



UNIVERSITÀ POLITECNICA DELLE MARCHE
Repository ISTITUZIONALE

Efficient discontinuous Galerkin implementations and preconditioners for implicit unsteady compressible flow simulations

This is the peer reviewed version of the following article:

Original

Efficient discontinuous Galerkin implementations and preconditioners for implicit unsteady compressible flow simulations / Franciolini, M.; Fidkowski, K. J.; Crivellini, A.. - In: COMPUTERS & FLUIDS. - ISSN 0045-7930. - 203:(2020). [10.1016/j.compfluid.2020.104542]

Availability:

This version is available at: 11566/282554 since: 2024-07-02T09:47:07Z

Publisher:

Published

DOI:10.1016/j.compfluid.2020.104542

Terms of use:

The terms and conditions for the reuse of this version of the manuscript are specified in the publishing policy. The use of copyrighted works requires the consent of the rights' holder (author or publisher). Works made available under a Creative Commons license or a Publisher's custom-made license can be used according to the terms and conditions contained therein. See editor's website for further information and terms and conditions.

This item was downloaded from IRIS Università Politecnica delle Marche (<https://iris.univpm.it>). When citing, please refer to the published version.

(Article begins on next page)

Efficient discontinuous Galerkin implementations and preconditioners for implicit unsteady compressible flow simulations

Matteo Franciolini^{a,b,c,*}, Krzysztof J. Fidkowski^a, Andrea Crivellini^b

^aDepartment of Aerospace Engineering, University of Michigan,
1320 Beal Ave, 48109 Ann Arbor (MI), United States

^bDepartment of Industrial Engineering and Mathematical Sciences,
Polytechnic University of Marche, Via Breccie Bianche 12, 60131 Ancona, Italy

^cNASA Ames Research Center, 94035 Moffett Field (CA), United States

Abstract

This work presents and compares efficient implementations of high-order discontinuous Galerkin methods: a modal matrix-free discontinuous Galerkin (DG) method, a hybridizable discontinuous Galerkin (HDG) method, and a *primal* formulation of HDG, applied to the implicit solution of unsteady compressible flows. The matrix-free implementation allows for a reduction of the memory footprint of the solver when dealing with implicit time-accurate discretizations. HDG reduces the number of globally-coupled degrees of freedom relative to DG, at high order, by statically condensing element-interior degrees of freedom from the system in favor of face unknowns. The *primal* formulation further reduces the element-interior degrees of freedom by eliminating the gradient as a separate unknown. This paper introduces a p -multigrid preconditioner implementation for these discretizations and presents results for various flow problems. Benefits of the p -multigrid strategy relative to simpler, less expensive, preconditioners are observed for stiff systems, such as those arising from low-Mach number flows at high-order approximation. The p -multigrid preconditioner also shows excellent scalability for parallel computations. Additional savings in both speed and memory occur with a matrix-free/reduced version of the preconditioner.

Keywords:

high-order, DG, HDG, p -multigrid, preconditioners, parallel efficiency

1. Introduction

In recent years, high-order discontinuous Galerkin (DG) methods have garnered attention in the field of Computational Fluid Dynamics. With increasing availability of high-performance computing (HPC) resources, the use of high-order methods for unsteady flow simulations has become popular. The success of these methods can be attributed to attractive dispersion and diffusion properties at high orders, ease of parallelization thanks to their compact stencil,

*Corresponding author

Email address: matteo.franciolini@nasa.gov (Matteo Franciolini)

6 and accuracy on unstructured meshes around complex geometries. However, the implementation of an efficient so-
7 lution strategy for high-order DG methods is still a subject of active research, especially for unsteady flow problems
8 involving the solution of the Navier–Stokes (NS) equations.

9 Several previous studies, see for example [1, 2], demonstrated that implicit schemes in the context of high-
10 order spatial discretizations are necessary to efficiently overcome the strict stability limits of explicit time integration
11 schemes [3]. Implicit strategies require the solution of a large system of equations, which is typically performed with
12 iterative solvers such as the generalized minimal residual method (GMRES). The choice of the preconditioner is a
13 key aspect of the strategy and has been explored extensively in the literature, see for example [4, 5, 6, 7]. Among
14 those, the use of multilevel algorithms to precondition a flexible GMRES solver [8] has been demonstrated as a
15 promising choice for both compressible [4, 9, 10, 6] and incompressible flow problems [11, 12]. Superior iterative
16 performance compared to single-grid preconditioned solvers has been observed, as well as better parallel efficiency
17 on distributed-memory architectures.

18 The application of implicit time integration strategies to DG discretizations is hindered by computational time and
19 memory expenses associated with the assembly and storage of the residual Jacobian matrix. Although the Jacobian is
20 sparse, the number of non-zero entries scales as k^{2d} , where k is the approximation order and d is the spatial dimension.
21 Thus, the costs grow rapidly with approximation order, particularly in three-dimensional problems. Motivated by
22 this scaling, previous works [13, 14, 15, 12] considered the possibility of a reduced matrix storage ("matrix-free")
23 implementation of the iterative solver. This implementation avoids the allocation of the Jacobian matrix but still
24 requires the allocation of a preconditioner operator which in some cases may still be quite large. In this context,
25 the use of multilevel matrix-free strategies with cheap element-wise block-Jacobi preconditioners on the finest level
26 appears to balance computational efficiency and memory considerations with iterative performance for stiff systems.
27 The latter is relevant to solvers applied to DG discretizations, for which the condition number scales as $O(h^{-2})$ [16],
28 where h is the mesh dimension.

29 The size of the DG linear system can be reduced through hybridizable discontinuous Galerkin (HDG) methods,
30 which have been recently considered as an alternative to the standard discontinuous Galerkin discretization [17, 18].
31 HDG methods introduce an additional trace variable on the mesh faces but can reduce the number of globally-coupled
32 degrees of freedom relative to DG, when a high order of polynomial approximation is employed. The reduction occurs
33 through a static condensation of the element-interior degrees of freedom, exploiting the block structure of the HDG
34 Jacobian matrix. Thanks to this operation, the memory footprint of the solver scales as $k^{2(d-1)}$. Additionally, HDG
35 methods exhibit superconvergence properties of the gradient variable in diffusion-dominated regimes. On the other
36 hand, they increase the number of operations local to each element, both before and after the linear system solution.
37 While several works have compared the accuracy and cost of HDG versus continuous [19, 20] and discontinuous [21,
38 18] Galerkin methods, a comparison considering the efficiency of iterative solvers applied to the solution of unsteady
39 flows is missing in this context. In fact, it is worth pointing out that, whereas HDG reduces the number of globally-
40 coupled degrees of freedom relative to DG, at high approximation orders, its element-local operation count is non-

41 trivial. This is particularly the case for viscous problems, in which the state gradient is approximated as a separate
42 variable in a *mixed*-type fashion. An alternative approach is to only approximate the state, and to obtain the gradient
43 when needed by differentiating the state. This leads to the *primal* HDG formulation [22, 23], which we also consider
44 in this work. The advantage of *primal* HDG relative to standard, *mixed* HDG lies mainly in the reduction of element-
45 local operations, which translates into improved computational performance.

46 While for DG the use of multilevel strategies to deal with ill-conditioned systems has been previously studied,
47 their use in HDG contexts appears not to have yet been explored, especially for unsteady flow problems. A multilevel
48 technique has been introduced in the context of an h -multigrid strategy built using the trace variable projection on a
49 continuous finite element space [16]. In addition, the use of an algebraic multigrid method applied to a linear finite
50 element space obtained by Galerkin projection has been proposed in the context of elliptic problems [24]. A similar
51 idea is also considered to speed-up the iterative solution process in HDG [25].

52 The present work focuses on the comparison of implicit solution strategies in the context of unsteady flow sim-
53 ulations for the three aforementioned implementations, *i.e.*, DG, *mixed* HDG, and *primal* HDG. The comparison
54 includes efficient preconditioning, such as p -multigrid, to deal with the solution of stiff linear systems arising from
55 high-order time discretizations. In particular, an efficient algorithm to inherit the coarse space operators at a low
56 computational cost in the context of HDG is presented for the first time. The scalability of the linear solution process
57 is also considered and compared to standard single-grid preconditioners, such as a incomplete lower-upper factoriza-
58 tion, ILU(0) [6]. The efficiency of the different solution strategies and the overall memory footprint is assessed on
59 two-dimensional laminar compressible flow problems. The results of these cases demonstrate that (i) the different
60 discretizations attain similar error levels, (ii) the use of a multilevel strategy reduces the number of linear iterations
61 in all cases tested, (iii) only for the DG discretizations is this advantage reflected in the CPU time, and iv) the *primal*
62 HDG and p -multigrid matrix-free DG solvers yield comparable solution times and memory footprints, faster than
63 standard, single-grid preconditioners like ILU(0) applied to DG.

64 The paper is structured as follows. Section 2 presents the differential equations, and Sections 3 and 4 show the
65 spatial and temporal discretizations used in this work. Section 6 presents the p -multigrid preconditioner implemen-
66 tations. Section 7 reports numerical experiments using different preconditioning strategies, including ILU(0), block
67 Jacobi, and p -multigrid. These experiments are performed on a range of test cases, including two-dimensional airfoils
68 and a circular cylinder. The methods are compared in terms of iterations, computational time, and memory footprint.
69 **To clarify our nomenclature, we report in Table 1 the meaning of each abbreviation that will be used throughout the**
70 **text.**

DG	standard discontinuous Galerkin discretization
MB	matrix-based (F)GMRES
MF	matrix-free (F)GMRES
MFL	matrix-free (F)GMRES with preconditioner lagging
mHDG	mixed hybridizable discontinuous Galerkin discretization
pHDG	primal hybridizable discontinuous Galerkin discretization
BJ	element-by-element block Jacobi preconditioner
BILU	partition-by-partition block ILU(0) preconditioner

Table 1: Nomenclature used throughout the text.

71 2. Governing equations

72 This work considers solutions of the compressible Navier–Stokes (NS) equations, which can be written as

$$\begin{aligned}
\frac{\partial \rho}{\partial t} + \nabla \cdot (\rho \vec{v}) &= 0, \\
\frac{\partial}{\partial t}(\rho \vec{v}) + \nabla \cdot (\rho \vec{v} \otimes \vec{v} + p \mathbf{I}) &= \nabla \cdot \boldsymbol{\tau}, \\
\frac{\partial}{\partial t}(\rho e_0) + \nabla \cdot (\rho \vec{v} h_0) &= \nabla \cdot (\vec{v} \cdot \boldsymbol{\tau} - \vec{q}),
\end{aligned} \tag{1}$$

73 with $\vec{v} \in \mathbb{R}^d$ the velocity, and d the number of space dimensions. The total energy e_0 , total enthalpy h_0 , pressure p ,
74 total stress tensor $\boldsymbol{\tau}$, and heat flux \vec{q} are given by

$$\begin{aligned}
e_0 &= e + (\vec{v} \cdot \vec{v})/2, & h_0 &= e_0 + p/\rho, & p &= (\gamma - 1)\rho e, \\
\boldsymbol{\tau} &= 2\mu \left(\mathbf{S} - \frac{1}{3}(\nabla \cdot \vec{v}) \mathbf{I} \right), & \vec{q} &= -\frac{\mu}{\text{Pr}} \nabla h, & \mathbf{S} &= \frac{1}{2}(\nabla \vec{v} + (\nabla \vec{v})^T).
\end{aligned}$$

76 Here e is the internal energy, h is the enthalpy, $\gamma = c_p/c_v$ is the ratio of gas specific heats, μ is the viscosity, Pr is the
77 molecular Prandtl number, and \mathbf{S} is the mean strain-rate tensor. The space and time discretizations are outlined in the
78 following sections.

79 3. Spatial discretization

80 Three versions of a modal discontinuous Galerkin (DG) finite element method are considered in this work. The
81 first one is a standard DG implementation, which employs basis functions defined in the reference element space.
82 The second one, commonly referred to as the hybridizable discontinuous Galerkin (HDG) method, introduces an
83 additional set of variables defined on the mesh element interfaces to reduce the globally coupled degrees of freedom
84 compared to DG, as shown in Figure 1. This implementation explicitly uses a mixed form for the gradient states
85 (also known as the *dual variable*), *i.e.* the gradients are used as an additional element-wise variable, and increase the
86 accuracy of the gradient evaluation. A third implementation, *primal* HDG [22], reduces the computational costs of the
87 solver by removing the dual variable from the equations. This approach is desirable when an increased accuracy of

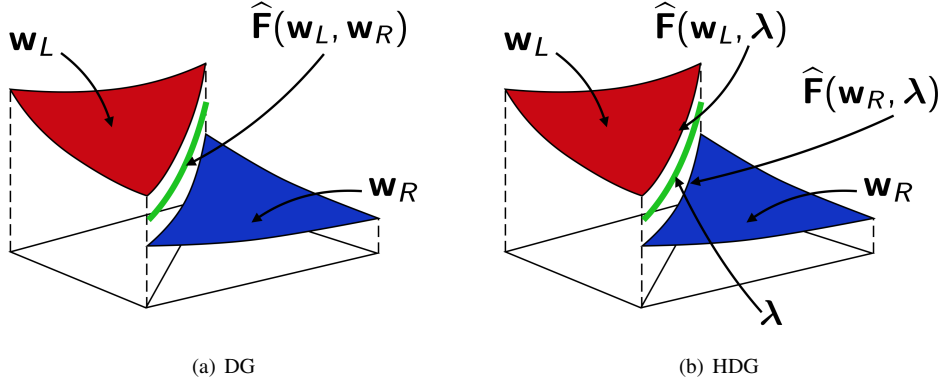


Figure 1: Schematic comparison of the solution approximation and flux evaluation in DG versus HDG.

88 the gradient variable is not strictly necessary or achievable for the problem. In all cases, two types of basis functions
 89 in the reference space have been considered: polynomial functions of maximum degree equal to k as well as tensor
 90 product functions of degree k in each dimension. In this work the former is employed within triangular grids, and
 91 the number of degrees of freedom per equation per element is $n_v^\ell = \prod_{i=1}^d (k+i)/i$. The latter approach is used for
 92 quadrilateral mesh elements, and $n_v = (k+1)^d$.

93 In all cases, the discretization is based on an approximation Ω_h of the domain Ω and a triangulation $\mathcal{T}_h = \{K\}$ of
 94 Ω_h made by a set of n_e non-overlapping elements, denoted by K . Here, \mathcal{F}_h^i stands for the set of internal element faces,
 95 \mathcal{F}_h^b the set of boundary element faces and $\mathcal{F}_h = \mathcal{F}_h^i \cup \mathcal{F}_h^b$ their union. We also define

$$\Gamma_h^i = \bigcup_{F \in \mathcal{F}_h^i} F, \quad \Gamma_h^b = \bigcup_{F \in \mathcal{F}_h^b} F, \quad \Gamma_h = \Gamma_h^i \cup \Gamma_h^b. \quad (2)$$

96 where F denotes a generic mesh element face. Following Brezzi et al. [26], we also introduce the average trace
 97 operator, which on a generic internal face $F \in \mathcal{F}_h^i$ is defined as $\{\cdot\} \stackrel{\text{def}}{=} \frac{(\cdot)^+ + (\cdot)^-}{2}$, where (\cdot) denotes a generic scalar
 98 or vector quantity. This definitions can be suitably extended to domain boundary faces by accounting for the weak
 99 imposition of boundary conditions.

100 3.1. Discontinuous Galerkin

101 In compact form, the system (1) can be expressed as

$$\frac{\partial \mathbf{u}}{\partial t} + \nabla \cdot \vec{\mathbf{F}}_c(\mathbf{u}) + \nabla \cdot \vec{\mathbf{F}}_v(\mathbf{u}, \nabla \mathbf{u}) = \mathbf{0}, \quad (3)$$

102 where $\mathbf{u} \in \mathbb{R}^m$ is the vector of conservative variables and $\vec{\mathbf{F}}_c, \vec{\mathbf{F}}_v \in \mathbb{R}^{m \times d}$ are the inviscid and viscous fluxes, with m
 103 the number of equations and d the number of space dimensions. Note that the diffusive fluxes are connected to the
 104 gradients of \mathbf{u} via the functional relation $\vec{\mathbf{F}}_d = -\mathbf{K} \nabla \mathbf{u}$, with \mathbf{K} the diffusivity tensor.

