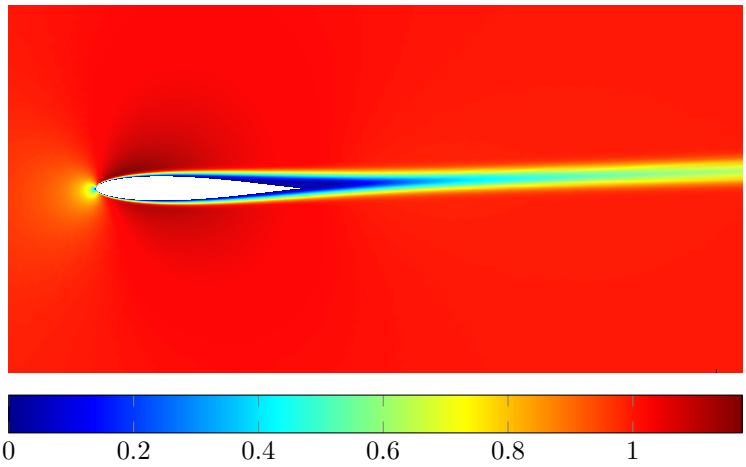


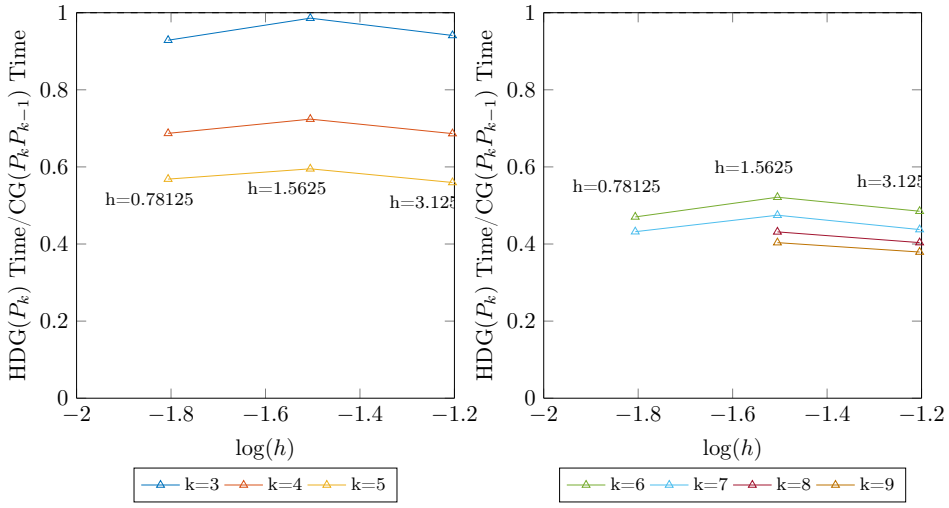
Figure 16: NACA airfoil: velocity field with $k = 5$ and refinement level 4.

biggest element in the mesh. All the ratios lie below 1, leading to the conclusion that HDG is more efficient than CG. Additionally, in this test, the ratios are nearly constant for each degree of approximation. The computational domain in this example is relatively big and hence, larger CPU times are noticed, which are more reliable and reproducible than for the Stokes problem for Kovasznay flow.

CPU time with error is plotted in figure 18 by interpolating the CPU times of HDG as stated in Kovasznay flow section. Again, all the ratios stay below 1, leading to similar conclusions: HDG is more computationally efficient than CG when CPU times for the direct solver are considered.

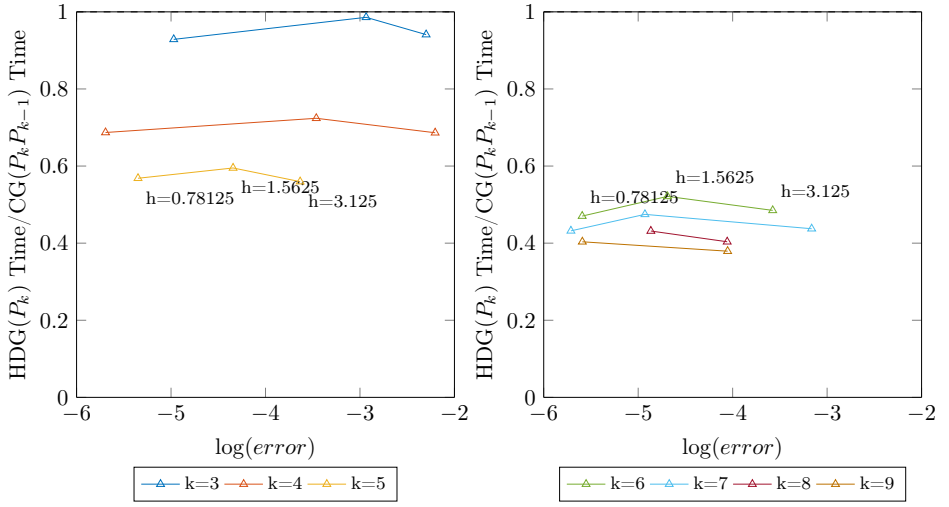
It is worth noting that the efficiency of HDG is in part due to the convergence of order $k + 1$ for the pressure and for the gradient, leading to convergence of order $k + 1$ for the lift coefficient, as compared to the order k of CG.

Finally, figure 19 shows the maximum condition number in Newton–Raphson iterations in HDG and CG. All the condition numbers presented are for triangular elements and $h = 0.78125$. On the one hand, in the case of HDG, condition number remains practically constant with increasing degree of approximation. On the other hand, in the case of CG, there is no particular trend observed,



(a) k = 3-5. (b) k = 6-9.

Figure 17: NACA airfoil: ratio of CPU times for direct solver *vs.* element size for triangular elements.



(a) k = 3-5. (b) k = 6-9.