105 The state vector is approximated by a polynomial expansion with no continuity constraints imposed between
 106 adjacent elements, *i.e.* $\mathbf{u}_h \in [\mathcal{V}_h]^m$ where

$$\mathcal{V}_h = \{\phi_h \in L_2(\Omega) : \phi_h|_K \in \mathbb{P}_k, \forall K \in \mathcal{T}_h\}, \quad (4)$$

107 and k is the order of polynomial approximation. The weak form of (3) follows from multiplying the PDE by the set
 108 of test functions in the same approximation space, integrating by parts, and coupling elements via numerical fluxes.
 109 The variational formulation for each element $K \in \mathcal{T}_h$ reads: find $\mathbf{u}_h \in [\mathcal{V}_h]^m$ such that

$$\begin{aligned} & \int_K \mathbf{w}_h \cdot \frac{\partial \mathbf{u}_h}{\partial t} \, dK - \int_K \nabla_h \mathbf{w}_h : \vec{\mathbf{F}}(\mathbf{u}_h, \nabla_h \mathbf{u}_h) \, dK \\ & + \int_{\partial K} \mathbf{w}_h^+ \cdot \widehat{\mathbf{F}}(\mathbf{u}_h^\pm, \nabla_h \mathbf{u}_h^\pm, \vec{n}^+) \, d\sigma - \int_{\partial K} \nabla \mathbf{w}_h^+ : (\mathbf{K}^+ \cdot \vec{n}^+) (\mathbf{u}_h^+ - \hat{\mathbf{u}}_h) \, d\sigma = 0, \end{aligned} \quad (5)$$

110 for all $\mathbf{w}_h \in [\mathcal{V}_h]^m$. Here, $\vec{\mathbf{F}}$ is the sum of the inviscid and viscous flux functions, while $\widehat{\mathbf{F}}$ is the numerical flux and
 111 $\hat{\mathbf{u}}_h = (\mathbf{u}_h^+ + \mathbf{u}_h^-)/2$. Note that the quantities $(\cdot)^+$ and $(\cdot)^-$ denote element interior and element neighbor quantities,
 112 respectively.

113 Uniqueness and local conservation of the solution are achieved by the use of proper numerical interface fluxes.
 114 The Roe [27] approximate Riemann solver is employed for the inviscid part $\widehat{\mathbf{F}}_c$, while the second form of Bassi and
 115 Rebay (BR2) [28] is employed for the viscous part, $\widehat{\mathbf{F}}_v$. Following BR2, the numerical viscous flux is given by

$$\widehat{\mathbf{F}}_v(\mathbf{u}_h^\pm, \nabla_h \mathbf{u}_h^\pm, \vec{n}^+) \stackrel{\text{def}}{=} \{\widehat{\mathbf{F}}_v(\mathbf{u}_h, \nabla_h \mathbf{u}_h)\} \cdot \vec{n} + \eta_F \{\delta_F(\mathbf{u}_h^+ - \mathbf{u}_h^-)\} \cdot \vec{n}^+, \quad (6)$$

116 where, according to [29, 26], the penalty factor η_F must be greater than the number of faces of the elements. The
 117 auxiliary variable δ_F is determined from the jump of \mathbf{u}_h , via the solution of the following auxiliary problem:

$$\int_K \vec{\tau}_h^+ : \delta_F^+ \, dK = \frac{1}{2} \int_F \vec{\tau}_h^+ : (\mathbf{K}^+ \cdot \vec{n}^+) (\mathbf{u}_h^+ - \mathbf{u}_h^-) \, d\sigma, \quad \forall \tau_h \in [\mathcal{V}_h]^{d \times m}. \quad (7)$$

118 At the boundary of the domain, the numerical flux function appearing in equation (5) must be consistent with the
 119 boundary conditions of the problem. In practice, this is accomplished by properly defining a boundary state which
 120 accounts for the boundary data and, together with the internal state, allows for the computation of the numerical fluxes
 121 and the lifting operator on the portion Γ_h^b of the boundary Γ_h , see [28, 30].

122 A system of ordinary differential equations for the degrees of freedom (DoFs) arising from Equation (5) can be
 123 compactly written in the form

$$\mathbf{M} \frac{d\mathbf{W}}{dt} + \mathbf{R}(\mathbf{W}) = \mathbf{0}, \quad (8)$$

124 where \mathbf{M} is the block-diagonal spatial mass matrix, \mathbf{W} is the vector of the DoFs of the problem, and \mathbf{R} is the spatial
 125 residual vector.

126 3.2. Hybridizable discontinuous Galerkin

127 3.2.1. Mixed form

128 The second spatial discretization considered in this work is the HDG method in mixed form (m HDG), see [18]. A
 129 system of first-order partial differential equations can be obtained from (1) by introducing $\vec{\mathbf{q}} \in \mathbb{R}^{m \times d}$,

$$\begin{aligned} \vec{\mathbf{q}} - \nabla \mathbf{u} &= \vec{\mathbf{0}}, \\ \frac{\partial \mathbf{u}}{\partial t} + \nabla \cdot \mathbf{F}_c(\mathbf{u}) + \nabla \cdot \mathbf{F}_v(\mathbf{u}, \vec{\mathbf{q}}) &= \mathbf{0}. \end{aligned} \quad (9)$$

130 The HDG discretization approximates the state and gradient variables as $\mathbf{u}_h \in [\mathcal{V}_h]^m$ and $\vec{\mathbf{q}}_h \in [\mathcal{V}_h]^{m \times d}$, with \mathcal{V}_h
 131 defined in (4). An additional trace variable, $\lambda_h \in [\mathcal{M}_h]^m$, is defined on the faces, using the space

$$\mathcal{M}_h = \left\{ \mu \in L_2(\mathcal{F}_h^i) : \mu|_F \in \mathbb{P}_k, \forall F \in \mathcal{F}_h^i \right\}, \quad (10)$$

132 where \mathbb{P}_k is the space of polynomials of order k on face F . We remark that the trace variable is defined on the internal
 133 faces only, while a properly defined boundary value is used for the flux computation on \mathcal{F}_h^b .

134 The weak form is obtained by weighting the equations in (9) with appropriate test functions, integrating by parts,
 135 and using the interface variable λ_h for the face state. Consistent and stable numerical fluxes are required at the mesh
 136 element interfaces. The variational formulation reads: find $\mathbf{u}_h \in [\mathcal{V}_h]^m$, $\vec{\mathbf{q}}_h \in [\mathcal{V}_h]^{m \times d}$, $\lambda_h \in [\mathcal{M}_h]^m$, such that

$$\int_K \vec{\mathbf{v}}_h : \vec{\mathbf{q}}_h \, dK + \int_K (\nabla_h \cdot \mathbf{v}_h) \cdot \mathbf{u}_h \, dK - \int_{\partial K} (\vec{\mathbf{v}}_h^+ \cdot \vec{\mathbf{n}}^+) \cdot \lambda_h \, d\sigma = \mathbf{0}, \quad (11)$$

$$\int_K \mathbf{w}_h \cdot \frac{\partial \mathbf{u}_h}{\partial t} \, dK - \int_K \nabla_h \mathbf{w}_h : \vec{\mathbf{F}}(\mathbf{u}_h, \vec{\mathbf{q}}_h) \, dK + \int_{\partial K} \mathbf{w}_h^+ \cdot \widehat{\mathbf{F}}(\mathbf{u}_h^+, \vec{\mathbf{q}}_h^+, \lambda_h, \vec{\mathbf{n}}^+) \, d\sigma = \mathbf{0}, \quad (12)$$

$$\int_{\partial K} \boldsymbol{\mu}_h \cdot \left\{ \widehat{\mathbf{F}}(\mathbf{u}_h^+, \vec{\mathbf{q}}_h^+, \lambda_h, \vec{\mathbf{n}}^+) + \widehat{\mathbf{F}}(\mathbf{u}_h^-, \vec{\mathbf{q}}_h^-, \lambda_h, \vec{\mathbf{n}}^-) \right\} \, d\sigma = \mathbf{0}, \quad (13)$$

139 for all $\mathbf{w}_h \in [\mathcal{V}_h]^m$, $\vec{\mathbf{v}}_h \in [\mathcal{V}_h]^{m \times d}$, $\boldsymbol{\mu}_h \in [\mathcal{M}_h]^m$. The third equation, which weakly imposes flux continuity across
 140 interior faces, is required to close the system. We remark that, when using the mixed form, the same theoretical
 141 convergence rate is observed for the state variable \mathbf{u}_h and the gradient variable $\vec{\mathbf{q}}_h$. In diffusion-dominated regimes,
 142 this allows for a local post-processing of the state to a higher order [17].

143 In HDG, the numerical flux function $\widehat{\mathbf{F}}$, which is the sum of the inviscid and viscous fluxes, is defined as

$$\widehat{\mathbf{F}}(\mathbf{u}_h, \vec{\mathbf{q}}_h, \lambda_h, \vec{\mathbf{n}}) = \vec{\mathbf{F}}(\lambda_h, \vec{\mathbf{q}}_h) \cdot \vec{\mathbf{n}} + \boldsymbol{\tau}(\lambda_h, \mathbf{u}_h, \vec{\mathbf{n}}), \quad (14)$$

144 where $\boldsymbol{\tau} = \boldsymbol{\tau}_c + \boldsymbol{\tau}_v$ is a stabilization term for both the inviscid and viscous parts of the flux. In this work $\boldsymbol{\tau}_c$ is chosen
 145 in a Roe-like fashion as

$$\boldsymbol{\tau}_c = \left| \vec{\mathbf{F}}'_c(\lambda_h) \cdot \vec{\mathbf{n}} \right| (\mathbf{u}_h - \lambda_h), \quad (15)$$

146 while the viscous stabilization term $\boldsymbol{\tau}_v$ is based on the BR2 scheme,

$$\boldsymbol{\tau}_v = \eta_F \vec{\boldsymbol{\delta}}_F(\mathbf{u}_h - \lambda_h) \cdot \vec{\mathbf{n}}, \quad (16)$$

147 with $\vec{\boldsymbol{\delta}}_F$ the lifting operator applied to the jump $(\mathbf{u}_h - \lambda_h)$, and η_F the stabilization factor.

148 Similarly to DG, at the boundary of the domain, the numerical flux function is made consistent with the boundary
 149 conditions of the problem through the definition of a boundary state which accounts for the boundary data and,

150 together with the internal state, allows for the computation of numerical fluxes and the lifting operator on the portion
 151 Γ_h^b of the boundary Γ_h .

152 Defining \mathbf{R}^Q , \mathbf{R}^U and \mathbf{R}^Λ as the residual vectors arising from Equations (11), (12) and (13), the discretized system
 153 of nonlinear equations can be written as

$$\begin{aligned} \mathbf{R}^Q &= \mathbf{0}, \\ \mathbf{M}^U \frac{d\mathbf{U}}{dt} + \mathbf{R}^U &= \mathbf{0}, \\ \mathbf{R}^\Lambda &= \mathbf{0}. \end{aligned} \quad (17)$$

154 where \mathbf{M}^U is the element-based mass matrix. The compact form of (17) can be written using the solution vector of
 155 the discrete unknowns, $\mathbf{W} = [\mathbf{Q}; \mathbf{U}; \mathbf{\Lambda}]$, and the concatenated vector of residuals $\mathbf{R} = [\mathbf{R}^Q; \mathbf{R}^U; \mathbf{R}^\Lambda]$,

$$\mathbf{M} \frac{d\mathbf{W}}{dt} + \mathbf{R}(\mathbf{W}) = \mathbf{0}, \quad (18)$$

156 where the matrix \mathbf{M} is given by

$$\mathbf{M} = \begin{bmatrix} \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{M}^U & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} \end{bmatrix}. \quad (19)$$

157 3.2.2. Primal form

158 A variant of the mixed hybridizable discontinuous Galerkin method presented in Section 3.2.1 is the *primal* HDG
 159 (*p*HDG) method and follows the work in [22, 23]. In *p*HDG, the dual variable is eliminated by introducing the
 160 definition of the gradient in (12).

161 The *p*HDG discretization approximates the variable $\mathbf{u}_h \in [\mathcal{V}_h]^m$, with \mathcal{V}_h defined in Equation (4). The trace
 162 variable $\lambda_h \in [\mathcal{M}_h]^m$ is still employed for hybridization, and the variational formulation reads: find $\mathbf{u}_h \in [\mathcal{V}_h]^m$,
 163 $\lambda_h \in [\mathcal{M}_h]^m$ such that

$$\begin{aligned} & \int_K \mathbf{w}_h \cdot \frac{\partial \mathbf{u}_h}{\partial t} dK - \int_K \nabla_h \mathbf{w}_h : \vec{\mathbf{F}}(\mathbf{u}_h, \nabla_h \mathbf{u}_h) dK \\ & + \int_{\partial K} \mathbf{w}_h^+ \cdot \widehat{\mathbf{F}}(\mathbf{u}_h^+, \nabla_h \mathbf{u}_h^+, \lambda_h, \vec{n}^+) d\sigma - \int_{\partial K} \nabla \mathbf{w}_h^+ : (\mathbf{K}^+ \cdot \vec{n}^+) (\mathbf{u}_h^+ - \hat{\mathbf{u}}_h) d\sigma = \mathbf{0}, \end{aligned} \quad (20)$$

$$\int_{\partial K} \boldsymbol{\mu}_h \cdot \left\{ \widehat{\mathbf{F}}(\mathbf{u}_h^+, \vec{\mathbf{q}}_h^+, \lambda_h, \vec{n}^+) + \widehat{\mathbf{F}}(\mathbf{u}_h^-, \vec{\mathbf{q}}_h^-, \lambda_h, \vec{n}^-) \right\} d\sigma = \mathbf{0}, \quad (21)$$

165 for all $\mathbf{w}_h \in [\mathcal{V}_h]^m$, $\boldsymbol{\mu}_h \in [\mathcal{M}_h]^m$. We note that (20)–(21) are not obtained by just substituting $\vec{\mathbf{q}}_h = \nabla \mathbf{u}_h$ from (11)–
 166 (13). In fact the fourth term of (20) arises from the elimination of the variable $\vec{\mathbf{q}}_h$. This term ensures symmetry
 167 and adjoint-consistency of the *primal* HDG discretization. This type of discretization, involving a smaller number of
 168 element-wise degrees of freedom than mixed HDG, does not suffer significantly from overhead costs of dealing with
 169 the gradients: eliminating the dual variable and adding the adjoint-consistency term typically results in a faster solver.

170 We note that in this case, the gradients are of one order lower accuracy than the state variable \mathbf{u}_h . Numerical flux
 171 functions, stabilizing terms, and boundary condition enforcement are defined in the same manner as in mixed HDG.

172 Defining \mathbf{R}^U and \mathbf{R}^Λ as the residuals vectors arising from (20)–(21), the ODE system of equations can be written
 173 as

$$\begin{aligned} \mathbf{M}^U \frac{d\mathbf{U}}{dt} + \mathbf{R}^U &= \mathbf{0}, \\ \mathbf{R}^\Lambda &= \mathbf{0}, \end{aligned} \quad (22)$$

174 where \mathbf{M}^U is the elemental mass matrix. Therefore, the compact form of (22) is written using the solution vector of
 175 discrete unknowns, $\mathbf{W} = [\mathbf{U}; \Lambda]$, and the concatenated vector of residuals, $\mathbf{R} = [\mathbf{R}^U; \mathbf{R}^\Lambda]$,

$$\mathbf{M} \frac{d\mathbf{W}}{dt} + \mathbf{R}(\mathbf{W}) = \mathbf{0}, \quad (23)$$

176 where the matrix \mathbf{M} is given by

$$\mathbf{M} = \begin{bmatrix} \mathbf{M}^U & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{bmatrix}. \quad (24)$$

177 4. Temporal discretization

178 The temporal discretization used in this work is an explicit-first stage, singly-diagonal-implicit Runge–Kutta (ES-
 179 DIRK) scheme. The general formulation of the scheme for (18) is

$$\begin{aligned} \mathbf{M}\mathbf{W}^i &= \mathbf{M}\mathbf{W}^n - \Delta t \sum_{j=1}^i a_{ij} \mathbf{R}(\mathbf{W}^j), \\ \mathbf{W}^{n+1} &= \mathbf{W}^n + \Delta t \sum_{i=1}^s \beta_i \mathbf{W}^i, \end{aligned} \quad (25)$$

180 for $i = 1, \dots, s$ where s is the number of stages, a_{ij} and b_i are the coefficients of the scheme, and n is the time index.
 181 Within each stage, the solution of a non-linear system is required. This is performed by the Newton–Krylov method,
 182 which requires the solution of a sequence of linear systems within each stage. In this regard, the k^{th} Newton–Krylov
 183 iteration assumes the form

$$\left(\frac{\mathbf{M}}{a_{ii}\Delta t} + \frac{\partial \mathbf{R}}{\partial \mathbf{W}} \right) (\mathbf{W}_{k+1}^i - \mathbf{W}_k^i) = -\frac{\mathbf{M}}{a_{ii}\Delta t} (\mathbf{W}_k^i - \mathbf{W}^n) - \sum_{j=1}^{i-1} \frac{a_{ij}}{a_{ii}} \mathbf{R}(\mathbf{W}^j) - \mathbf{R}(\mathbf{W}_k^i), \quad (26)$$

184 with $i = 1, \dots, s$. In this work the third-order ESDIRK3 scheme [31] is employed. The method involve three non-linear
 185 solutions, following an explicit first stage.