Figure 18: NACA airfoil: ratio of CPU times for direct solver *vs.* error for triangular elements.

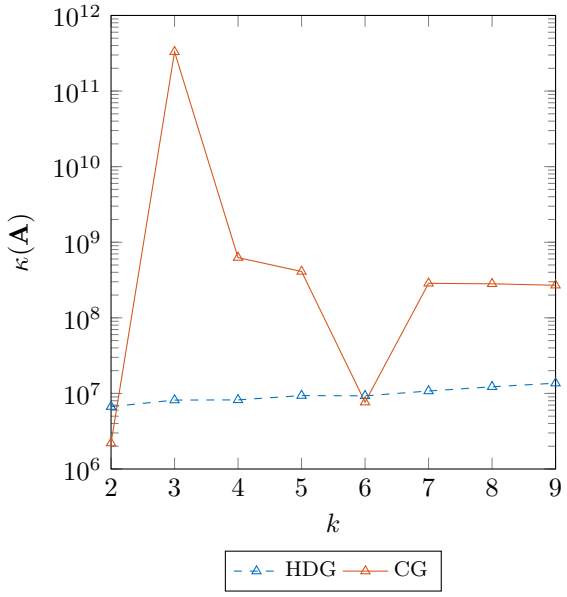


Figure 19: NACA airfoil: condition numbers with triangular elements and $h = 0.78125$.

except that for the cases $k = \{2, 6\}$, $\kappa(\mathbf{A})$ is several orders of magnitude higher, relatively to HDG.

7. Comparison of stability in the presence of sharp fronts

In this section, the robustness of HDG and CG is compared in terms of stability. The problem chosen to make this study was first reported in [50]. The analytical solution of this manufactured problem is given as follows,

$$\boldsymbol{u} = \begin{bmatrix} \left(1 - \cos\left(\dfrac{2\pi(\exp(R_1 x_1) - 1)}{\exp(R_1) - 1}\right)\right) \sin\left(\dfrac{2\pi(\exp(R_2 x_2) - 1)}{\exp(R_2) - 1}\right) \dfrac{R_2}{2\pi} \dfrac{\exp(R_2 x_2)}{(\exp(R_2) - 1)} \\ -\left(1 - \cos\left(\dfrac{2\pi(\exp(R_2 x_2) - 1)}{\exp(R_2) - 1}\right)\right) \sin\left(\dfrac{2\pi(\exp(R_1 x_1) - 1)}{\exp(R_1) - 1}\right) \dfrac{R_1}{2\pi} \dfrac{\exp(R_1 x_1)}{(\exp(R_1) - 1)} \end{bmatrix},$$
$$p = R_1 R_2 \sin\left(\frac{2\pi(\exp(R_2 x_2) - 1)}{\exp(R_2) - 1}\right) \sin\left(\frac{2\pi(\exp(R_1 x_1) - 1)}{\exp(R_1) - 1}\right) \frac{\exp(R_1 x_1)\exp(R_2 x_2)}{(\exp(R_1) - 1)(\exp(R_2) - 1)}, \tag{9}$$

where R_1 and R_2 are two positive parameters. The body force is the exact one corresponding to the solution of the Navier–Stokes equations (9). The problem

is solved in the computational domain $[0,1] \times [0,1]$.

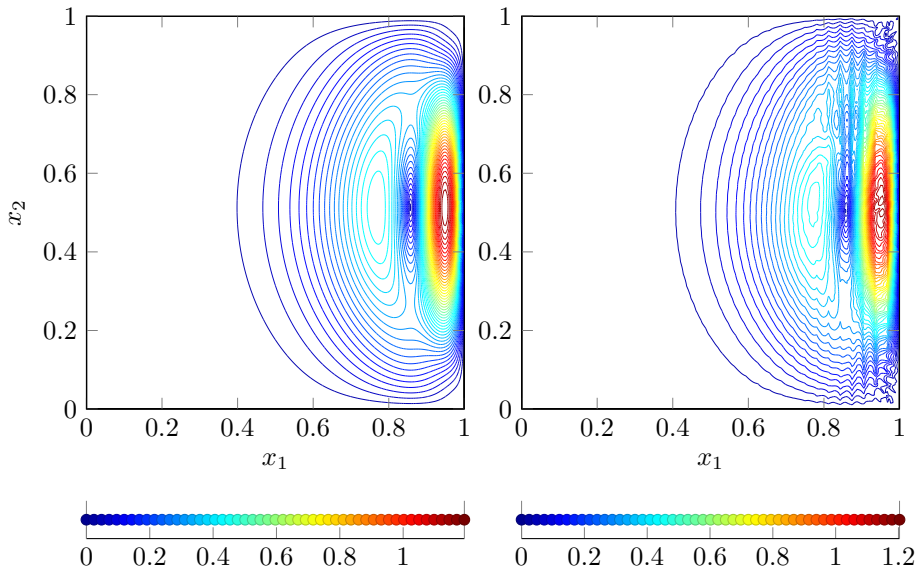
The velocity field of this solution is similar to a counter clockwise vortex. If (x_0, y_0) are the coordinates of the center of vortex, then the relation between R_1 and R_2 with (x_0, y_0) is

$$x_0 = \frac{\log((\exp(R_1)+1)/2)}{R_1}, \quad y_0 = \frac{\log((\exp(R_2)+1)/2)}{R_2}.$$

By increasing R_1, the center goes towards the right side ($x = 1$) of the domain, while by increasing R_2 the center approaches the top edge ($y = 1$) of the domain. In the present study, R_2 is fixed at 0.1 which gives $y_0 = 0.5125$. Hence, the center of vortex is on the line $y_0 = 0.5125$ and, in this case, its distance to the right side is $1/\sqrt[4]{Re}$, where $Re = 1/\nu$. R_1 is chosen in such a way that it satisfies the equation $1/R_1 \log((\exp(R_1)+1)/2) = 1 - (1/\sqrt[4]{Re})$. Thus, a boundary layer is formed near the right handed edge ($x = 1$) of the domain, which enables to study the relative stability between HDG and CG. Note that this solution is non-symmetric as the line of symmetry is $y = 0.5125$, which does not coincide with the line of symmetry of the domain. Hence, the numerical solutions obtained for the present problem are non-symmetric too.