186 5. Linear system solution

187 The linear system of ODEs can be solved numerically using iterative solvers. To this end, we apply the generalized
 188 minimal residual (GMRES) method to the system in (26). The solution process differs between standard DG and
 189 HDG, as the latter discretization takes advantage of the introduction of face unknowns in order to reduce the size of
 190 the matrix to be allocated. This section provides details of the implementation.

191 *5.1. Discontinuous Galerkin discretization*

192 The linear system arising from a DG discretizations takes the following general form

$$\mathcal{K}\mathbf{x} + \mathbf{b} = \mathbf{0}, \quad (27)$$

193 where $\mathcal{K} = (\mathbf{M}/(a_{ii}\Delta t) + \partial\mathbf{R}/\partial\mathbf{W})$ is the iteration matrix, $\mathbf{x} = \Delta\mathbf{W}$ is the vector of degrees of freedom updates, and \mathbf{b}
 194 is the right-hand side. The GMRES implementation can follow two approaches. The first one is denoted as matrix-
 195 based (MB) and consists of computing and storing the iteration matrix explicitly to perform matrix-vector products
 196 as needed within the iterative solution. A second, matrix-free (MF) approach takes advantage of the structure of the
 197 matrix vector products, which can be approximated using the matrix-free formula

$$\left(\frac{\mathbf{M}}{a_{ii}\Delta t} + \frac{\partial\mathbf{R}}{\partial\mathbf{W}} \right) \Delta\mathbf{W} \approx \frac{\mathbf{M}}{a_{ii}\Delta t} \Delta\mathbf{W} + \frac{\mathbf{R}(\mathbf{W} + h\Delta\mathbf{W}) - \mathbf{R}(\mathbf{W})}{h}, \quad (28)$$

198 with

$$h = \varepsilon \frac{\sqrt{1 + \|\Delta\mathbf{W}\|}}{\|\mathbf{W}\|} \quad (29)$$

199 and $\varepsilon \approx 10^{-7}$. The latter approach offers several advantages over the former, as the Jacobian matrix is no longer
 200 required to maintain the temporal accuracy of the solution. The GMRES solver still requires a preconditioning matrix,
 201 which generally needs to be stored. However, this matrix can be approximated or frozen for a certain number of
 202 iterations without losing the formal order of accuracy of the time integration scheme, thus providing an improvement
 203 in computational efficiency. For example, when using a block-Jacobi preconditioning approach, the memory footprint
 204 required for the Jacobian matrix can be one order of magnitude lower, as only the memory for the on-diagonal blocks
 205 needs to be allocated. The additional residual evaluation, which are required in (28), have a computational cost similar
 206 to a matrix-vector product for high-order polynomials. Further details can be found in previous work [15, 13].

207 *5.2. Hybridizable discontinuous Galerkin discretization*

208 *5.2.1. Mixed form*

209 The system in (26) can be conveniently arranged using the definition of element-interior and face DoFs. To this
 210 end, considering first the mixed form of the HDG discretization, it is convenient to define the following elemental
 211 block matrices at stage i of the Newton-Krylov method

$$\begin{aligned} \mathbf{A}^{QQ} &= \frac{\partial\mathbf{R}^Q}{\partial\mathbf{Q}}, & \mathbf{A}^{QU} &= \frac{\partial\mathbf{R}^Q}{\partial\mathbf{U}}, & \mathbf{B}^{Q\Lambda} &= \frac{\partial\mathbf{R}^Q}{\partial\Lambda}, \\ \mathbf{A}^{UQ} &= \frac{\partial\mathbf{R}^U}{\partial\mathbf{Q}}, & \mathbf{A}^{UU} &= \frac{\mathbf{M}^U}{a_{ii}\Delta t} + \frac{\partial\mathbf{R}^U}{\partial\mathbf{U}}, & \mathbf{B}^{U\Lambda} &= \frac{\partial\mathbf{R}^U}{\partial\Lambda}, \\ \mathbf{C}^{\Lambda Q} &= \frac{\partial\mathbf{R}^\Lambda}{\partial\mathbf{Q}}, & \mathbf{C}^{\Lambda U} &= \frac{\partial\mathbf{R}^\Lambda}{\partial\mathbf{U}}, & \mathbf{D} &= \frac{\partial\mathbf{R}^\Lambda}{\partial\Lambda}, \end{aligned} \quad (30)$$

212 while the right hand side of Eq. (26) is obtained as

$$\begin{aligned}
\mathbf{f}^Q &= -\sum_{j=1}^{i-1} \frac{a_{ij}}{a_{ii}} \mathbf{R}^Q(\mathbf{Q}^j, \mathbf{U}^j, \Lambda^j) - \mathbf{R}^Q(\mathbf{Q}_k^i, \mathbf{U}_k^i, \Lambda_k^i), \\
\mathbf{f}^U &= -\frac{\mathbf{M}^U}{a_{ii} \Delta t} (\mathbf{U}_k^i - \mathbf{U}^n) - \sum_{j=1}^{i-1} \frac{a_{ij}}{a_{ii}} \mathbf{R}^U(\mathbf{Q}^j, \mathbf{U}^j, \Lambda^j) - \mathbf{R}^U(\mathbf{Q}_k^i, \mathbf{U}_k^i, \Lambda_k^i), \\
\mathbf{g} &= -\sum_{j=1}^{i-1} \frac{a_{ij}}{a_{ii}} \mathbf{R}^\Lambda(\mathbf{Q}^j, \mathbf{U}^j, \Lambda^j) - \mathbf{R}^\Lambda(\mathbf{Q}_k^i, \mathbf{U}_k^i, \Lambda_k^i).
\end{aligned} \tag{31}$$

213 The full system of equations can be therefore written in the compact form as

$$\begin{bmatrix} \mathbf{A}^{QQ} & \mathbf{A}^{QU} & \mathbf{B}^{Q\Lambda} \\ \mathbf{A}^{UQ} & \mathbf{A}^{UU} & \mathbf{B}^{U\Lambda} \\ \mathbf{C}^{\Lambda Q} & \mathbf{C}^{\Lambda U} & \mathbf{D} \end{bmatrix} \begin{pmatrix} \Delta \mathbf{Q} \\ \Delta \mathbf{U} \\ \Delta \Lambda \end{pmatrix} + \begin{pmatrix} \mathbf{f}^Q \\ \mathbf{f}^U \\ \mathbf{g} \end{pmatrix} = \mathbf{0}. \tag{32}$$

214 5.2.2. Primal form

215 In the *primal* formulation, the elemental block matrices related to the gradient variables are no longer present in
216 the linear system. Moreover, the right-hand side can be evaluated via the following equations

$$\begin{aligned}
\mathbf{f}^U &= -\frac{\mathbf{M}^U}{a_{ii} \Delta t} (\mathbf{U}_k^i - \mathbf{U}^n) - \sum_{j=1}^{i-1} \frac{a_{ij}}{a_{ii}} \mathbf{R}^U(\mathbf{U}^j, \Lambda^j) - \mathbf{R}^U(\mathbf{U}_k^i, \Lambda_k^i), \\
\mathbf{g} &= -\sum_{j=1}^{i-1} \frac{a_{ij}}{a_{ii}} \mathbf{R}^\Lambda(\mathbf{U}^j, \Lambda^j) - \mathbf{R}^\Lambda(\mathbf{U}_k^i, \Lambda_k^i).
\end{aligned} \tag{33}$$

217 that do not depend anymore on the internal DoFs \mathbf{Q} . The linear system for the primal HDG discretization assumes the
218 form

$$\begin{bmatrix} \mathbf{A}^{UU} & \mathbf{B}^{U\Lambda} \\ \mathbf{C}^{\Lambda U} & \mathbf{D} \end{bmatrix} \begin{pmatrix} \Delta \mathbf{U} \\ \Delta \Lambda \end{pmatrix} + \begin{pmatrix} \mathbf{f}^U \\ \mathbf{g} \end{pmatrix} = \mathbf{0}, \tag{34}$$

219 5.3. Static condensation and back-solve

220 Considering the block structure of the matrices appearing in (32) and (34), the solution of the system can involve
221 a smaller number of DoFs by statically condensing out the element-interior variables. Partitioning the matrix into
222 element-interior and face components, $[\mathbf{A}, \mathbf{B}; \mathbf{C}, \mathbf{D}]$, and similarly for the right-hand side vector, $[\mathbf{f}; \mathbf{g}]$, the Schur-
223 complement linear system reads

$$\underbrace{(\mathbf{D} - \mathbf{C}\mathbf{A}^{-1}\mathbf{B})}_{\mathcal{K}} \Delta \Lambda = \underbrace{(\mathbf{g} - \mathbf{C}\mathbf{A}^{-1}[\mathbf{f}])}_{\mathbf{b}}, \tag{35}$$

224 which assumes the same form as system (27) and can be solved using a GMRES algorithm. The definition of each
225 block can be found from (32) and (34). The static condensation is an operation that involves matrix-matrix products
226 for the iteration matrix, as well as matrix-vector products for the right-hand side. Fortunately, the compact structure
227 of the residual Jacobian prevents us from having to allocate global matrices for the computation of the condensed
228 matrix, *i.e.* the operations described in Eq. (35) are local to each element. In addition, the computation of \mathbf{A}^{-1} can

229 be performed in place. By doing so, we do not increase the memory footprint of the HDG implementation during the
 230 solve.

231 After the solution of (35), the interior states have to be recovered for the residual evaluation in the next time step.
 232 This operation is commonly referred to as the *back solve* and assumes the following form

$$\Delta \mathbf{U} = -\mathbf{A}^{-1} (\mathbf{f} + \mathbf{B} \Delta \Lambda). \quad (36)$$

233 Our implementation choice of assembling the condensed matrix on-the-fly requires re-evaluation of the inverse of the
 234 matrix \mathbf{A} in an element-wise fashion during the back-solve.

235 As a final remark for the two solvers, we point out that for both the *mixed* and *primal* form of HDG, memory
 236 allocation and time spent on the global solve are lower than that of a DG solver due to the smaller number of globally-
 237 coupled degrees of freedom at high orders. On the other hand, the inversion of the \mathbf{A} block-structured matrix of
 238 equation (35), although local to each element, increases the amount of element-wise operations.

239 6. Multigrid preconditioning

240 The use of a p -multigrid strategy to precondition a flexible implementation [8] of GMRES is explored in the
 241 context of the spatial discretizations presented. The basic multigrid idea is to exploit iterative solvers to smooth-out
 242 high-frequency components of the error with respect to an unknown exact solution. Such iterative solvers are not
 243 sufficiently effective at damping low-frequency error components, and to this end, an iterative solution built using
 244 coarser problems can be useful to shift, via system projection, the low-frequency modes towards the upper side of the
 245 spectrum. This simple and effective strategy allows us to obtain satisfactory rates of convergence over the entire range
 246 of error frequencies.

247 In p -multigrid the coarse problems are built based on lower-order discretizations with respect to the original
 248 problem of degree k . We consider L coarse levels denoted by the index $\ell = 0, \dots, L$ and indicate the fine and coarse
 249 levels with $\ell = 0$ and $\ell = L$, respectively. The polynomial degree of level ℓ is k_ℓ and the polynomial degrees of
 250 the coarse levels are chosen such that $k_\ell < k_{\ell-1}$, with $k_0 = k$. These orders are used to build coarser linear systems
 251 $\mathcal{K}_\ell \mathbf{x}_\ell = \mathbf{b}_\ell$. In order to avoid additional integration of the residuals and Jacobians on the coarse level, we employ
 252 subspace inheritance of the matrix operators assembled on the finest space to build the coarse space operators \mathcal{K}_i for
 253 both DG and HDG discretizations. This choice involves projections of the matrix operators and right hand sides, which
 254 are computed only once on the finest level. Compared to subspace non-inheritance, which requires the re-evaluation of
 255 the Jacobians in proper coarser-space discretizations of the problem, inheritance is cheaper in processing and memory.
 256 Although previous work has shown lower convergence rates when using such cheaper operators [32, 11, 12], especially
 257 in the context of elliptic problems and incompressible flows, we found these operators sufficiently efficient for our
 258 target problems involving the compressible NS equations, as will be demonstrated in the results section.

259 In the context of standard discontinuous Galerkin discretizations, see for example [9, 6, 10, 11, 12], the p -multigrid
 260 approach has been thoroughly investigated and exploited in several ways, *e.g.* h -, p -, and hp -strategies. On the other

Algorithm 1 MG_{full}

```
1: for  $\ell = L, 0, -1$  do
2:   if  $\ell = L$  then
3:      $\mathbf{b}_\ell = \mathbf{I}_0^\ell \mathbf{b}_0$ 
4:     SOLVE  $\mathbf{A}_\ell \mathbf{x}_\ell^{FMG} = \mathbf{b}_\ell$ 
5:   else
6:      $\mathbf{b}_\ell = \mathbf{I}_0^\ell \mathbf{b}_0$ 
7:      $\tilde{\mathbf{x}}_\ell = \mathbf{I}_{\ell+1}^\ell \mathbf{x}_{\ell+1}^{FMG}$ 
8:      $\mathbf{x}_\ell^{FMG} = MG_V(\ell, \mathbf{b}_\ell, \tilde{\mathbf{x}}_\ell)$ 
9:   end if
10: end for
11: return  $\mathbf{x}_\ell^{FMG}$ 
```

Algorithm 2 $MG_{\mathcal{V}}(\ell, \mathbf{b}_\ell, \mathbf{x}_\ell)$

```
1: if  $\ell = L$  then
2:   SOLVE  $\mathbf{A}_\ell \bar{\mathbf{x}}_\ell = \mathbf{b}_\ell$ 
3: else
4:    $\bar{\mathbf{x}}_\ell = \text{SMOOTH}(\mathbf{x}_\ell, \mathbf{A}_\ell, \mathbf{b}_\ell)$ 
5:    $\mathbf{r}_\ell = \mathbf{b}_\ell - \mathbf{A}_\ell \bar{\mathbf{x}}_\ell$ 
6:    $\mathbf{r}_{\ell+1} = \mathbf{I}_{\ell+1}^{\ell+1} \mathbf{r}_\ell$ 
7:    $\mathbf{e}_{\ell+1} = MG_V(\ell + 1, \mathbf{r}_{\ell+1}, \mathbf{0})$ 
8:    $\hat{\mathbf{x}}_\ell = \bar{\mathbf{x}}_\ell + \mathbf{I}_{\ell+1}^\ell \mathbf{e}_{\ell+1}$ 
9:    $\bar{\mathbf{x}}_\ell = \text{SMOOTH}(\hat{\mathbf{x}}_\ell, \mathbf{A}_\ell, \mathbf{b}_\ell)$ 
10: end if
11: return  $\bar{\mathbf{x}}_\ell$ 
```

261 hand, the use of p -multigrid for HDG has not been as widely studied. See, for example, preliminary works [16, 24, 25]
262 related to this research area. The definition of the restriction and prolongation operators, as well as the coarse grid
263 operators and right hand sides, is not straightforward when considering the statically condensed system. We will
264 therefore first introduce the concept of subspace inheritance for a standard DG solver and then extend it to HDG.

265 We employ a full multigrid (FMG) \mathcal{V} -cycle solver, outlined in Algorithm 1. The FMG cycle constructs a good
266 initial guess for a \mathcal{V} -cycle iteration, which starts on the fine space. To do so, the solution is initially obtained on the
267 coarsest level (L), and then prolonged to the next refined one ($L-1$). At this point, a standard \mathcal{V} -cycle is called, such
268 that an improved approximation of the solution can be used for the \mathcal{V} -cycle at level $L-2$. This procedure is repeated
269 until the \mathcal{V} -cycle on the finest level is completed. The single \mathcal{V} -cycle is outlined in Algorithm 2. Starting from a
270 level ℓ , the solution is initially smoothed using an iterative solver (SMOOTH). The residual of the solution, \mathbf{r}_ℓ , is then
271 computed and projected to the coarser level $\ell + 1$, where another \mathcal{V} -cycle is recursively called to obtain a coarse-grid
272 correction $\mathbf{e}_{\ell+1}$. This quantity is prolonged to level ℓ and used to correct the solution to be smoothed again. When
273 the coarsest level is reached, the problem is solved with a larger number of iterations to decrease as much as possible
274 the solution error at a low computational cost.