To make the study, a regular mesh with triangular elements is chosen, with degree of approximation $k = 3$, on a mesh of $h = 0.03125$, which is relatively fine. The value of ν is chosen to be 4×10^{-4}, which corresponds to a Re of 2500. Dirichlet boundary conditions are applied on all the boundary, computed from the analytical solution, and also including the corresponding body force. In the case of HDG, the stabilization parameter, τ, is taken as 1. Steady state Navier–Stokes equations fail to converge for the mesh and problem data considered for both HDG and CG. Hence,the problem is solved using unsteady Navier–Stokes equations until a steady state solution is reached. BDF3 time integration scheme is used to perform the time integration. The initial solution is set as the analytical solution.

Figure 20 shows the solution for $Re = 2500$. HDG provides a smooth solution using $\tau = 1$ with a good resolution of boundary layer, whereas oscillations close



(a) HDG postprocessed solution of Navier–Stokes. $\|e_{\boldsymbol{u}^\star}\|_{\mathcal{L}_2} = 2.04 \times 10^{-4}$

(b) CG solution of Navier–Stokes.

Figure 20: Stability study: isolines of velocity field at $Re = 2500$ and $k = 3$.

to the boundary layer can be noticed in CG solution. The error obtained in the post-processed solution of HDG is 2.04×10^{-4}, which is reasonable compared to CG error of 1.09×10^{-2}. It is noticed that the post-processed solution of HDG is only slightly more accurate than the HDG solution. However, as expected super-convergence of the HDG is lost in this example.

8. Conclusions

Theoretical DOF count is compared between HDG, with degree k for all variables, and CG with degree k for the velocity and $k-1$ for the pressure. In the case of 2D, HDG has fewer DOFs than CG, when degree of approximation is more than 5, for both triangular and quadrilateral elements. In the case of 3D, they tend to be same only for very high degrees. However, when HDG with degree $k-1$ is compared to CG with degree k for velocity and $k-1$ for pressure, *i.e.*, with the same theoretical rates of convergence, the ratio of number of DOFs favours HDG when $k > 4$ for 2D and, in the case of 3D, $k > 6$ and $k > 12$ for hexahedral and tetrahedral elements, respectively. The number of non-zero entries in the global system and its factor are also provided for HDG and CG Stokes problems. HDG systems leads to fewer non zero entries when the degree of approximation, $k \geq 5$.

Then, Kovasznay flow, a benchmark for Stokes and Navier–Stokes problems, is considered to present the convergence rates of HDG and CG in 2D. A test case is designed to study the relative performance of HDG and CG using direct solvers. It is noticed that, using a high-order coarser mesh is computationally more efficient than using a low-order finer mesh, with respect to CPU time for direct solver. Numerical tests also show that HDG takes lesser CPU time for direct solver when compared to CG, for the same level of accuracy and for degree greater than 2.

The same comparison is carried out with NACA0012 airfoil example, with the error measured in the lift coefficient value. Again, HDG is more computationally efficient than CG for a given level of accuracy.

The condition numbers for HDG and CG are presented for Kovasznay flow and NACA airfoil examples. In most of the cases, HDG produces lower condition number values than CG.

Finally, a comparison of stability between HDG and CG is also presented, using a manufactured solution with a boundary layer. It is concluded that HDG can exhibit superior stability properties than CG in the presence of sharp fronts, which occurs at high Reynolds numbers.

Appendix A: Definition of elemental matrices in HDG and CG frameworks

In this appendix, the elemental matrices that arise from HDG and CG methods are defined. All the variables presented in this section are the elemental variables. Variable $\boldsymbol{L}^{(e)}$ is a second–order tensor and it is represented as a column vector, $[l_{11}\, l_{12}\, l_{21}\, l_{22}]^{(e)T}$, in the numerical computations.

The independent variables $(\boldsymbol{L}_h^{(e)}, \boldsymbol{u}_h^{(e)}, p_h^{(e)}, \hat{\boldsymbol{u}}_h^{(e)}, \rho_h^{(e)})$ over each element, Ω_e, can be approximated as follows,

$$\begin{aligned} \boldsymbol{L}_h^{(e)}(\boldsymbol{\xi}) = \boldsymbol{\psi}_L(\boldsymbol{\xi})\mathbf{L}^{(e)}, \quad \boldsymbol{u}_h^{(e)}(\boldsymbol{\xi}) = \boldsymbol{\psi}_u(\boldsymbol{\xi})\mathbf{u}^{(e)}, \quad p_h^{(e)}(\boldsymbol{\xi}) = \boldsymbol{\psi}_p(\boldsymbol{\xi})\mathbf{p}^{(e)} \qquad & \text{in} \quad \Omega_e, \\ \hat{\boldsymbol{u}}_h^{(e)}(\xi) = \boldsymbol{\psi}_{\hat{u}}(\xi)\hat{\mathbf{u}}^{(e)} \qquad & \text{on} \quad \partial\Omega_e, \end{aligned}$$