275 In this work the smoothers consist of preconditioned GMRES solvers. Any combination of single grid precondi-
276 tioners can be coupled with an iterative solver to properly operate as a smoother in a multigrid strategy. Devising a
277 methodology to optimally set the multigrid preconditioner is beyond the scope of the present work. However, previ-
278 ous work [12] has shown that an optimal and scalable solver can be obtained using an aggressive preconditioner on
279 the coarsest space discretization, where the factorization of the matrix can be performed at a low computational cost,
280 and the system has to be solved with a higher accuracy. On the other hand, cheaper operators can be used on the
281 finest levels of the discretization, where the systems need not be solved to a high degree of accuracy. In the following

subsections, the details are given about how the matrices and vectors are restricted and prolonged between multigrid levels.

6.1. DG subspace inheritance

Let us define a sequence of approximation spaces $\mathcal{V}_\ell \subseteq \mathcal{V}_h$ on the same triangulation \mathcal{T}_h , ℓ being a multigrid level, with $\mathcal{V}_h = \mathcal{V}_0 \supset \mathcal{V}_1 \supset \dots \supset \mathcal{V}_L$ and L the number of coarse levels. Note that \mathcal{V}_L denotes the coarsest space. In our p -multigrid setting, each approximation space is defined similarly to (4), but using k_ℓ polynomials, with $k_0 > \dots > k_\ell > \dots > k_L$.

In this context, the prolongation operator can be defined as $\mathcal{I}_{\ell+1}^\ell : \mathcal{V}_{\ell+1} \rightarrow \mathcal{V}_\ell$ such that

$$\sum_{K \in \mathcal{T}_h} \int_K (\mathcal{I}_{\ell+1}^\ell u_{\ell+1} - u_{\ell+1}) dK = 0, \quad \forall u_{\ell+1} \in \mathcal{V}_{\ell+1}. \quad (37)$$

Similarly, the restriction operator can be defined as the L_2 projection $\mathcal{I}_\ell^{\ell+1} : \mathcal{V}_\ell \rightarrow \mathcal{V}_{\ell+1}$ such that

$$\sum_{K \in \mathcal{T}_h} \int_K (\mathcal{I}_\ell^{\ell+1} u_\ell - u_\ell) v_{\ell+1} dK = 0, \quad \forall (u_\ell, v_{\ell+1}) \in \mathcal{V}_\ell \times \mathcal{V}_{\ell+1}. \quad (38)$$

Such a definition can be extended to operate on vector functions $\mathbf{u}_h \in [\mathcal{V}_\ell]^m$ component-wise, i.e. $\mathcal{I}_\ell^{\ell+1} \mathbf{u}_h = \{\mathcal{I}_\ell^{\ell+1} u_i\}$.

Regarding coarse-space matrices, discrete matrix operators $\mathbf{I}_\ell^{\ell+1} \in \mathbb{R}^{p \times q}$, with $p = n_e n_v^\ell m$, $q = n_e n_v^{\ell+1} m$ and n_v^ℓ the number of DoFs on level ℓ , have to be considered to inherit the fine-space iteration matrix \mathcal{K}_0 . This matrix can be restricted up to level ℓ via $\mathcal{K}_\ell = (\mathbf{I}_0^\ell) \mathcal{K}_0 (\mathbf{I}_0^\ell)^T$. Fortunately, the explicit assembly of the operators can be avoided and the projection can be applied for each matrix block b of size $(n_v^\ell m)^2$, which assumes the following form

$$\mathcal{K}_{\ell+1}^b = \mathbf{M}_{\ell+1, \ell} \mathcal{K}_\ell^b (\mathbf{M}_{\ell+1, \ell})^T, \quad (39)$$

where

$$(\mathbf{M}_{\ell+1, \ell}) = (\mathbf{M}_{\ell+1})^{-1} \int_K \boldsymbol{\phi}^{\ell+1} \otimes \boldsymbol{\phi}^\ell dK, \quad \mathbf{M}_{\ell+1} = \int_K \boldsymbol{\phi}^{\ell+1} \otimes \boldsymbol{\phi}^{\ell+1} dK. \quad (40)$$

Here, $\boldsymbol{\phi}^\ell$ denotes the set of basis functions defined in element K of order k_ℓ . We point out that, thanks to the use of basis functions defined in a reference element, the projection operators are identical for each element and pair of polynomial orders, and they can be used in the same way for the on-diagonal and off-diagonal blocks of the iteration matrix. As the prolongation operators can be obtained from the restriction by the transpose, $\mathbf{I}_{\ell+1}^\ell = (\mathbf{I}_\ell^{\ell+1})^T$, the method requires the allocation of only L matrices of size $n_v^{\ell+1} \times n_v^\ell$ for each different type of element, which is inexpensive from a memory footprint viewpoint.

6.2. HDG subspace approximate-inheritance

In HDG, the globally-coupled unknowns are those related to the face DoFs, and the iteration matrix is obtained through static condensation, see Equation (35), which allows us to solve the system for the face unknowns only. Theoretically speaking, for element-interior degrees of freedom, the same operators of Section 6.1 can be employed,

307 while those for faces degrees of freedom can be obtained through similar considerations. In this case, the sequence of
 308 approximation spaces $\mathcal{M}_\ell \subseteq \mathcal{M}_h$ is properly defined on the interior mesh element faces, \mathcal{F}_h , with ℓ a multigrid level.
 309 To this end, we define $\mathcal{M}_h = \mathcal{M}_0 \supset \mathcal{M}_1 \supset \dots \supset \mathcal{M}_L$. The prolongation operator is now $\mathcal{J}_{\ell+1}^\ell : \mathcal{M}_{\ell+1} \rightarrow \mathcal{M}_\ell$, defined
 310 by

$$\sum_{F \in \mathcal{F}_h} \int_F (\mathcal{J}_{\ell+1}^\ell \lambda_{\ell+1} - \lambda_{\ell+1}) d\sigma = 0, \quad \forall \lambda_{\ell+1} \in \mathcal{M}_{\ell+1}. \quad (41)$$

311 The restriction operator is defined as $\mathcal{J}_\ell^{\ell+1} : \mathcal{M}_\ell \rightarrow \mathcal{M}_{\ell+1}$ such that

$$\sum_{F \in \mathcal{F}_h} \int_F (\mathcal{J}_\ell^{\ell+1} \lambda_\ell - \lambda_\ell) \mu_{\ell+1} d\sigma = 0, \quad \forall (\lambda_\ell, \mu_{\ell+1}) \in \mathcal{M}_\ell \times \mathcal{M}_{\ell+1}. \quad (42)$$

312 These definitions can also be extended to operate on vector functions $\lambda_h \in [\mathcal{M}_\ell]^M$ and are assumed to act component-
 313 wise.

314 Applying the same subspace-inheritance idea used for DG, one obtains the coarse space condensed HDG matrix
 315 and right hand side through the application of element-interior and face DoF projections,

$$\mathcal{K}_\ell = \left((\mathbf{J}_0^\ell) \mathbf{D}_0(\mathbf{J}_\ell^0) - [(\mathbf{J}_0^\ell) \mathbf{C}_0(\mathbf{I}_\ell^0)] [(\mathbf{I}_0^\ell) \mathbf{A}_0(\mathbf{I}_\ell^0)]^{-1} [(\mathbf{I}_0^\ell) \mathbf{B}_0(\mathbf{J}_\ell^0)] \right), \quad (43)$$

$$\mathbf{b}_\ell = \left((\mathbf{J}_0^\ell) \mathbf{g}_0 - [(\mathbf{J}_0^\ell) \mathbf{C}_0(\mathbf{I}_\ell^0)] [(\mathbf{I}_0^\ell) \mathbf{A}_0(\mathbf{I}_\ell^0)]^{-1} [(\mathbf{I}_0^\ell) \mathbf{f}_0] \right). \quad (44)$$

317 We point out that this operation involves the application of mixed element-interior and face degrees of freedom
 318 Galerkin projections prior to the static condensation of the system. Thus, it comes with an increased operation count
 319 compared to DG subspace inheritance. To minimize the number of operations involved in the projection, we introduce
 320 an *approximate*-inherited approach where the coarse space matrices and right-hand sides are obtained by simply
 321 applying face projections to the condensed matrices and vectors on the finest space. In other words, we obtain \mathcal{K}_ℓ and
 322 \mathbf{b}_ℓ through

$$\mathcal{K}_\ell = (\mathbf{J}_0^\ell) \mathcal{K}_0(\mathbf{J}_\ell^0), \quad (45)$$

$$\mathbf{b}_\ell = (\mathbf{J}_0^\ell) \mathbf{b}_0. \quad (46)$$

324 This coarse space matrix and right-hand side will in general differ from those of Equation (43) and (44), as the
 325 element-interior operator in the Schur complement term is projected differently.

326 Similar considerations have to be made for the residual evaluation on the coarse levels, which are required in the
 327 projection from level ℓ to $\ell + 1$. The residual on a level ℓ should be computed as

$$\mathbf{r}_\ell = \left(\mathbf{R}_\ell^\Lambda - [(\mathbf{J}_0^\ell) \mathbf{C}_0(\mathbf{I}_\ell^0)] [(\mathbf{I}_0^\ell) \mathbf{A}_0^{QQ}(\mathbf{I}_\ell^0)]^{-1} \mathbf{R}_\ell^{QU}, \right) \quad (47)$$

328 and this requires the evaluation of the element-interior degrees of freedom from the face unknowns. This operation
 329 would require a back-solve on the coarse levels, which increases the amount of operations within each multigrid cycle.

330 To reduce the operation count as much as possible we again rely on the computation of an approximate projection that
 331 is evaluated using the working variable $\Delta\Lambda$ only:

$$\mathbf{r}_\ell = \mathcal{K}_\ell \Delta\Lambda_\ell + \mathbf{b}_\ell. \quad (48)$$

332 Despite the use of those approximations compared to DG subspace inheritance, such a strategy exhibits very good
 333 performance in HDG solutions of the compressible Navier–Stokes equations, as will be demonstrated in the results
 334 section.

335 Similarly to DG, the global assembly of the projection matrix \mathbf{J}_0^ℓ for face degrees of freedom is not strictly required
 336 for HDG, since the projection is applied to each block b of the matrix of size $(n_v^\ell m)^\ell$, with n_v^ℓ related to face degrees
 337 of freedom on level ℓ . The projection of each block b of the iteration matrix assumes the form

$$\mathcal{K}_{\ell+1}^b = \mathbf{N}_{\ell+1,\ell} \mathcal{K}_\ell^b (\mathbf{N}_{\ell+1,\ell})^T, \quad (49)$$

338 where

$$(\mathbf{N}_{\ell+1,\ell}) = (\mathbf{N}_{\ell+1})^{-1} \int_F \boldsymbol{\mu}^{\ell+1} \otimes \boldsymbol{\mu}^\ell \, d\sigma, \quad \mathbf{N}_{\ell+1} = \int_F \boldsymbol{\mu}^{\ell+1} \otimes \boldsymbol{\mu}^{\ell+1} \, d\sigma. \quad (50)$$

339 In this case $\boldsymbol{\mu}^\ell$ denotes the set of basis functions defined in the element face F of order k_ℓ . Thanks to the use of
 340 basis functions defined in the reference element of the space discretization, similar considerations to those related to
 341 element-interior degrees of freedom projection hold true.

342 To assess the impact of the approximate inheritance approximation on the HDG coarse-space condensed matrix,
 343 \mathcal{K} , we conduct an eigenvalue analysis of the inherited and approximate-inherited operators for a simplified test case.
 344 The problem of interest is scalar advection-diffusion in a unit square domain ($L = 1$), with advective velocity $\vec{v} =$
 345 $(0.8, 0.6)$, diffusivity $\nu = |\vec{v}|L/Pe$, unit Dirichlet boundary condition on the bottom boundary, and homogenous zero
 346 Dirichlet boundary conditions on all other boundaries. Note, Pe is the Péclet number, defined by $Pe = |\vec{v}|L/\nu$.

347 The computational grid consists of uniform $N \times N$ squares, each subdivided into two triangles, for a total of $2N^2$
 348 elements. We use $N = 8$, a fine-space order of $p_0 = 2$, and a coarse-space order of $p_1 = 1$. For various Pe , we compute
 349 both the inherited condensed matrix, $\mathcal{K}_1^{\text{inherited}}$ via (43), and the approximate-inherited condensed matrix, $\mathcal{K}_1^{\text{approx}}$ via
 350 (45). Note that for this linear problem, the inherited matrix is equivalent to the condensed matrix computed directly
 351 on the coarse space. Figure 2 compares the eigenvalue spectra of the condensed matrices for three different Péclet
 352 numbers. The eigenvalues are not identical, but they are located in generally the same areas and have similar real and
 353 imaginary components. Iterative solution properties of the matrices are therefore expected to be similar.

354 6.3. Preconditioning options and memory footprint

355 In this work we exploit single-grid preconditioners both for benchmarking and to precondition the smoothers of
 356 the multigrid strategy. Two operators will be considered in this work. The first and simplest one will be here labelled
 357 as element-wise block-Jacobi (BJ). The BJ preconditioner extracts the block-diagonal portion of the iteration matrix
 358 and factorizes it, in a local-to-each element fashion, using the PLU factorization. This cheap and memory saving

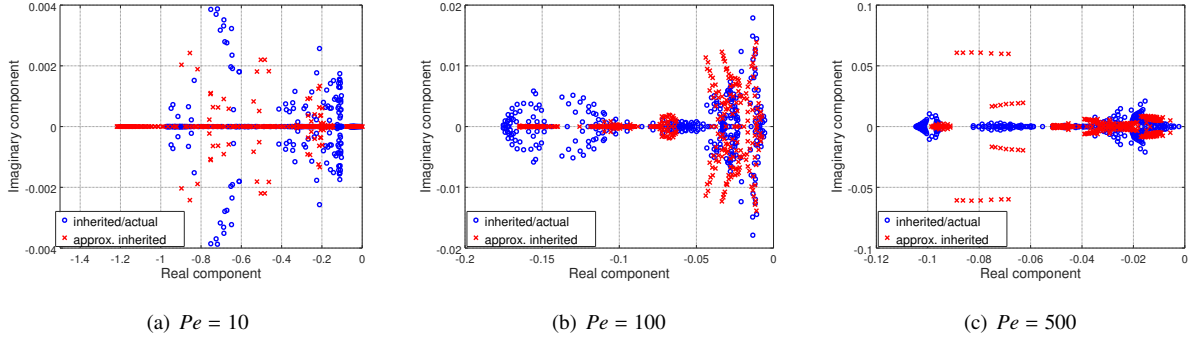


Figure 2: Eigenvalues of the inherited and approximate-inherited condensed HDG matrices, \mathcal{K} , on the coarse $p = 1$ space for a scalar advection-diffusion test case.

359 preconditioner becomes more effective as the time step of the discretization decreases, *i.e.* the matrix becomes more
 360 diagonally dominant and the condition number decreases. In addition, the BJ preconditioner is applied in the same
 361 way in serial and parallel computations.

362 The second preconditioner is the incomplete lower-upper factorization with zero-fill, ILU(0). In particular, the
 363 minimum discarded fill reordering proposed in [5] is employed. The algorithm was shown to be suited for stiff spatial
 364 discretizations, and it allows for an in-place factorization [6]. When it is applied in parallel, the ILU(0) is performed
 365 separately on each square, partition-wise block of the iteration matrix. In this case this preconditioner will be labelled
 366 as block-ILU(0) (BILU). As a natural downside, BILU loses preconditioning efficiency when it is applied in parallel.
 367 A variant that compensates for this effect is the Additive Schwarz method, which extends the partition-wise block of
 368 the Jacobian with a number of overlapping elements between the mesh partitions. This algorithm, employed in the
 369 context of incompressible Navier–Stokes equations, increases the memory footprint of the solver when a few elements
 370 per partition are used [12]. We do not use such an approach in the present study of compressible flow.

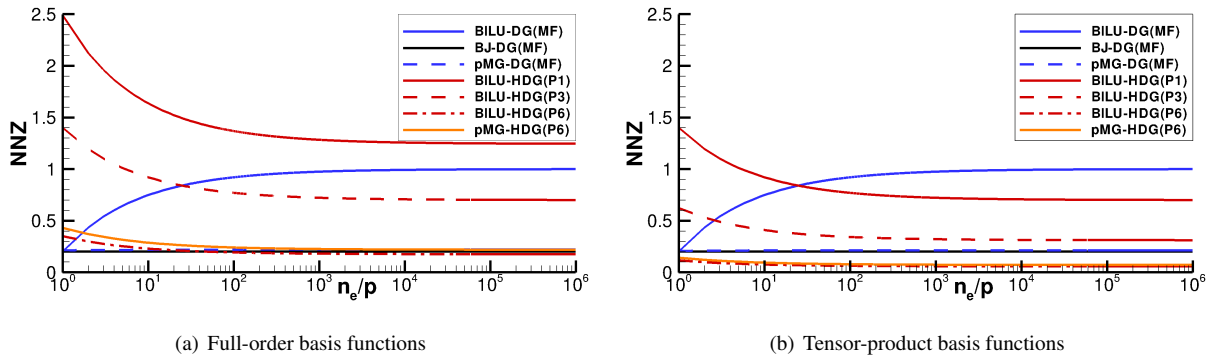


Figure 3: Allocated number of non-zeros (NNZ) non-dimensionalized by the memory allocation of the DG Jacobian matrix. DG matrix-free solvers compared to HDG for different values of polynomial orders and preconditioning type. p -multigrid preconditioning (p MG) is assumed to be that of Table 3.