where $\boldsymbol{\psi}_L(\boldsymbol{\xi})$, $\boldsymbol{\psi}_u(\boldsymbol{\xi})$, $\boldsymbol{\psi}_p(\boldsymbol{\xi})$ and $\boldsymbol{\psi}_{\hat{u}}(\xi)$ are matrices that gather the approximation functions of respective unknowns, while $\mathbf{L}^{(e)}$, $\mathbf{u}^{(e)}$, $\mathbf{p}^{(e)}$ and $\hat{\mathbf{u}}^{(e)}$ are the elemental nodal column vectors of gradient of velocity, velocity, pressure and velocity trace, respectively. $\boldsymbol{\xi}$ and ξ represent the coordinate in the area and line reference domain, respectively. $\hat{\mathbf{u}}^{(e)}$ contains the trace of velocity on each face of the element and it can be represented as $\left[\hat{\mathbf{u}}^{\mathbf{F}_{e1}} \dots \hat{\mathbf{u}}^{\mathbf{F}_{en}}\right]^T$, where $\mathbf{F}_{ef}$ is the f^{th} face of e^{th} element. Here, $n = 3$ in the case of triangular elements, while $n = 4$ for quadrilateral elements. From now on explicit dependence on $\boldsymbol{\xi}$ and ξ is omitted for the sake of simplicity. The approximation functions can be

represented as follows,

$$\boldsymbol{\psi}_L = \begin{bmatrix} \boldsymbol{\psi} & & & \\ & \boldsymbol{\psi} & & \\ & & \boldsymbol{\psi} & \\ & & & \boldsymbol{\psi} \end{bmatrix}, \quad \boldsymbol{\psi}_u = \begin{bmatrix} \boldsymbol{\psi} & \\ & \boldsymbol{\psi} \end{bmatrix}, \quad \boldsymbol{\psi}_p = \boldsymbol{\psi},$$
$$\boldsymbol{\psi}_{\hat{u}} = \begin{bmatrix} \boldsymbol{\psi}_{\mathbf{F}_{e1}} & & \cdots & \boldsymbol{\psi}_{\mathbf{F}_{en}} \\ & \boldsymbol{\psi}_{\mathbf{F}_{e1}} & \cdots & & \boldsymbol{\psi}_{\mathbf{F}_{en}} \end{bmatrix},$$

where $\boldsymbol{\psi}$ is the matrix that gathers the shape functions associated to the nodes of the elements and $\boldsymbol{\psi}_{\mathbf{F}_{ef}}$ is the matrix collecting the shape functions associated to the nodes along the sides of the element.

Some notation used to represent the element matrices in case of both HDG and CG is given as follows,

$$\tilde{\boldsymbol{\nabla}} \equiv \begin{bmatrix} \partial/\partial x_1 & \partial/\partial x_2 & & \\ & & \partial/\partial x_1 & \partial/\partial x_2 \end{bmatrix}, \quad \tilde{\mathbf{N}} \equiv \begin{bmatrix} n_1 & n_2 & & \\ & & n_1 & n_2 \end{bmatrix}.$$

The global stiffness matrix, $\mathbf{K}$, and the elemental stiffness matrix, $\mathbf{K}^{(e)}$, are related by following expression, where $\mathbb{A}^{(e)}$ is the assembly operator.

$$\mathbf{K} = \mathbb{A}^{(e)}\mathbf{K}^{(e)}.$$

The elemental matrices of the corresponding global matrices presented in equation (3) are as follows,

$$\begin{aligned} \mathbf{K}^{(e)} &= \left(\left(\tilde{\boldsymbol{\nabla}}^T \boldsymbol{\psi}_u \right)^T, \nu \tilde{\boldsymbol{\nabla}}^T \boldsymbol{\psi}_u \right)_{\Omega_e}, \\ \mathbf{C}^{(e)}(\boldsymbol{u}) &= \left(\boldsymbol{\psi}_u^T \boldsymbol{u}, \tilde{\boldsymbol{\nabla}}^T \boldsymbol{\psi}_u \right)_{\Omega_e}, \\ \mathbf{G}^{(e)} &= -\left(\left(\boldsymbol{\nabla} \boldsymbol{\psi}_p \right)^T, \boldsymbol{\psi}_u \right)_{\Omega_e}, \\ \mathbf{f}^{(e)} &= \left(\boldsymbol{\psi}_u^T, \boldsymbol{f} \right)_{\Omega_e} + \langle \boldsymbol{\psi}_u^T, \boldsymbol{t} \rangle_{\partial \Omega_{eN}}. \end{aligned}$$

Similarly, the elemental matrices in the HDG framework presented in equa-

tion (6) are given. The matrices corresponding to equation (4a) are,

$$\mathbf{A}_{LL}^{(e)} = \left(\boldsymbol{\psi}_L^T, \boldsymbol{\psi}_L\right)_{\Omega_e}, \mathbf{A}_{Lu}^{(e)} = \left((\tilde{\boldsymbol{\nabla}}\boldsymbol{\psi}_L)^T, \boldsymbol{\psi}_u\right)_{\Omega_e},$$
$$\mathbf{A}_{L\hat{u}}^{(e)} = -\langle(\tilde{\mathbf{N}}\,\boldsymbol{\psi}_L)^T, \boldsymbol{\psi}_{\hat{u}}\rangle_{\partial\Omega_e}.$$

Similarly, non-linear convective matrices in equation (4b) can be expressed as follows,

$$\begin{aligned}\mathbf{C}_{uu}^{(e)}(\boldsymbol{u}) &= -\begin{bmatrix}\left(\boldsymbol{\psi}_{u,1}^T, u_1\boldsymbol{\psi}_u\right)_{\Omega_e} & \left(\boldsymbol{\psi}_{u,2}^T, u_1\boldsymbol{\psi}_u\right)_{\Omega_e}\\ \left(\boldsymbol{\psi}_{u,1}^T, u_2\boldsymbol{\psi}_u\right)_{\Omega_e} & \left(\boldsymbol{\psi}_{u,2}^T, u_2\boldsymbol{\psi}_u\right)_{\Omega_e}\end{bmatrix},\\ \mathbf{C}_{u\hat{u}}^{(e)}(\hat{\boldsymbol{u}}) &= \begin{bmatrix}\langle\boldsymbol{\psi}_u^T, \hat{u}_1\boldsymbol{\psi}_{\hat{u}}n_1\rangle_{\partial\Omega_e} & \langle\boldsymbol{\psi}_u^T, \hat{u}_1\boldsymbol{\psi}_{\hat{u}}n_2\rangle_{\partial\Omega_e}\\ \langle\boldsymbol{\psi}_u^T, \hat{u}_2\boldsymbol{\psi}_{\hat{u}}n_1\rangle_{\partial\Omega_e} & \langle\boldsymbol{\psi}_u^T, \hat{u}_2\boldsymbol{\psi}_{\hat{u}}n_2\rangle_{\partial\Omega_e}\end{bmatrix}.\end{aligned} \tag{A.1}$$

The discretization of the rest of the terms in equation (4b) results in the following,

$$\mathbf{A}_{uL}^{(e)} = -\left(\boldsymbol{\psi}_u^T, \nu\tilde{\boldsymbol{\nabla}}\boldsymbol{\psi}_L\right)_{\Omega_e}, \mathbf{A}_{up}^{(e)} = \left(\boldsymbol{\psi}_u^T, \boldsymbol{\nabla}\boldsymbol{\psi}_p\right)_{\Omega_e},$$
$$\mathbf{A}_{uu}^{(e)} = \langle\boldsymbol{\psi}_u^T, \tau\,\boldsymbol{\psi}_u\rangle_{\partial\Omega_e}, \mathbf{A}_{u\hat{u}}^{(e)} = -\langle\boldsymbol{\psi}_u^T, \tau\,\boldsymbol{\psi}_{\hat{u}}\rangle_{\partial\Omega_e},$$
$$\mathbf{f}^{(e)} = \left(\boldsymbol{\psi}_u^T, \boldsymbol{f}\right)_{\Omega_e}.$$

The matrices of the continuity equation (4c) and constraint (4d) are presented as follows,

$$\mathbf{A}_{pu}^{(e)} = -\left((\boldsymbol{\nabla}\boldsymbol{\psi}_p)^T, \boldsymbol{\psi}_u\right)_{\Omega_e}, \mathbf{A}_{p\hat{u}}^{(e)} = \langle(\boldsymbol{n}\,\boldsymbol{\psi}_p)^T, \boldsymbol{\psi}_{\hat{u}}\rangle_{\partial\Omega_e},$$
$$\mathbf{A}_{\rho p}^{(e)} = \tfrac{1}{|\partial\Omega_e|}\langle\boldsymbol{\psi}_p, 1\rangle_{\partial\Omega_e}.$$

Finally, the elemental matrices of global problem (5a) and (5b) can be expressed as,

$$\mathbf{A}_{\hat{u}p}^{(e)} = -\langle\boldsymbol{\psi}_{\hat{u}}^T, (\boldsymbol{n}\,\boldsymbol{\psi}_p)\rangle_{\partial\Omega_e}, \mathbf{A}_{\hat{u}L}^{(e)} = \langle\boldsymbol{\psi}_{\hat{u}}^T, (\tilde{\mathbf{N}}\,\boldsymbol{\psi}_L)\rangle_{\partial\Omega_e},$$
$$\mathbf{A}_{\hat{u}\hat{u}}^{(e)} = \langle\boldsymbol{\psi}_{\hat{u}}^T, \tau\,\boldsymbol{\psi}_{\hat{u}}\rangle_{\partial\Omega_e}, \mathbf{A}_{\hat{u}u}^{(e)} = -\langle\boldsymbol{\psi}_{\hat{u}}^T, \tau\,\boldsymbol{\psi}_u\rangle_{\partial\Omega_e},$$
$$\mathbf{A}_{\rho\hat{u}}^{(e)} = \langle\boldsymbol{n}^T\boldsymbol{\psi}_{\hat{u}}, 1\rangle_{\partial\Omega_e}, \mathbf{t}^{(e)} = \langle\boldsymbol{\psi}_{\hat{u}}^T, \boldsymbol{t}\rangle_{\partial\Omega_{eN}}.$$

The tangent operator terms, necessary within the Newton–Raphson iterative method, associated with the residual (6) are trivial, except for the ones associated with $\mathbf{C}_{uu}^{(e)}(\boldsymbol{u})$ and $\mathbf{C}_{u\hat{u}}^{(e)}(\hat{\boldsymbol{u}})$ matrices, given by (A.1). These are expressed

by

$$\mathbf{C}_{Tuu}^{(e)} = \begin{bmatrix} \left(\boldsymbol{\psi}_{u,1}^T, u_1 \boldsymbol{\psi}_u\right)_{\Omega_e} + \left(\boldsymbol{\psi}_{u,2}^T, u_2 \boldsymbol{\psi}_u\right)_{\Omega_e} & \mathbf{0} \\ \mathbf{0} & \left(\boldsymbol{\psi}_{u,1}^T, u_1 \boldsymbol{\psi}_u\right)_{\Omega_e} + \left(\boldsymbol{\psi}_{u,2}^T, u_2 \boldsymbol{\psi}_u\right)_{\Omega_e} \end{bmatrix},$$

$$\mathbf{C}_{Tu\hat{u}}^{(e)} = \begin{bmatrix} \langle \boldsymbol{\psi}_u^T, (\hat{\boldsymbol{u}} \cdot \boldsymbol{n}) \boldsymbol{\psi}_{\hat{u}} \rangle_{\partial\Omega_e} & \mathbf{0} \\ \mathbf{0} & \langle \boldsymbol{\psi}_u^T, (\hat{\boldsymbol{u}} \cdot \boldsymbol{n}) \boldsymbol{\psi}_{\hat{u}} \rangle_{\partial\Omega_e} \end{bmatrix}.$$

Acknowledgements

This work was supported by the Erasmus Mundus Joint Doctorate SEED project (European Commission, 2013-1436/001-001-EMJD). The second author also wishes to acknowledge the DAFOH2 project (Ministerio de Economia y Competitividad, MTM2013-46313-R) and the Catalan government (Generalitat de Catalunya, 2009SGR875). The work of the third author is part of the research activity carried out at Civil Engineering Research and Innovation for Sustainability (CERIS) and has been partially financed by Fundação para a Ciência e a Tecnologia (FCT) in the framework of project UID/ECI/04625/2013.