371 We now provide estimates of the memory footprint of the solver as well as the computational time spent evalu-
 372 ating the matrix operators. It is worth pointing out that, for DG, the matrix assembly time, as well as its operation
 373 count, is a function of the number of non-zeros of the matrix itself, while HDG involves the overhead costs of the
 374 static condensation and back solve. Considering a square, two-dimensional, bi-periodic domain made of quadrilateral
 375 elements, we obtain the results shown in Figure 3, where the number of non-zeros (NNZ), non-dimensionalized by
 376 the number of nonzeros of the Jacobian arising from the DG discretization, is reported as a function of the number of
 377 elements per partition, n_e/p . It can be observed that:

- 378 1. The memory footprint of DG, matrix-based as well as HDG solvers is always equal to that of a Jacobian matrix,
 379 and this value is a function of the polynomial order for HDG. This is due to the fact that the preconditioner is
 380 always evaluated using the same memory as the Jacobian matrix.
- 381 2. For a matrix-free, DG discretization, the allocation involves only the preconditioner operator. When BJ is
 382 considered, NNZ reduces by 80% with respect to the allocation of a full DG Jacobian. As $n_e/p \rightarrow 1$, the NNZ
 383 of the BILU solver approaches that of the element-wise block Jacobi, while for $n_e/p \gg 1$ it tends to be that of a
 384 Jacobian matrix. This is due to the fact that as the domain is partitioned, the ILU(0) factorization is performed in
 385 the squared, partition-wise block of the iteration matrix and therefore, in a matrix-free fashion, the off-partition
 386 blocks can be neglected during the assembly phase.
- 387 3. The p -multigrid (p MG) matrix-free preconditioning approach applied to a DG discretization, here assumed to
 388 be that of Table 3, requires a memory footprint in line with that of an element-wise block Jacobi method, as
 389 already observed in [12]. In fact, when using lower-order polynomial spaces with $k_\ell \ll k$, the size of those
 390 matrices is considerably smaller than that of the finest space since they scale with k^{2d} .
- 391 4. For HDG, only for high-order polynomials is NNZ reduced with respect to the iteration matrix of a DG method.
 392 For $k = 6$, a memory footprint in line with that of a BJ, matrix-free approach is observed, while for $k = 1, 3$ the
 393 memory is considerably larger. It is worth pointing out that the use of full-order basis functions (Figure 3(a))
 394 and tensor-product basis functions (Figure 3(b)) only affects the NNZ ratio in the HDG case: in particular, the
 395 memory footprint reduction when using a tensor-product basis is larger due to the higher amount of element-
 396 wise unknowns compared to internal face unknowns.
- 397 5. The NNZ ratio for HDG increases when $n_e/p \rightarrow 1$. The reason for this is that the face-to-element ratio within
 398 the computational mesh of the domain partition increases.

399 Finally, it is important to remark that for a matrix-free iterative solver employed in DG contexts, it is possible to
 400 optimize the matrix assembly evaluation to compute only the blocks required by the preconditioner. For example, for
 401 BJ, the evaluation of the off-diagonal blocks of the Jacobian can be neglected, while for p MG matrix-free with BJ on
 402 the finest space, the off-diagonal blocks could be computed at a reduced polynomial order consistent with that of the
 403 coarser spaces.

404 7. Numerical results on a model test case

405 We present numerical experiments to assess the performance of the HDG discretizations in comparison to DG.
 406 First, Navier–Stokes solutions of a vortex transported by uniform flow at $M = 0.05$ and $Re = 100$ are reported. The
 407 objective is i) to show the convergence rates of the solver both in space and time; ii) to investigate the effects of grid re-
 408 finement for the approximate-inherited approach proposed for HDG, providing mesh-independent convergence rates;
 409 and iii) compare the effects of polynomial order, time step size and space discretization on the parallel performance
 410 of the solution strategy.

411 7.1. Test case description

412 The test case is a modified version of the VII case studied in the 5th International Workshop on High Order
 413 CFD Methods [33], and it consists of a two-dimensional mesh of the domain $(x, y) \in [0, 0.1] \times [0, 0.1]$ with periodic
 414 boundary conditions on each side. The flow initialization involves the definition of the following state

$$\begin{aligned}
 u &= U_\infty \left(1 - \beta \left(\frac{y - Y_c}{R} \right) e^{-r^2/2} \right) \\
 v &= U_\infty \beta \left(\frac{x - X_c}{R} \right) e^{-r^2/2} \\
 T &= T_\infty - \left(\frac{U_\infty^2 \beta^2}{2C_p} \right) e^{-r^2}
 \end{aligned} \tag{51}$$

415 with the heat capacity at constant pressure $C_p = R_{\text{gas}} \gamma / (\gamma - 1)$, the non dimensional distance to the initial vortex
 416 core position $r = \sqrt{(x - X_c)^2 + (y - Y_c)^2} / R$, X_c, Y_c the coordinates of the vortex center, and the free stream velocity
 417 $U_\infty = M_\infty \sqrt{\gamma R_{\text{gas}} T_\infty}$. The fluid pressure p , temperature T , and density ρ are prescribed to ensure a steady solution
 418 of the problem in the freestream co-moving frame, *i.e.* $\rho_\infty = p_\infty / R_{\text{gas}} T_\infty$, $\rho = \rho_\infty (T / T_\infty)^{1/(\gamma-1)}$, $p = \rho R_{\text{gas}} T$. The
 419 parameters were chosen such that $M_\infty = 0.05$, $\beta = 1/50$ and $R = 0.005$. In contrast to the inviscid-flow case studied
 420 in the workshop, the governing equations in the present study are Navier-Stokes, with $Re = 100$ based on the domain
 421 size.

422 7.2. Assessment of the solution accuracy

423 Numerical experiments have been performed to assess the output error, both in space and time. The meshes
 424 for the study consist of regular quadrilaterals. The mesh density ranges from 2×2 to 64×64 , while the polynomial
 425 order range is $k \in \{1, 2, 3, 4, 5, 6\}$. The L_2 state error was computed relative to the solution on a 128×128 , \mathbb{P}_6 space
 426 discretization, after one convective period, T . Contour plots of the solution at the initial and final states are shown in
 427 Figure 4. Figure 5(a) reports space discretization errors. The tests were performed using a very small time step size,
 428 $T/\Delta t = 4000$, and the ESDIRK3 scheme to ensure a negligible time discretization error, with an absolute tolerance on
 429 the non-linear system of 10^{-10} , and a relative tolerance of 10^{-5} on GMRES. Even though DG suffers less than HDG
 430 from pre-asymptotic behaviour on such a smooth solution, all three implementations show comparable error levels

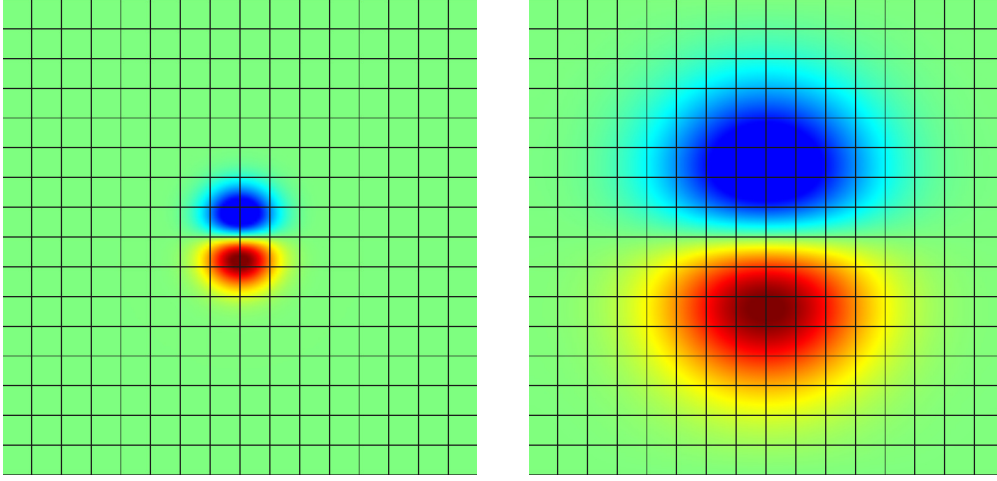


Figure 4: Convected vortex at $Re = 100$, $M = 0.05$. Mach number contours. Solution at $t = 0$ (left) and $t = T$ (right).

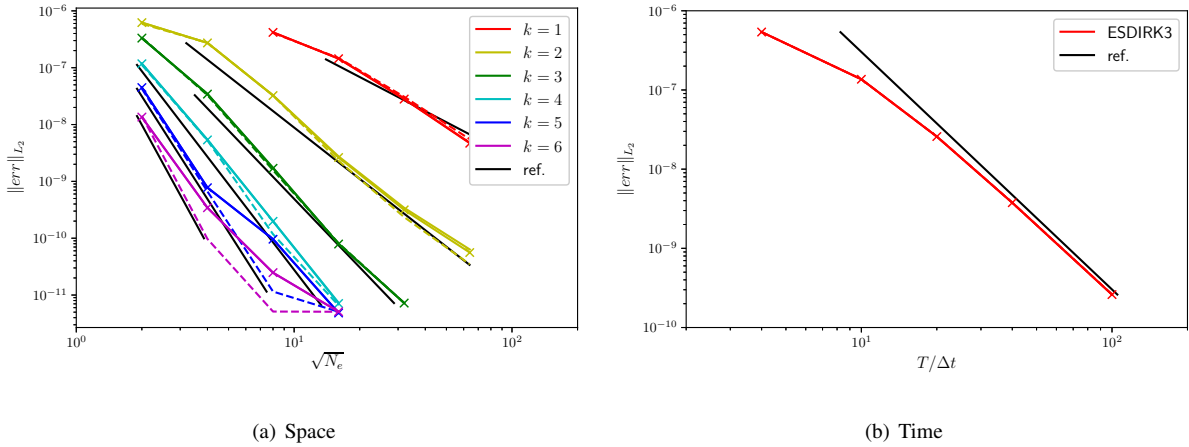


Figure 5: L_2 solution error. Laminar vortex test case at $Re = 100$, $M = 0.05$. Convergence rates for the DG (dashed), $mHDG$ (solid) and $pHDG$ (crosses) discretizations versus theoretical estimates in space and time.

431 and converge with the theoretical convergence rates for every polynomial approximation shown. As a consequence
 432 of such analysis, and considering that both the DG and HDG implementations share the same code base, we will
 433 consider only the CPU time as a measure of the time-to-solution efficiency.

434 Regarding the time integration scheme, the convergence rates of ESDIRK3 are also reported in Figure 5(b), and
 435 these were obtained using the 16×16 grid, with \mathbb{P}_6 polynomials for the three space discretization strategies. The
 436 theoretical third-order convergence rate of the ESDIRK3 time integration scheme can be observed for all three space
 437 discretizations with comparable error levels.

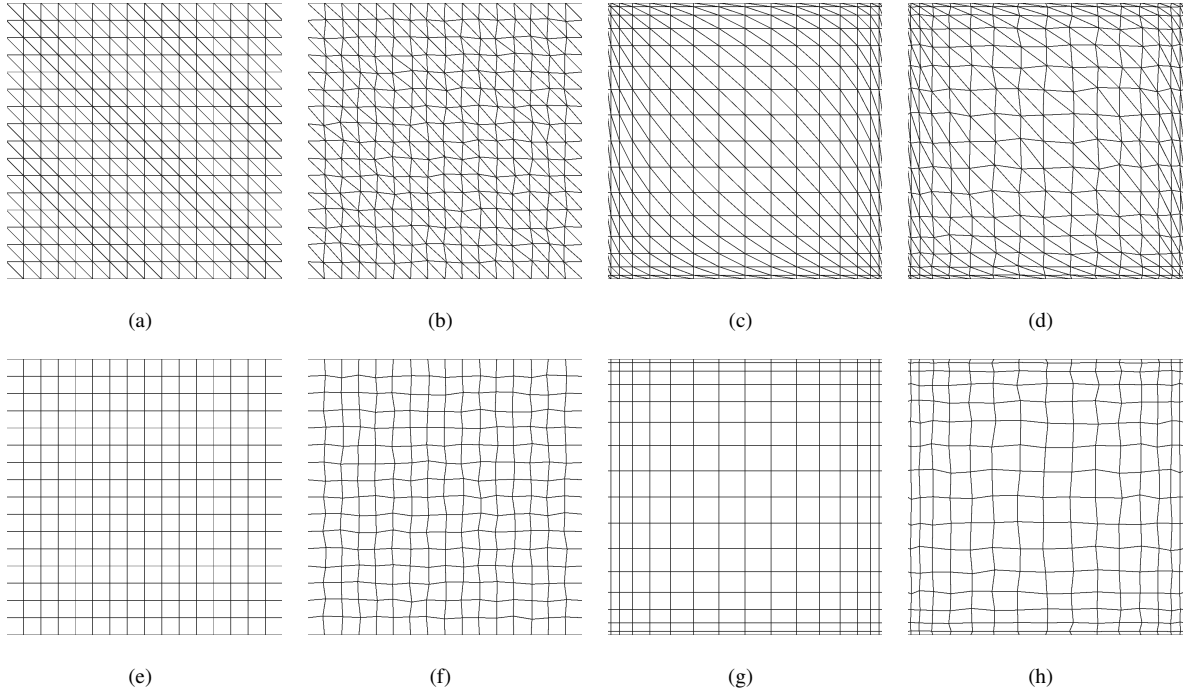


Figure 6: Convected vortex at $Re = 100$, $M = 0.05$. Examples of triangular and quadrilateral meshes involving i) regular elements; ii) randomly distorted elements; iii) regular and clustered elements; and iv) clustered-distorted elements.

438 7.3. Assessment of the approximate-inherited multigrid approach for HDG

439 To demonstrate the efficacy of the multigrid approach proposed herein for HDG, we report numerical experiments
 440 obtained by reducing the mesh element size for different element types. Four mesh sequences are considered in the
 441 study, obtained as i) regular elements; ii) randomly distorted elements; iii) regular and clustered elements; and iv)
 442 clustered-distorted elements. The study was performed on both triangular and quadrilateral elements, see Figure 6.
 443 We remark that the random mesh perturbation was limited to 10% of the minimum dimension of the element, and
 444 that the clustering has been obtained by placing the mesh element nodes using Gauss-Lobatto rules. A full multigrid
 445 strategy has been employed for the study. The strategy combines three multigrid levels, and for each of them a BILU-
 446 preconditioned GMRES smoother is considered. To avoid the impact of domain decomposition, all computations are
 447 performed in serial. Three smoothing iterations were performed on each level, while 400 iterations were employed
 448 on the coarsest level to ensure that the results are not polluted by a lack of coarse-level resolution. This configuration
 449 was found to be optimal for the present serial computations, and it is consistent with previous work in the literature,
 450 see [12].

451 Table 2 report the results on triangular and quadrilateral mesh elements. The numerical experiment consists of
 452 one time period such that $T/\Delta t = 10$ using the ESDIRK3 scheme, with T the convective period of the problem.
 453 The average number of iterations as well as the average convergence rate (CR) ρ are reported. The CR is defined as
 454 $\rho = (r_{IT}/r_0)^{1/IT}$ with IT the average number of GMRES iterations, r_0 and r_{IT} the residuals at the first and IT^{th} iteration

	Reg Tri		Dis Tri		Reg Grad Tri		Dis Grad Tri	
n_e	IT_a	ρ_a	IT_a	ρ_a	IT_a	ρ_a	IT_a	ρ_a
$16^2 \cdot 2$	3.000	0.0060	2.833	0.0056	3.000	0.0120	3.000	0.0131
$32^2 \cdot 2$	3.000	0.0123	3.000	0.0119	3.500	0.0245	3.333	0.0195
$64^2 \cdot 2$	2.500	0.0058	3.000	0.0132	3.667	0.0315	4.000	0.0422
$128^2 \cdot 2$	2.333	0.0047	2.667	0.0086	3.500	0.0290	4.167	0.0560
	Reg Quad		Dis Quad		Reg Grad Quad		Dis Grad Quad	
n_e	IT_a	ρ_a	IT_a	ρ_a	IT_a	ρ_a	IT_a	ρ_a
16^2	2.833	0.0053	2.333	0.0033	3.000	0.0069	3.000	0.0057
32^2	2.833	0.0101	2.833	0.0081	3.000	0.0087	3.000	0.0096
64^2	3.167	0.0183	3.000	0.0153	3.833	0.0386	3.667	0.0336
128^2	3.333	0.0246	3.500	0.0273	4.333	0.0637	4.500	0.0659

Table 2: Laminar vortex test case at $Re = 100$, $M = 0.05$. h -independence test on tri-element (top) and quad-element (bottom) meshes. of a FGMRES(MG_{full}) solver built on three levels. GMRES(BILU) smoothers with 400 iterations on the coarsest level $\ell = 2$. 3 smoothing iterations were employed for $\ell = \{0, 1\}$.