References

[1] R. Courant, Variational methods for the solution of problems of equilibrium and vibrations, Bulletin of the American Mathematical Society 49 (1) (1943) 1–23, also available in the Classic Reprints Series, International Journal for Numerical Methods in Engineering, volume 37, issue 13, pages 2159–2187, 1994.

[2] W. H. Reed, T. Hill, Triangular mesh methods for the neutron transport equation, Los Alamos Report LA-UR-73-479.

[3] F. Bassi, S. Rebay, A high-order accurate discontinuous finite element method for the numerical solution of the compressible Navier–Stokes equations, Journal of Computational Physics 131 (2) (1997) 267 – 279. doi:http://dx.doi.org/10.1006/jcph.1996.5572.

URL http://www.sciencedirect.com/science/article/pii/S0021999196955722

[4] D. N. Arnold, An interior penalty finite element method with discontinuous elements, SIAM Journal on Numerical Analysis 19 (4) (1982) 742–760. arXiv:http://dx.doi.org/10.1137/0719052, doi:10.1137/0719052.
URL http://dx.doi.org/10.1137/0719052

[5] B. Cockburn, C.-W. Shu, The local discontinuous Galerkin method for time-dependent convection-diffusion systems, SIAM Journal on Numerical Analysis 35 (6) (1998) 2440–2463. arXiv:http://dx.doi.org/10.1137/S0036142997316712, doi:10.1137/S0036142997316712.
URL http://dx.doi.org/10.1137/S0036142997316712

[6] J. Peraire, P.-O. Persson, The compact discontinuous Galerkin (CDG) method for elliptic problems, SIAM Journal on Scientific Computing 30 (4) (2008) 1806–1824. arXiv:http://dx.doi.org/10.1137/070685518, doi:10.1137/070685518.
URL http://dx.doi.org/10.1137/070685518

[7] P. Ciarlet, The Finite Element Method for Elliptic Problems, Society for Industrial and Applied Mathematics, 2002. arXiv:http://epubs.siam.org/doi/pdf/10.1137/1.9780898719208, doi:10.1137/1.9780898719208.
URL http://epubs.siam.org/doi/abs/10.1137/1.9780898719208

[8] B. Cockburn, J. Gopalakrishnan, R. Lazarov, Unified hybridization of discontinuous Galerkin, mixed, and continuous Galerkin methods for second order elliptic problems, SIAM Journal on Numerical Analysis 47 (2) (2009) 1319–1365. arXiv:http://dx.doi.org/10.1137/070706616, doi:10.1137/070706616.
URL http://dx.doi.org/10.1137/070706616

[9] N. Nguyen, J. Peraire, B. Cockburn, An implicit high-order hybridizable discontinuous Galerkin method for linear convection–diffusion equations, Journal of Computational Physics 228 (9) (2009) 3232 – 3254.

doi:http://dx.doi.org/10.1016/j.jcp.2009.01.030.
URL http://www.sciencedirect.com/science/article/pii/S0021999109000308

[10] B. Cockburn, K. Shi, Devising methods for Stokes flow: An overview, Computers & Fluids 98 (2014) 221 – 229, 12th USNCCM mini-symposium of High-Order Methods for Computational Fluid Dynamics - A special issue dedicated to the 80th birthday of Professor Antony Jameson. doi:http://dx.doi.org/10.1016/j.compfluid.2013.11.017.
URL http://www.sciencedirect.com/science/article/pii/S0045793013004568

[11] N. Nguyen, J. Peraire, B. Cockburn, A hybridizable discontinuous Galerkin method for Stokes flow, Computer Methods in Applied Mechanics and Engineering 199 (9–12) (2010) 582 – 597. doi:http://dx.doi.org/10.1016/j.cma.2009.10.007.
URL http://www.sciencedirect.com/science/article/pii/S0045782509003521

[12] N. Nguyen, J. Peraire, B. Cockburn, An implicit high-order hybridizable discontinuous Galerkin method for the incompressible Navier–Stokes equations, Journal of Computational Physics 230 (4) (2011) 1147 – 1170. doi:http://dx.doi.org/10.1016/j.jcp.2010.10.032.
URL http://www.sciencedirect.com/science/article/pii/S0021999110005887

[13] N. Nguyen, J. Peraire, B. Cockburn, Hybridizable discontinuous Galerkin methods for the time-harmonic Maxwell equations, Journal of Computational Physics 230 (19) (2011) 7151 – 7175. doi:http://dx.doi.org/10.1016/j.jcp.2011.05.018.
URL http://www.sciencedirect.com/science/article/pii/S0021999111003226

[14] N. Nguyen, J. Peraire, B. Cockburn, High-order implicit hybridizable discontinuous Galerkin methods for acoustics and elastodynamics, Journal of Computational Physics 230 (10) (2011) 3695 – 3718. doi:http://dx.doi.org/10.1016/j.jcp.2011.01.035.
URL http://www.sciencedirect.com/science/article/pii/S002199911100060X

[15] J. Peraire, N. C. Nguyen, B. Cockburn, A hybridizable discontinuous Galerkin method for the compressible Euler and Navier–Stokes equations (2010).