455 respectively. In all of the numerical experiments, the p -multigrid strategy appears to work optimally as the number of
456 iterations only slightly grows with the number of mesh elements, even for distorted and graded mesh sequences.

457 7.4. Evaluation of the solver efficiency

458 We report results of numerical experiments devoted to assess the performance of the p -multigrid preconditioning
459 strategy for HDG in comparison to other operators, as well as state-of-the-art preconditioned DG discretizations. In
460 particular, we take as a reference the matrix-based, BILU preconditioned DG discretization and we aim at reporting
461 insights on the computational time of the solution, in order to provide an overall idea of the computational efficiency
462 of the solver. The space discretization relies on the 16×16 mesh made by regular quadrilaterals and two polynomial
463 orders, *i.e.* $k = \{3; 6\}$. As for the time discretization, non dimensional time steps of $\Delta t = \{1; 0.1\}$ are employed. In both
464 cases, a single time step is computed, which corresponds to three non-linear problems. We observe that our Newton
465 solver converges in two iterations, which means that the CPU time and the average number of iterations are evaluated
466 considering a total of six linear system solutions. Each linear system is solved up to a non-preconditioned relative
467 linear tolerance of 10^{-5} , while an absolute tolerance of 10^{-10} was used for the nonlinear solver.

468 Special attention is hereby given to the parallel efficiency of the computations, both in terms of CPU time and
469 average number of GMRES iterations. To this extent, the ideas reported in [12] on the choice of the number of levels
470 and the smoothing type have been proposed. A very similar behaviour of the solver has been presently observed and
471 thus we explicitly refer to that work for a more in-depth discussion about the effects of the smoothing. We here report
472 only the general idea: a scalable multigrid strategy to precondition linear systems arising from the Navier–Stokes

Level	Order	Solver	Preconditioner	Iterations
1	6	GMRES	BJ	10
2	2	GMRES	BJ	10
3	1	GMRES	BILU	30

Table 3: Computational settings for the p -multigrid smoothers employed within the paper.

473 equations can be obtained by the use of simple BJ-preconditioned GMRES smoothers for all of the levels except
474 the coarse one, where more powerful preconditioners can be cheaply introduced on the smoothers due to the low
475 computational cost of the coarse matrix factorization.

476 In the numerical experiments we rely on a three-level multigrid strategy. In both the cases, $k_{1,2} = \{2, 1\}$ have
477 been used on the coarser spaces smoothed with BJ- and BILU-preconditioned GMRES solvers, respectively. $\{10, 30\}$
478 smoothing iterations have been employed, which were found to be sufficient to provide optimal results both in serial
479 and in parallel computations for HDG and DG as well. On the finest space, 10 smoothing iterations of GMRES(BJ)
480 were used. The computational settings are summarized in Table 3. We remark that these settings have been found to
481 be sufficiently computationally efficient for all of the numerical experiments reported in this paper.

482 Table 4 compares the performance of the multigrid preconditioner to those of BILU for $k = 3$ using $\Delta t = 1$ (left)
483 and $\Delta t = 0.1$ (right), for three space discretizations, *i.e.* DG, m HDG and p HDG. **The table is designed in such a**
484 **way that the merits of the discretization are presented along vertical blocks, while the effects of the preconditioner**
485 **and time step size are presented along horizontal blocks.** For all three space discretization, the performance of the
486 BILU preconditioner degrades in view of the considerable increase in the number of GMRES iterations moving from
487 serial to parallel runs. On the other hand, the multigrid preconditioning strategy shows higher parallel efficiencies
488 in all cases, with the increase in number of iterations either very low (for the largest time step) or non-existent (for
489 the smallest time step). In fact, with the exception of the coarsest space solution, the algorithm is not affected by the
490 domain decomposition.

491 Regarding the CPU time, we report the speed-up values non-dimensionalized by the CPU time of the DG, matrix-
492 based and BILU-preconditioned computation. For the largest time step size, switching from single-grid BILU to
493 multigrid provides consistent benefits on all three space discretizations in view of a higher parallel efficiency of the
494 approach. In fact, despite the speed-up factor being lower for serial computations, it peaks at 32 cores. The strategy
495 provides a speedup of 2.09 for DG, 2.21 and 3.05 for m HDG and p HDG, respectively.

496 For the smaller time step, **the right-hand side of Table 4**, the situation is slightly different. In fact, the reduced
497 conditioning of the matrix reduces the iterative solution times, as well as increases the relative cost of the matrix
498 assembly. For m HDG, where the assembly costs are the highest, we observed speed-up values below one. Conversely,
499 p HDG still outperforms the reference, providing a speed-up factor in the range $[1.29, 1.75]$ due to the smaller number
500 of operations during the Jacobian assembly with respect to the *mixed* form. However, as opposed to what happens for
501 DG, where an improvement in computational efficiency is still observed, the use of p -multigrid does not benefit the

Discr.	$k = 3, \Delta t = 1$						$k = 3, \Delta t = 0.1$					
Solver	BILU-DG			MG _{full} -DG			BILU-DG			MG _{full} -DG		
n_p	Time	E	IT_a	SU_{MB}	E	IT_a	Time	E	IT_a	SU_{MB}	E	IT_a
1	38.52		81.00	1.03		3.50	23.51		27.83	0.88		2.17
2	27.53	0.70	137.33	1.43	0.97	3.50	15.12	0.78	53.17	1.08	0.96	2.17
4	15.10	0.64	154.83	1.51	0.94	3.50	7.68	0.77	55.33	1.06	0.92	2.17
8	8.72	0.55	181.17	1.58	0.85	3.67	4.15	0.71	65.50	1.04	0.84	2.17
16	6.03	0.40	229.50	1.78	0.69	4.00	2.23	0.66	72.33	0.99	0.74	2.17
32	5.82	0.21	284.50	2.09	0.42	4.67	1.57	0.47	81.83	0.96	0.51	2.17
Solver	BILU- m HDG			MG _{full} - m HDG			BILU- m HDG			MG _{full} - m HDG		
n_p	SU_{MB}	E	IT_a	SU_{MB}	E	IT_a	SU_{MB}	E	IT_a	SU_{MB}	E	IT_a
1	1.49		45.17	1.23		2.50	0.96		20.33	0.79		2.00
2	1.75	0.82	77.17	1.54	0.88	2.50	1.03	0.83	36.00	0.88	0.87	2.00
4	1.63	0.70	95.50	1.50	0.78	2.50	0.91	0.72	41.00	0.80	0.78	2.00
8	1.67	0.62	113.17	1.56	0.70	2.50	0.90	0.66	46.83	0.78	0.70	2.00
16	1.76	0.47	138.67	1.75	0.57	2.83	0.79	0.54	52.67	0.68	0.56	2.00
32	1.93	0.27	183.50	2.21	0.37	3.50	0.81	0.39	64.50	0.70	0.42	2.00
Solver	BILU- p HDG			MG _{full} - p HDG			BILU- p HDG			MG _{full} - p HDG		
n_p	SU_{MB}	E	IT_a	SU_{MB}	E	IT_a	SU_{MB}	E	IT_a	SU_{MB}	E	IT_a
1	2.45		44.33	1.95		2.00	1.62		44.33	1.19		2.00
2	2.83	0.81	75.17	2.48	0.89	2.00	1.75	0.84	75.17	1.36	0.89	2.00
4	2.60	0.68	94.17	2.26	0.74	2.50	1.55	0.73	94.17	1.22	0.79	2.00
8	2.57	0.58	112.33	2.34	0.67	2.50	1.51	0.66	112.33	1.19	0.71	2.00
16	2.67	0.44	136.83	2.60	0.53	2.67	1.29	0.52	136.83	1.05	0.58	2.00
32	2.59	0.22	182.83	3.05	0.32	3.50	1.38	0.40	182.83	1.05	0.41	2.00

Table 4: Computational efficiency comparison using DG and HDG discretizations. Laminar vortex test case at $Re = 100$, $M = 0.05$, discretized using 16×16 mesh with \mathbb{P}_3 polynomials. Two time steps, $\Delta t = \{1; 0.1\}$ using the ESDIRK3 scheme are reported. SU_{MB} stands for the speed-up factor relative to the DG, matrix-based, BILU-preconditioned computation, E is the parallel efficiency and IT_a the average number of GMRES iterations.

502 performance of the solver.

503 Table 5 shows the same results for a $k = 6$ space discretization. Similar observations to those reported in Table 4
504 can be made on the overall parallel performance of the preconditioning strategies here considered, *i.e.* the increase in
505 the number of iterations of the preconditioner reflects the performance degradation of the solution strategy when n_p
506 increases. However, in this case, the advantages arising from the use of a multigrid preconditioning strategy are more
507 evident. In fact, for the largest time step size, the speed-up values are higher than those reported in Table 4 involving
508 3rd order polynomials. In particular, when using 32 cores, the speedup reaches 2.65, 1.87 and 3.84 for DG, m HDG and
509 p HDG respectively. In contrast to what was observed previously, when reducing the time step size, the advantages
510 of using a multigrid strategy still appear evident for DG, which provides speed-ups in the range [1.86, 2.29]. While
511 for higher-order polynomials and smaller time steps a *mixed* HDG implementation provides performance in line with

Discr.	$k = 6, \Delta t = 1$						$k = 6, \Delta t = 0.1$					
Solver	BILU-DG			MG _{full} -DG			BILU-DG			MG _{full} -DG		
n_p	Time	E	IT_a	SU_{MB}	E	IT_a	Time	E	IT_a	SU_{MB}	E	IT_a
1	533.69		87.33	1.73		5.67	390.37		32.00	1.90		3.00
2	381.18	0.70	190.83	2.37	0.96	5.83	247.56	0.79	80.83	2.29	0.95	3.00
4	198.53	0.67	213.50	2.42	0.94	5.83	123.33	0.79	85.33	2.25	0.94	3.00
8	109.00	0.61	257.50	2.69	0.95	5.33	62.84	0.78	99.17	2.24	0.92	3.00
16	63.14	0.53	313.17	2.55	0.78	6.33	32.33	0.75	105.83	2.04	0.81	3.00
32	47.91	0.35	392.33	2.65	0.53	7.17	19.61	0.62	110.67	1.86	0.61	3.00

Solver	BILU- m HDG			MG _{full} - m HDG			BILU- m HDG			MG _{full} - m HDG		
n_p	SU_{MB}	E	IT_a	SU_{MB}	E	IT_a	SU_{MB}	E	IT_a	SU_{MB}	E	IT_a
1	1.46		46.50	1.45		2.67	1.08		23.17	1.07		2.17
2	1.75	0.84	112.50	1.76	0.85	2.67	1.16	0.85	50.67	1.16	0.85	2.17
4	1.58	0.73	139.50	1.61	0.75	2.67	1.01	0.74	54.50	1.01	0.74	2.17
8	1.58	0.66	155.83	1.61	0.68	2.83	0.94	0.68	62.17	0.93	0.68	2.17
16	1.52	0.55	203.83	1.59	0.58	3.67	0.83	0.58	71.83	0.83	0.58	2.17
32	1.74	0.42	268.50	1.87	0.45	5.17	0.80	0.46	82.50	0.80	0.46	2.17

Solver	BILU- p HDG			MG _{full} - p HDG			BILU- p HDG			MG _{full} - p HDG		
n_p	SU_{MB}	E	IT_a	SU_{MB}	E	IT_a	SU_{MB}	E	IT_a	SU_{MB}	E	IT_a
1	3.15		45.83	3.11		2.50	2.36		22.17	2.30		2.00
2	3.71	0.83	106.50	3.81	0.86	2.50	2.52	0.84	48.50	2.50	0.86	2.00
4	3.35	0.71	133.00	3.46	0.75	2.67	2.20	0.74	52.50	2.18	0.75	2.00
8	3.35	0.65	149.33	3.46	0.68	2.67	2.04	0.67	59.50	2.03	0.69	2.00
16	3.19	0.54	193.33	3.39	0.58	3.50	1.80	0.58	69.17	1.78	0.58	2.00
32	3.38	0.37	257.00	3.84	0.43	5.17	1.73	0.46	79.67	1.71	0.46	2.00

Table 5: Computational efficiency comparison using DG and HDG discretizations. Laminar vortex test case at $Re = 100$, $M = 0.05$, discretized using 16×16 mesh with \mathbb{P}_6 polynomials. Two time steps, $\Delta t = \{1; 0.1\}$ using ESDIRK3 are reported. SU_{MB} stands for the speed-up factor referred to the DG, matrix-based, BILU-preconditioned computation, E is the parallel efficiency and IT_a the average number of GMRES iterations.

512 that of a matrix-based, BILU-DG solver, the *primal* implementation exhibits better performance overall, even when
513 it shows an overall lower parallel efficiency. In particular, BILU- p HDG is the best performing solution strategy,
514 providing speed-up values in the range $[1.73, 2.36]$. In this case, p -multigrid performs similarly to BILU.

515 Table 6 reports for comparison the speed-up values obtained using a matrix-free implementation of the iterative
516 solver for the DG space discretization. In particular, we compute the matrix-based speedup SU_{MB} using as a reference
517 the matrix-based, BILU-DG solver to show the performance compared to the reference algorithm. The performance
518 is evaluated using the same preconditioners reported in Tables 4 and 5. Considering the single-grid matrix-free,
519 BILU-DG numerical experiments, one can summarize that:

- 520 1. For $k = 3$, switching MB to MF penalizes the CPU time. The reason for this is two-fold: first, the serial
521 computation is 35% slower for MF than for MB. Second, when the solver is applied in parallel, the matrix-free

		$\Delta t = 1$				$\Delta t = 1/10$			
		BILU-DG		MG _{full} -DG		BILU-DG		MG _{full} -DG	
Case	n_p	MF	MFL	MF	MFL	MF	MFL	MF	MFL
$k = 3$	1	0.73	0.97	0.76	0.89	0.84	1.53	0.67	0.85
	2	0.68	0.80	1.02	1.21	0.74	1.09	0.81	1.04
	4	0.65	0.72	1.07	1.28	0.72	1.03	0.78	1.02
	8	0.61	0.73	1.13	1.32	0.68	0.91	0.80	1.03
	16	0.56	0.63	1.20	1.36	0.61	0.79	0.72	0.90
	32	0.59	0.62	1.41	1.59	0.58	0.71	0.69	0.86
$k = 6$	1	1.03	2.10	1.84	2.56	1.00	3.10	1.97	3.36
	2	0.99	1.47	2.42	3.18	0.99	2.03	2.33	3.98
	4	0.96	1.37	2.41	3.32	0.98	1.92	2.23	3.78
	8	0.93	1.25	2.69	3.78	0.95	1.75	2.18	3.65
	16	0.85	1.06	2.35	3.11	0.90	1.56	1.90	3.15
	32	0.78	0.98	2.37	3.03	0.86	1.41	1.70	2.77

Table 6: Computational efficiency of a matrix-free DG strategy. Laminar vortex test case at $Re = 100$, $M = 0.05$, discretized using 16×16 mesh with \mathbb{P}_3 and \mathbb{P}_6 polynomials. Two time steps, $\Delta t = \{1; 0.1\}$ using the ESDIRK3 scheme are reported. $S U_{MB}$ stands for the speed-up factor referred to the matrix-based, BILU-DG computation reported in Tables 4 and 5, $S U_{MF}$ is the speedup computed considering the matrix-free BILU-DG settings. Results obtained by lagging the preconditioner evaluation (MFL) for the entire solution process, *i.e.* six linear systems, are also reported.

522 implementation seems to be less parallel efficient, since the speed-up factor decreases with an increasing the
523 number of processors.

524 2. For $k = 6$, the penalization decreases. In fact, in serial computation, the same computational time has been
525 recovered, meaning that a matrix-free iteration performs similarly to a matrix-vector product. However, a
526 slightly lower parallel efficiency is still observed.

527 3. When a small time step is employed, higher speed-up values are achieved for MF compared to MB. In other
528 words, the lower the number of GMRES iterations, the higher the performance. A slightly higher parallel
529 efficiency is also observed.