[16] C. Gürkan, E. Sala-Lardies, M. Kronbichler, S. Fernández-Méndez, eXtended hybridizable discontinous Galerkin (X-HDG) for void problems, Journal of Scientific Computing 66 (3) (2016) 1313–1333. doi:10.1007/s10915-015-0066-8.
URL http://dx.doi.org/10.1007/s10915-015-0066-8

[17] T. J. Hughes, G. Scovazzi, P. B. Bochev, A. Buffa, A multiscale discontinuous Galerkin method with the computational structure of a continuous Galerkin method, Computer Methods in Applied Mechanics and Engineering 195 (19-22) (2006) 2761 – 2787. doi:http://dx.doi.org/10.1016/j.cma.2005.06.006.
URL http://www.sciencedirect.com/science/article/pii/S0045782505002288

[18] S. Güzey, B. Cockburn, H. K. Stolarski, The embedded discontinuous Galerkin method: application to linear shell problems, International Journal for Numerical Methods in Engineering 70 (7) (2007) 757–790. doi:10.1002/nme.1893.
URL http://dx.doi.org/10.1002/nme.1893

[19] B. Cockburn, B. Dong, J. Guzman, M. Restelli, R. Sacco, A hybridizable discontinuous Galerkin method for steady-state convection-diffusion-reaction problems, SIAM Journal on Scientific Computing 31 (5) (2009)

3827–3846. arXiv:http://dx.doi.org/10.1137/080728810, doi:10.1137/080728810.
URL http://dx.doi.org/10.1137/080728810

[20] R. M. Kirby, S. J. Sherwin, B. Cockburn, To CG or to HDG: A comparative study, Journal of Scientific Computing 51 (1) (2012) 183–212. doi:10.1007/s10915-011-9501-7.
URL http://dx.doi.org/10.1007/s10915-011-9501-7

[21] S. Yakovlev, D. Moxey, R. M. Kirby, S. J. Sherwin, To CG or to HDG: A comparative study in 3D, Journal of Scientific Computing 67 (1) (2016) 192–220. doi:10.1007/s10915-015-0076-6.
URL http://dx.doi.org/10.1007/s10915-015-0076-6

[22] S. Rhebergen, B. Cockburn, J. J. van der Vegt, A space-time discontinuous Galerkin method for the incompressible Navier–Stokes equations, Journal of Computational Physics 233 (2013) 339 – 358. doi:http://dx.doi.org/10.1016/j.jcp.2012.08.052.
URL http://www.sciencedirect.com/science/article/pii/S0021999112005372

[23] X. Roca, C. Nguyen, J. Peraire, Scalable parallelization of the hybridized discontinuous Galerkin method for compressible flow, in: 21st AIAA Computational Fluid Dynamics Conference, American Institute of Aeronautics and Astronautics, 2013. doi:doi:10.2514/6.2013-2939.
URL http://dx.doi.org/10.2514/6.2013-2939

[24] A. Huerta, A. Angeloski, X. Roca, J. Peraire, Efficiency of high-order elements for continuous and discontinuous Galerkin methods, International Journal for Numerical Methods in Engineering 96 (9) (2013) 529–560. doi:10.1002/nme.4547.
URL http://dx.doi.org/10.1002/nme.4547

[25] S.-C. Soon, B. Cockburn, H. K. Stolarski, A hybridizable discontinuous Galerkin method for linear elasticity, International Journal for Numerical

Methods in Engineering 80 (8) (2009) 1058–1092. doi:10.1002/nme.2646.
URL http://dx.doi.org/10.1002/nme.2646

[26] G. Giorgiani, D. Modesto, S. Fernández-Méndez, A. Huerta, High-order continuous and discontinuous Galerkin methods for wave problems, International Journal for Numerical Methods in Fluids 73 (10) (2013) 883–903. doi:10.1002/fld.3828.
URL http://dx.doi.org/10.1002/fld.3828

[27] M. Woopen, A. Balan, G. May, J. Schütz, A comparison of hybridized and standard DG methods for target-based hp-adaptive simulation of compressible flow, Computers & Fluids 98 (2014) 3 – 16, 12th USNCCM mini-symposium of High-Order Methods for Computational Fluid Dynamics - A special issue dedicated to the 80th birthday of Professor Antony Jameson. doi:http://dx.doi.org/10.1016/j.compfluid.2014.03.023.
URL http://www.sciencedirect.com/science/article/pii/S0045793014001224

[28] C. Lehrenfeld, J. SchÃ¶berl, High order exactly divergence-free Hybrid Discontinuous Galerkin methods for unsteady incompressible flows, Computer Methods in Applied Mechanics and Engineering 307 (Supplement C) (2016) 339 – 361. doi:https://doi.org/10.1016/j.cma.2016.04.025.
URL http://www.sciencedirect.com/science/article/pii/S004578251630264X

[29] B. Cockburn, J. Gopalakrishnan, The derivation of hybridizable discontinuous Galerkin methods for Stokes flow, SIAM Journal on Numerical Analysis 47 (2) (2009) 1092–1125. arXiv:http://dx.doi.org/10.1137/080726653, doi:10.1137/080726653.
URL http://dx.doi.org/10.1137/080726653

[30] G. A. Holzapfel, Nonlinear Solid Mechanics: A Continuum Approach for Engineering, John Wiley & Sons, 2000.

[31] C. Taylor, P. Hood, A numerical solution of the Navier–Stokes equations using the finite element technique, Computers & Fluids 1 (1) (1973) 73–100.

[32] F. Brezzi, M. Fortin, Mixed and Hybrid Finite Element Methods, Springer Series in Computational Methematics, Springer-Verlag, New York, 1991.