530 The possibility of lagging the preconditioner evaluation is also explored and the results are reported in Table 6
531 (MFL). In particular, we skip the recomputation of the preconditioner for six consecutive iterations, which means that
532 the Jacobian matrix is evaluated only for the first non-linear iteration of the first stage of the ESDIRK3 scheme. By
533 doing so, it is observed that the linear system converges with the same number of iterations, which are not reported for
534 brevity. This shows, at least for this kind of problem, that the preconditioner does not lose its efficiency throughout
535 the stages of the same time step. Moreover, it confirms the powerful properties of the matrix-free iterative strategy,
536 which allow skipping of Jacobian evaluation without degrading the convergence of the linear solver. Obviously, from
537 the CPU time point of view, lagging the preconditioner improves the computational efficiency, since the matrix is
538 evaluated only once. This is reflected by the speed-up values (see the MFL result columns). We point out that the
539 maximum speed-up values are obtained for high orders, when the impact of the Jacobian assembly is large on the

540 overall CPU time, and for small time step sizes: in this case the conditioning of the matrix and the CPU time spent on
541 the iterative solution process reduce, and so the computational time saved by skipping the Jacobian assembly reflects
542 on the overall efficiency of the method. It is worth pointing out that by doing so, the matrix-free penalization at low
543 orders is reduced, while speed-up values in the range $[1.41, 3.10]$ are achieved for the $k = 6$, $\Delta t = 0.1$ case.

544 Similar observations hold true when the matrix-free approximation is employed within a multigrid strategy. Also
545 in this case, the problem has been solved using the same number of GMRES iterations with respect to the non-lagging
546 computation. However, from the CPU time view, a higher speed-up value is achieved both in serial and in parallel
547 runs thanks to the optimal multigrid scalability with respect to single-grid BILU preconditioning. The speed-up factor
548 for the largest polynomial order is now in the range $[2.56, 3.78]$ and $[2.77, 3.98]$ for the large and small time step,
549 respectively. Those speed-up values are larger than the ones obtained for a fully matrix-based implementation of the
550 DG discretization, see Table 5. In comparison to HDG, it can be seen that by using the Jacobian lagging option, larger
551 speed-up values are generally achieved for small time step sizes at high order. However, the use of $MG_{\text{full}}-p\text{HDG}$
552 may be preferred to $MG_{\text{full}}\text{-DG}$, even coupled with matrix-free and preconditioner lagging for the largest time step
553 employed, as better suitability for dealing with very stiff and high-order discretizations has been highlighted.

554 We remark that, for computational efficiency, we employ the matrix-free implementation of the iterative solver
555 only on the finest space of the multilevel iterative solution, similarly to what has been done in [12]. This choice is
556 consistent with the results obtained for the single-grid preconditioner in Table 6. In fact, the matrix-free iteration cost
557 is similar to that of a matrix-based one only for high orders, while it is larger for lower-order polynomials. It therefore
558 is appropriate to employ matrix-based smoothers on the coarse levels. When discretizing the equations using high
559 orders, this idea seems to work optimally. Note also that the overall memory footprint of the application is dominated
560 by the allocation of the block-Jacobi preconditioner for the finest space smoother, as shown in Figure 3.

561 7.5. Remarks

562 We summarize the main points of the previous section. First, we see clearly that having large time step sizes
563 maximizes the advantages of coupling HDG and p -multigrid, since the expense of using static condensation together
564 with a more powerful and expensive preconditioning strategy produce a faster solver only for high condition numbers
565 of the system, while the higher scalability of the multigrid algorithm as opposed to single-grid reflects on the overall
566 scalability of the solver. On the other hand, for small time step sizes, the computational time is dominated by the
567 Jacobian assembly and condensation for HDG, thus the CPU time as well as the parallel efficiency is dominated by
568 the matrix assembly.

569 For HDG, we demonstrate how the *mixed* and *primal* formulations provide comparable results in terms of error
570 levels by refining both in space and time, despite the fact that the *primal* form provides a one order lower convergence
571 rate for the gradient variable. From the algorithmic point of view, the statically condensed system is typically solved
572 using roughly the same number of iterations. However, the computational time is considerably lower for the $p\text{HDG}$
573 formulation, since it does not deal with the Jacobian entries, albeit local to each element, related to the state gradient.

574 For this reason, in the remainder of the work, only the *primal* formulation will be employed for benchmarking.
575 We stress that even for small time step sizes, the single-grid preconditioned *p*HDG solver performs fairly well in
576 comparison to the other solution strategies, showing the benefits of the approach for unsteady flow computations.

577 We remark that the current implementation of the parallelization strategy makes HDG less parallel efficient than
578 DG. This fact is ascribed to the higher amount of duplicate operations due to the integration over halo mesh elements
579 and faces created by the domain decomposition required for the static condensation of the interior degrees of freedom
580 of the Jacobian matrix. On the other hand, in DG the duplicate work involves only partition faces. Other implemen-
581 tation choices, such as those related to the minimization of the duplicate work on the partition boundaries, may be
582 considered in future works.

583 We finally point out that, despite being appealing, a matrix-free implementation of the hybridizable discontinuous
584 Galerkin method is not at all straightforward for obtaining satisfactory performance in CPU time [24], and thus its
585 development is beyond the scope of the present work.

586 8. Results on complex test cases

587 The second family of numerical experiments deals with the solution of two test cases: i) laminar flow over a
588 two-dimensional circular cylinder at $Re = 100$ and $M = 0.2$ [34]; and ii) the solution of the plunging motion of a
589 NACA 0012 airfoil at $Re = 1000$ and Mach number $M = 0.2$. The latter case is solved by using the ALE mesh motion
590 formulation introduced in [18]. Those results are devoted to extend the comparison to more complex unsteady flow
591 problems. In all the cases reported herein, the right-preconditioning approach is still employed such that the conver-
592 gence of the linear solver is not affected by changing the preconditioning operator, and therefore all the numerical
593 experiments are performed using a similar accuracy.

594 8.1. Circular cylinder

595 Laminar flow around a circular cylinder at Mach number $M = 0.2$ and Reynolds number $Re = 100$ has been
596 solved on a grid of $n_e = 960$ mesh elements using a \mathbb{P}_6 space discretization. Figure 7 shows a snapshot of the
597 computed Mach number contours. To integrate the governing equations in time, the four-stage, third-order explicit-
598 first-stage, diagonally-implicit Runge–Kutta method (ESDIRK3) was employed. The solution accuracy was assessed
599 by comparing with literature data [34]. To this end, Table 7 shows the averaged drag and lift coefficients, as well as the
600 Strouhal number of the body forces (C_d , C_l , St) for several temporal refinements on the same grid. The coefficients
601 were obtained by averaging a statistically-developed solution over ten shedding periods. An overall good agreement
602 has been found, while a temporal convergence can be observed by using a non-dimensional time step of $\Delta t \leq 0.25$.
603 It is well known [15] that the time step size greatly affects the iterative solution process, and therefore it has to be
604 taken into account to evaluate the performance. Presently, we use the largest time step size that yields both converged
605 body forces and Strouhal numbers, and this maximizes the efficiency of the solution strategy. Considering the results

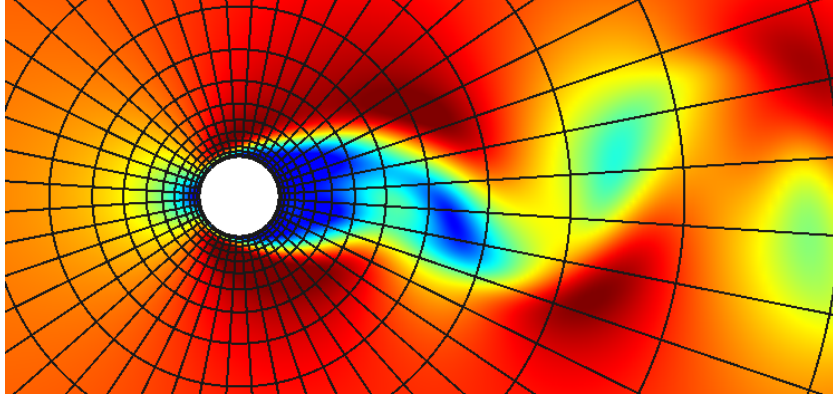


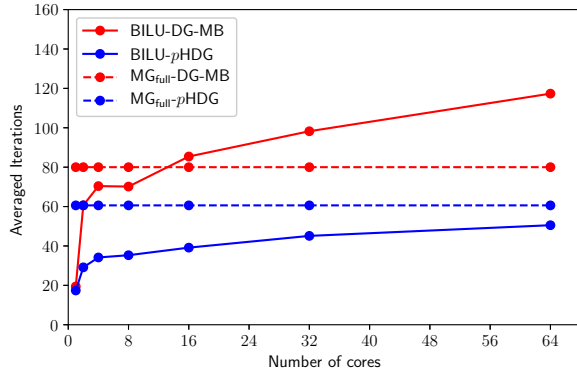
Figure 7: Laminar flow around a circular cylinder at $Re = 100$, $M = 0.2$. Mach number contours.

$(U/L)\Delta t$	C_d	C_l	St
0.5	1.3468	3.383e-03	0.16327
0.25	1.3519	-1.441e-03	0.16410
0.125	1.3527	-1.400e-04	0.16410
0.05	1.3528	-1.718e-04	0.16410
0.025	1.3528	-6.353e-06	0.16410

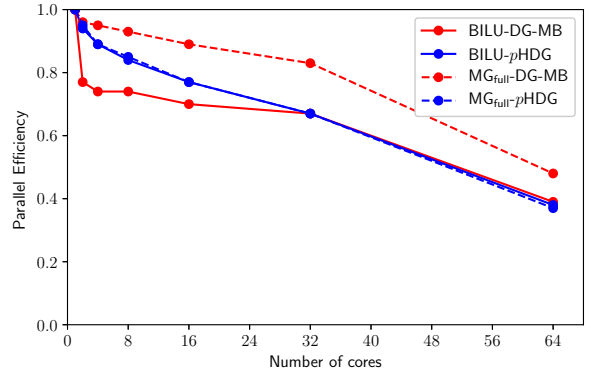
Table 7: Laminar vortex test case at $Re = 100$, $M = 0.05$. Time convergence rates for the DG and HDG spatial discretizations and the ESDIRK3 temporal scheme.

606 in Table 7, we choose $\Delta t = 0.25$. The parallel performance of the solution strategies introduced in the previous
607 sections is also assessed. To do so, a fully-developed flow field is integrated in time for 10 time steps to compute the
608 average number of GMRES iterations and the convergence rates during the non-linear solution. The computations are
609 performed on the range of 1 to 64 cores (n_p) on a platform based on two 16-core AMD Opteron processors arranged
610 in a two-processor per-node fashion, for a total of 32 cores per node. A fixed relative tolerance of 10^{-6} to stop the
611 GMRES solver is used, as well as an absolute tolerance of 10^{-5} for the Newton-Raphson method.

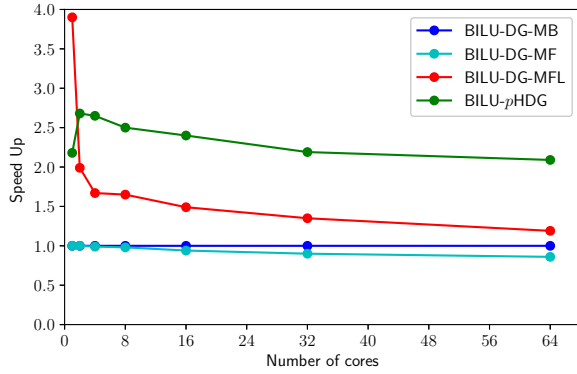
612 Figure 8 report the results of the computations. As observed for the convected vortex test case, the BILU pre-
613 conditioner shows an increase in the number of GMRES iterations with increasing number of processes in both the
614 DG and p HDG space discretizations. This can be observed in Figure 8(a), which reports the total number of itera-
615 tions performed on the finest level of the discretization, averaged throughout the solution process. This behavior is
616 attributed to the way the incomplete lower-upper factorization is performed, as it deals with the square, partition-wise
617 block of the iteration matrix. Therefore, the preconditioner effectiveness naturally decreases as n_p grows, since the
618 number of off-diagonal blocks neglected by the ILU increases with n_p . By switching from MB to MF, the number
619 of iterations remains the same and it is not reported for brevity. On the other hand, the use of a multigrid algorithm
620 with the provided settings results in an ideal algorithmic efficiency, *i.e.*, the number of fine space iterations do not



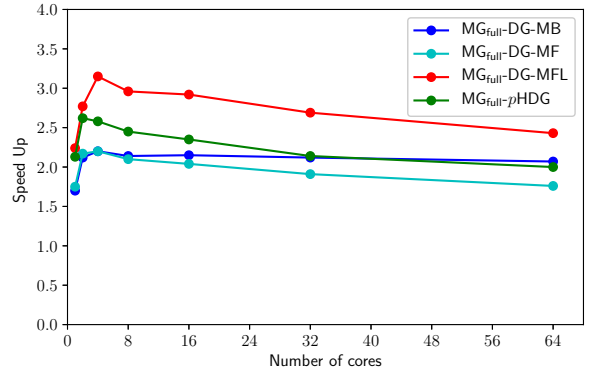
(a) Fine space iterations



(b) Parallel Efficiency



(c) BILU Speed Up



(d) MG_{full} Speed Up

Figure 8: L_2 solution error. Laminar vortex test case at $Re = 100$, $M = 0.05$. Convergence rates for the DG (dashed), $mHDG$ (solid) and $pHDG$ (crosses) discretizations versus theoretical estimates in space and time.

621 grow by increasing the number of parallel domains, for both DG and $pHDG$. In the multigrid case, the number of
 622 fine space iterations is computed by multiplying the number of outer FGMRES iterations with the number of pre and
 623 post smoothing on the finest space, which are reported in Table 3. Note that the computational cost of a multigrid fine
 624 space iteration is way cheaper than that of a BILU algorithm, since the element-by-element block Jacobi preconditioner
 625 is employed. It is worth also mentioning that in such a practical application the same numerical set-up as that
 626 of a DG solver reported in Table 3 has been employed to precondition the system, and that the method proves to work
 627 pretty well. Considering $pHDG$ with BILU preconditioning, the iterative solution process still suffers of algorithmic
 628 degradation by parallelizing the computation, despite the increase in the number of iterations being smaller than that
 629 observed for DG. In addition, we observe a lower number of iterations on average, meaning that the condition number
 630 of the matrix has been reduced by the static condensation. The superior parallel efficiency of the multigrid algorithm
 631 is clearly visible in Figure 8(b), which reports the strong scalability of the algorithms. While the use of BILU-DG-MB

632 drops the parallel efficiency below 80% by partitioning the computation on just two computational cores, the multi-
633 grid algorithm allows to keep good parallel efficiencies up to 32 cores. The parallel efficiency of the p HDG solver
634 lies somehow in between the two, and it does not change by modifying the preconditioner, which suggests that it is
635 dominated by the scalability of the static condensation/back solve portion of the application.

636 Figure 8(c) reports the speed up values of the matrix free (MF) and lagged matrix free (MFL) algorithms, which
637 updates the preconditioner operators only once per time step. The performance of the p HDG is also reported. In all
638 of the cases, the reference algorithm is BILU-DG-MB. An improvement in computational efficiency can be observed
639 for the MFL and p HDG cases, which speeds up values in favor of the latter. When switching to the full multigrid
640 preconditioner, see Figure 8(d), we observe an overall increase of performance for the DG algorithms. On the other
641 hand, the costs of p HDG, associated with the static condensation and back solve, cannot be offset simply by the use
642 of a better preconditioner and still dominate the percentage of the overall computational time. As a result, the MG_{full} -
643 DG-MFL strategy performs the best, with speed-up values in the range [2.24, 3.15]. Note that a similar performance
644 between MG_{full} -DG-MB, MG_{full} -DG-MF and p HDG is observed, but while the matrix based one requires a full
645 Jacobian matrix allocation, the matrix free and the hybridizable methods reduce considerably the memory footprint
646 according to what is reported in Figure 3.

647 As a final comparison, Table 8 reports the same test case solved using a grid obtained by splitting the quadrilateral
648 elements into triangles. Full-order basis functions were employed. Only the computation with $n_p = 64$ is reported.
649 It is worth pointing out that this space discretization reduces the total number of DoFs by the 23.8% with respect to
650 the previous one. By comparing the computational time and average number of GMRES iterations, similar speed-up
651 values are obtained using the DG-MB solver as well as p HDG. On the other hand, the DG-MF solver seems to be
652 slightly penalized with respect to the MB one, as the speed-up values of the computations drop by a factor between 7
653 to 15 percent. It is worth pointing out that the matrix-free penalization that arise in this case is in line to what reported
654 in previous studies [15, 12] using broken polynomial spaces, which reduce by increasing the number of DoFs per
655 element, for example in three-dimensional computations. On the other hand, a slightly larger speedup for p HDG
656 computations has been observed, which makes this solution strategy the most convenient from the CPU time point of
657 view.

658 8.2. Laminar flow around a heaving and pitching NACA 0012 airfoil

659 This test case is the CL1 (heaving and pitching airfoil) proposed for the 5th international workshop on high-order
660 CFD methods (HiOCFD5), see [33]. The simulation involves the compressible Navier–Stokes equations, with $\gamma = 1.4$,
661 $Pr = 0.72$, and constant viscosity, to simulate flow over a moving NACA 0012 airfoil. The airfoil is modified to obtain
662 a closed trailing edge via the equation

$$y(x) = \pm 0.6 \left(0.2969 \sqrt{x} - 0.1260x - 0.3516x^2 + 0.2843x^3 - 0.1036x^4 \right), \quad (52)$$

Preconditioner	BILU					
Case	DG-MB		DG-MF	DG-MFL	p HDG	
n_p	Time	IT_a	SU_{MB}	SU_{MB}	SU_{MB}	IT_a
64	322.26	129.050	0.80	1.00	2.40	49.750
Preconditioner	MG_{full}					
Case	DG-MB		DG-MF	DG-MFL	p HDG	
n_p	SU_{MB}	IT_a	SU_{MB}	SU_{MB}	SU_{MB}	IT_a
64	1.66	4.525	1.43	1.77	2.05	3.388

Table 8: Circular cylinder test case at $Re = 100$, $M = 0.2$, discretized using 1920 mesh elements with \mathbb{P}_6 full-order basis functions. Computational efficiency comparison of DG and p HDG solvers using BILU and p -multigrid, respectively. SU_{MB} stands for the speed-up factor relative to the matrix-based, BILU-DG computation and IT_a for the average number of GMRES iterations.

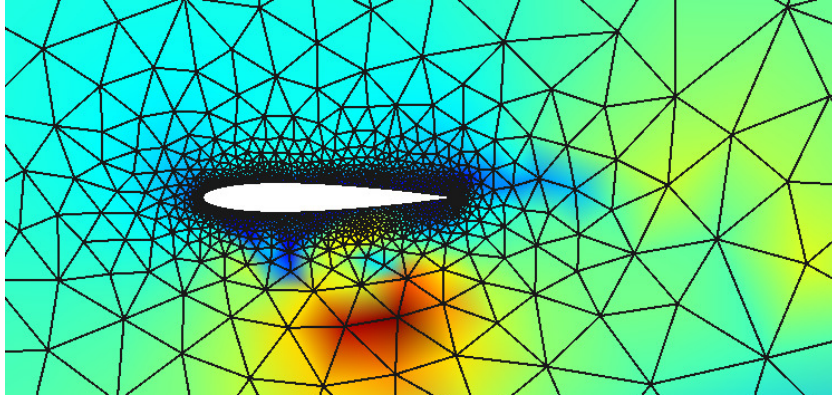


Figure 9: Laminar flow around a heaving NACA 0012 airfoil. $Re = 1000$, $M = 0.2$. Mach number contours at the final time $T = 2$.

663 with $x \in [0, 1]$. The initial condition is a steady state solution at a free-stream Mach number of $M = 0.2$ and a
664 Reynolds number $Re = 1000$. The motion is a pure plunging motion governed by the following function

$$h(t) = t^2(3 - t)/4, \quad (53)$$

665 and the output of interest is the total energy and vertical impulse exchanged between the airfoil and the fluid,

$$W = \int_T F_y(t) \dot{h}(t) dt, \quad I = \int_T F_y(t) dt, \quad (54)$$

666 where $F_y(t)$ is the vertical force computed on the airfoil surface, and $\dot{h}(t)$ is the time derivative of Eq. (53). In
667 particular, the output is evaluated at a non-dimensional time $T = 2$. The mesh consists of a triangulation of the
668 domain by $n_e = 2137$ elements, shown in Figure 9, and the solution approximation space is \mathbb{P}_6 .

669 The time step size of the simulation has been evaluated through a time-convergence analysis of the output quan-
670 tities, reported in Table 9. Table 10 compares the computational efficiency of the solution strategies in terms of the
671 speed-up factor, SU_{MB} , and the number of iterations, IT_a . Only the computation with the larger number of cores is
672 reported to show the efficiency on large and practical computations. The reference to compute the speed-up is the

$T/\Delta t$	W	I
20	-1.3860	-2.3667
40	-1.3839	-2.3444
80	-1.3837	-2.3391
160	-1.3837	-2.3378

Table 9: Time convergence analysis of the laminar flow around a plunging NACA 0012 airfoil. A satisfactory accuracy is observed for $T/\Delta t = 2$, where $T = 2$ is the length of the simulation and Δt is the time step size.

Preconditioner	BILU					
Case	DG-MB		DG-MF	DG-MFL	p HDG	
n_p	Time	IT_a	SU_{MB}	SU_{MB}	SU_{MB}	IT_a
64	900.19	136.119	0.69	0.81	2.35	49.750
Preconditioner	MG_{full}					
Case	DG-MB		DG-MF	DG-MFL	p HDG	
n_p	SU_{MB}	IT_a	SU_{MB}	SU_{MB}	SU_{MB}	IT_a
64	1.49	4.477	1.06	1.13	2.13	2.754

Table 10: Circular cylinder test case at $Re = 100$, $M = 0.2$, discretized using 2137 mesh elements with \mathbb{P}_6 full-order basis functions. Computational efficiency comparison of DG and p HDG solvers using BILU and p -multigrid, respectively. SU_{MB} stands for the speed-up factor referred to the matrix-based, BILU-DG computation and IT_a for the average number of GMRES iterations.

673 computational time of the matrix-based, BILU-DG solver. By switching from a matrix-based to a matrix-free im-
674 plementation, the computational strategy is penalized by higher computational cost of a single iterations, see the MF
675 column. By employing the Jacobian lagging (MFL), this penalization is reduced only slightly. On the other hand, the
676 p HDG implementation provides a solver which is more than twice as much as fast, while the system require a con-
677 siderably lower number of GMRES iterations with respect to the reference. The use of p -multigrid preconditioning
678 in a DG context provides a speed-up factor of about 1.49 for a fully matrix-based implementation, and the number of
679 iterations drops from 136 to around 4.4 on average. The use of matrix-free in this case still penalizes the solver, which
680 is about 12% faster for the MFL case. Multigrid preconditioning is shown to be robust enough to precondition the
681 linear system arising from the p HDG discretization, since it converges using an average of 2.754 iterations. However,
682 this gain is not reflected on the CPU time, which is slightly higher.

683 9. Conclusions

684 The paper compares, within the same framework, the computational efficiency of different high-order discontinu-
685 ous Galerkin implementations. The first involves a modal discontinuous Galerkin method coupled with matrix-based
686 and matrix-free iterative solvers, while the second one considers a hybridizable discontinuous Galerkin implementa-
687 tion, both in the *mixed* form, which allocates the components of the Jacobians related to the gradient variable, and
688 in the *primal* form, which forgoes separate approximation of the gradient variable and symmetrizes the discretization

689 by adding an additional adjoint-consistency term. The efficiency of the solution strategies is assessed by comparing
690 different single-level preconditioners as well as multilevel ones such as p -multigrid, on a variety of two-dimensional
691 test cases involving laminar viscous flows, including mesh motion, on meshes made by triangular and quadrilateral
692 elements. The effects of parallel efficiency and the use of different basis functions have also been considered. The
693 paper shows that the use of a matrix-free implementation of the iterative solver in the context of implicit discontinuous
694 Galerkin discretizations provides a memory footprint which is in line to that of an HDG method if a block-diagonal
695 preconditioner is employed within the smoother on the finest space. Moreover, the *primal* HDG method becomes
696 more efficient than the *mixed* one, having a lower number of non-zeros Jacobian entries, and it provides comparable
697 error levels. If compared to p HDG, the p -multigrid matrix-free solver is competitive in terms of CPU time when the
698 problems involve time marching with small time steps, since the preconditioner evaluation can be lagged. On the other
699 hand, HDG methods require expensive element-wise operations that become a bottleneck in those conditions. Finally,
700 a novel approximate-inherited p -multigrid strategy has also been introduced for HDG. Such a strategy is more robust
701 and efficient for different test cases and mesh types, and it is able to reduce considerably the number of iterations of the
702 solution process. However, this gain in number of iterations is not reflected in the CPU time of the solver. Future work
703 will be devoted to the validation of those strategies on stiff three-dimensional cases involving laminar and turbulent
704 flows, and possibly hybrid RANS-LES models.

705 Acknowledgements

706 M. Franciolini acknowledges Dr. Lorenzo Botti from the University of Bergamo for useful discussions and com-
707 ments. Funding received from the Department of Energy under grant DE-FG02-13ER26146/DE-SC0010341 is also
708 gratefully acknowledged.

709 References

- 710 [1] F. Bassi, A. Crivellini, D. A. Di Pietro, S. Rebay, An implicit high-order discontinuous Galerkin method for steady and unsteady incompressible flows, *Computers & Fluids* 36 (10) (2007) 1529–1546.
- 711 [2] F. Bassi, L. Botti, A. Colombo, A. Ghidoni, F. Massa, Linearly implicit Rosenbrock-type Runge–Kutta schemes applied to the Discontinuous Galerkin solution of compressible and incompressible unsteady flows, *Computers & Fluids* 118 (2015) 305–320.
- 712 [3] J. S. Hesthaven, T. Warburton, *Nodal discontinuous Galerkin methods: algorithms, analysis, and applications*, Springer Science & Business Media, 2007.
- 713 [4] L. Wang, D. J. Mavriplis, Implicit solution of the unsteady Euler equations for high-order accurate discontinuous Galerkin discretizations, *Journal of Computational Physics* 225 (2) (2007) 1994 – 2015.
- 714 [5] P.-O. Persson, J. Peraire, Newton-GMRES preconditioning for discontinuous Galerkin discretizations of the Navier–Stokes equations, *SIAM Journal on Scientific Computing* 30 (6) (2008) 2709–2733.
- 715 [6] L. T. Diosady, D. L. Darmofal, Preconditioning methods for discontinuous Galerkin solutions of the Navier–Stokes equations, *Journal of Computational Physics* 228 (11) (2009) 3917–3935.
- 716 [7] P. Birken, G. Gassner, M. Haas, C. D. Munz, Preconditioning for modal discontinuous Galerkin methods for unsteady 3D Navier–Stokes equations, *Journal of Computational Physics* 240 (2013) 20–35.
- 717
- 718
- 719
- 720
- 721
- 722
- 723

- 724 [8] Y. Saad, A flexible inner-outer preconditioned GMRES algorithm, *SIAM Journal on Scientific Computing* 14 (2) (1993) 461–469.
- 725 [9] K. J. Fidkowski, T. A. Oliver, J. Lu, D. L. Darmofal, p -Multigrid solution of high-order discontinuous Galerkin discretizations of the com-
726 pressible Navier–Stokes equations, *Journal of Computational Physics* 207 (1) (2005) 92–113.
- 727 [10] K. Shahbazi, D. J. Mavriplis, N. K. Burgess, Multigrid algorithms for high-order discontinuous Galerkin discretizations of the compressible
728 Navier–Stokes equations, *Journal of Computational Physics* 228 (21) (2009) 7917–7940.
- 729 [11] L. Botti, A. Colombo, F. Bassi, h -multigrid agglomeration based solution strategies for discontinuous Galerkin discretizations of incompress-
730 ible flow problems, *Journal of Computational Physics* 347 (2017) 382–415.
- 731 [12] M. Franciolini, L. Botti, A. Colombo, A. Crivellini, p -Multigrid matrix-free discontinuous Galerkin solution strategies for the under-resolved
732 simulation of incompressible turbulent flows, arXiv preprint arXiv:1809.00866 .
- 733 [13] A. Crivellini, F. Bassi, An implicit matrix-free discontinuous Galerkin solver for viscous and turbulent aerodynamic simulations, *Computers
734 & fluids* 50 (1) (2011) 81–93.
- 735 [14] M. Ceze, L. Diosady, S. M. Murman, Development of a high-order space-time matrix-free adjoint solver, in: 54th AIAA Aerospace Sciences
736 Meeting, 0833, 2016.
- 737 [15] M. Franciolini, A. Crivellini, A. Nigro, On the efficiency of a matrix-free linearly implicit time integration strategy for high-order Discontin-
738 uous Galerkin solutions of incompressible turbulent flows, *Computers & Fluids* 159 (2017) 276–294.
- 739 [16] B. Cockburn, O. Dubois, J. Gopalakrishnan, S. Tan, Multigrid for an HDG method, *IMA Journal of Numerical Analysis* 34 (4) (2014)
740 1386–1425.
- 741 [17] N. C. Nguyen, J. Peraire, B. Cockburn, An implicit high-order hybridizable discontinuous Galerkin method for nonlinear convection–diffusion
742 equations, *Journal of Computational Physics* 228 (23) (2009) 8841–8855.
- 743 [18] K. J. Fidkowski, A hybridized discontinuous Galerkin method on mapped deforming domains, *Computers & Fluids* 139 (2016) 80–91.
- 744 [19] 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.
- 745 [20] 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)
746 (2016) 192–220.
- 747 [21] M. Woopen, A. Balan, G. May, J. Schütz, A comparison of hybridized and standard DG methods for target-based hp-adaptive simulation of
748 compressible flow, *Computers & Fluids* 98 (2014) 3–16.
- 749 [22] J. Dahm, Toward Accurate, Efficient, and Robust Hybridized Discontinuous Galerkin Methods, Phd thesis, University of Michigan, 2017.
- 750 [23] P. Devloo, C. Faria, A. Farias, S. Gomes, A. Loula, S. Malta, On continuous, discontinuous, mixed, and primal hybrid finite element methods
751 for second-order elliptic problems, *International Journal for Numerical Methods in Engineering* .
- 752 [24] M. Kronbichler, W. A. Wall, A performance comparison of continuous and discontinuous Galerkin methods with fast multigrid solvers, *SIAM
753 Journal on Scientific Computing* 40 (5) (2018) A3423–A3448.
- 754 [25] J. Schütz, V. Aizinger, A hierarchical scale separation approach for the hybridized discontinuous Galerkin method, *Journal of Computational
755 and Applied Mathematics* 317 (2017) 500–509.
- 756 [26] D. N. Arnold, F. Brezzi, B. Cockburn, L. D. Marini, Unified analysis of discontinuous Galerkin methods for elliptic problems, *SIAM J.
757 Numer. Anal.* 39 (5) (2002) 1749–1779.
- 758 [27] P. L. Roe, Approximate Riemann solvers, parameter vectors, and difference schemes, *Journal of computational physics* 43 (2) (1981) 357–372.
- 759 [28] F. Bassi, S. Rebay, G. Mariotti, S. Pedinotti, M. Savini, A High-Order Accurate Discontinuous Finite Element Method for Inviscid and
760 Viscous Turbomachinery Flows, in: R. Decuyper, G. Dibelius (Eds.), 2nd European Conference on Turbomachinery Fluid Dynamics and
761 Thermodynamics, Technologisch Instituut, Antwerpen, Belgium, 99–108, 1997.
- 762 [29] F. Brezzi, G. Manzini, D. Marini, P. Pietra, A. Russo, Discontinuous Galerkin approximations for elliptic problems, *Numer. Methods Partial
763 Differential Equations* 16 (2000) 365–378.
- 764 [30] F. Bassi, S. Rebay, High-Order Accurate Discontinuous Finite Element Solution of the 2D Euler Equations, *J. Comput. Phys.* 138 (1997)
765 251–285.
- 766 [31] H. Bijl, M. H. Carpenter, V. N. Vatsa, C. A. Kennedy, Implicit time integration schemes for the unsteady compressible Navier–Stokes

- 767 equations: laminar flow, *Journal of Computational Physics* 179 (1) (2002) 313–329.
- 768 [32] P. F. Antonietti, M. Sarti, M. Verani, Multigrid algorithms for *hp*-discontinuous Galerkin discretizations of elliptic problems, *SIAM Journal*
769 *on Numerical Analysis* 53 (1) (2015) 598–618.
- 770 [33] 5th International Workshop on High-Order CFD Methods, <https://how5.cenaero.be/>, 2018.
- 771 [34] J. R. Meneghini, F. Saltara, C. L. R. Siqueira, J. A. Ferrari Jr, Numerical simulation of flow interference between two circular cylinders in
772 tandem and side-by-side arrangements, *Journal of Fluids and Structures* 15 (2) (2001) 327–350.