[33] G. Giorgiani, S. Fernández-Méndez, A. Huerta, Hybridizable Discontinuous Galerkin with degree adaptivity for the incompressible Navier–Stokes equations, Computers & Fluids 98 (2014) 196 – 208, 12th USNCCM mini-symposium of High-Order Methods for Computational Fluid Dynamics - A special issue dedicated to the 80th birthday of Professor Antony Jameson. doi:http://dx.doi.org/10.1016/j.compfluid.2014.01.011.
URL //www.sciencedirect.com/science/article/pii/S0045793014000188

[34] N. Nguyen, J. Peraire, B. Cockburn, An implicit high-order hybridizable discontinuous Galerkin method for nonlinear convection–diffusion equations, Journal of Computational Physics 228 (23) (2009) 8841 – 8855. doi:http://dx.doi.org/10.1016/j.jcp.2009.08.030.
URL http://www.sciencedirect.com/science/article/pii/S0021999109004756

[35] B. Cockburn, J. Gopalakrishnan, N. C. Nguyen, J. Peraire, F.-J. Sayas, Analysis of HDG methods for Stokes flow., Math. Comput. 80 (274) (2011) 723–760.
URL http://dblp.uni-trier.de/db/journals/moc/moc80.html#CockburnGNPS11

[36] Y. Chen, B. Cockburn, Analysis of variable-degree HDG methods for convection–diffusion equations. Part I: general nonconforming meshes, IMA Journal of Numerical Analysis 32 (4) (2012) 1267–1293. arXiv:http://imajna.oxfordjournals.org/content/32/4/1267.full.pdf+html, doi:10.1093/imanum/drr058.

URL http://imajna.oxfordjournals.org/content/32/4/1267.abstract

[37] Y. Chen, B. Cockburn, Analysis of variable-degree HDG methods for convection–diffusion equations. Part II: Semimatching nonconforming meshes, Mathematics of Computation 83 (285) (2014) 87–111.
URL http://www.ams.org/journals/mcom/2014-83-285/S0025-5718-2013-02711-1;http://www.ams.org/journals/mcom/2014-83-285/S0025-5718-2013-02711-1/S0025-5718-2013-02711-1.pdf

[38] M. A. Taylor, B. A. Wingate, R. E. Vincent, An algorithm for computing Fekete points in the triangle, SIAM Journal on Numerical Analysis 38 (5) (2000) 1707–1720. arXiv:http://dx.doi.org/10.1137/S0036142998337247, doi:10.1137/S0036142998337247.
URL http://dx.doi.org/10.1137/S0036142998337247

[39] M. Abramowitz, Handbook of Mathematical Functions, With Formulas, Graphs, and Mathematical Tables,, Dover Publications, Incorporated, 1974.

[40] J. Hesthaven, T. Warburton, Nodal high-order methods on unstructured grids: I. time-domain solution of Maxwell's equations, Journal of Computational Physics 181 (1) (2002) 186 – 221. doi:http://dx.doi.org/10.1006/jcph.2002.7118.
URL http://www.sciencedirect.com/science/article/pii/S0021999102971184

[41] J. Sarrate, A. Huerta, Efficient unstructured quadrilateral mesh generation, International Journal for Numerical Methods in Engineering 49 (10) (2000) 1327–1350. doi:10.1002/1097-0207(20001210)49:10<1327::AID-NME996>3.0.CO;2-L.
URL http://dx.doi.org/10.1002/1097-0207(20001210)49:10<1327::AID-NME996>3.0.CO;2-L

[42] J. Sarrate, A. Huerta, Generación automática de mallas no estructuradas y formadas exclusivamente por cuadriláteros sobre superficies curvas en R^3, Revista Internacional de Métodos Numéricos para Cálculo y Diseño en Ingeniería 18 (1) (2002) 79–93.

[43] LaCaN, EZ4U. mesh generation environment (2016).
URL http://www.lacan.upc.edu/ez4u.htm

[44] C. Geuzaine, J.-F. Remacle, Gmsh: a three-dimensional finite element mesh generator with built-in pre- and post-processing facilities, International Journal for Numerical Methods in Engineering 79 (11) (2009) 1309–1331.

[45] HSL, A collection of Fortran codes for large scale scientific computation. (2016).
URL http://www.hsl.rl.ac.uk/

[46] M. Arioli, J. W. Demmel, I. S. Duff, Solving sparse linear systems with sparse backward error, SIAM Journal on Matrix Analysis and Applications 10 (2) (1989) 165–190. arXiv:http://dx.doi.org/10.1137/0610013, doi:10.1137/0610013.
URL http://dx.doi.org/10.1137/0610013

[47] P. R. Amestoy, T. A. Davis, I. S. Duff, An approximate minimum degree ordering algorithm, SIAM Journal on Matrix Analysis and Applications 17 (4) (1996) 886–905. arXiv:http://dx.doi.org/10.1137/S0895479894278952, doi:10.1137/S0895479894278952.
URL http://dx.doi.org/10.1137/S0895479894278952

[48] W. W. Hager, Condition estimates, SIAM Journal on Scientific and Statistical Computing 5 (2) (1984) 311–316. arXiv:http://dx.doi.org/10.1137/0905023, doi:10.1137/0905023.
URL http://dx.doi.org/10.1137/0905023

[49] L. I. G. Kovasznay, Laminar flow behind a two-dimensional grid, Mathematical Proceedings of the Cambridge Philosophical Society 44 (1948)

58–62. doi:10.1017/S0305004100023999.
URL http://journals.cambridge.org/article_S0305004100023999

[50] S. Berrone, Adaptive discretization of stationary and incompressible Navier–Stokes equations by stabilized finite element methods, Computer Methods in Applied Mechanics and Engineering 190 (34) (2001) 4435 – 4455. doi:http://dx.doi.org/10.1016/S0045-7825(00)00327-3.
URL http://www.sciencedirect.com/science/article/pii/S0045782500003273