International Journal for Numerical Methods in Fluids

WILEY

RESEARCH ARTICLE OPEN ACCESS

The Matrix-Free Macro-Element Hybridized Discontinuous Galerkin Method for Steady and Unsteady Compressible Flows

Vahid Badrkhani | Marco F. P. ten Eikelder | René R. Hiemstra | Dominik Schillinger

Institute for Mechanics, Computational Mechanics Group, Technical University of Darmstadt, Darmstadt, Germany

Correspondence: Vahid Badrkhani (vahid.badrkhani@tu-darmstadt.de)

Received: 17 February 2024 | **Revised:** 4 October 2024 | **Accepted:** 5 November 2024

Funding: This work was supported by Deutsche Forschungsgemeinschaft (SCH 1249/2-1).

Keywords: compressible flows | hybridized discontinuous Galerkin method | iterative solvers | macro-elements | matrix-free | second-layer static condensation

ABSTRACT

The macro-element variant of the hybridized discontinuous Galerkin (HDG) method combines advantages of continuous and discontinuous finite element discretization. In this paper, we investigate the performance of the macro-element HDG method for the analysis of compressible flow problems at moderate Reynolds numbers. To efficiently handle the corresponding large systems of equations, we explore several strategies at the solver level. On the one hand, we utilize a second-layer static condensation approach that reduces the size of the local system matrix in each macro-element and hence the factorization time of the local solver. On the other hand, we employ a multi-level preconditioner based on the FGMRES solver for the global system that integrates well within a matrix-free implementation. In addition, we integrate a standard diagonally implicit Runge–Kutta scheme for time integration. We test the matrix-free macro-element HDG method for compressible flow benchmarks, including Couette flow, flow past a sphere, and the Taylor–Green vortex. Our results show that unlike standard HDG, the macro-element HDG method can operate efficiently for moderate polynomial degrees, as the local computational load can be flexibly increased via mesh refinement within a macro-element. Our results also show that due to the balance of local and global operations, the reduction in degrees of freedom, and the reduction of the global problem size and the number of iterations for its solution, the macro-element HDG method can be a competitive option for the analysis of compressible flow problems.

1 | Introduction

Discontinuous Galerkin (DG) methods [1] have been widely recognized for their favorable attributes in tackling conservation problems. They possess a robust mathematical foundation, the flexibility to employ arbitrary orders of basis functions on general unstructured meshes, and a natural stability property for convective operators [2–5]. About a decade ago, the hybridized discontinuous Galerkin (HDG) method was introduced, setting itself apart with distinctive features within the realm of DG methods [6]. The linear systems arising from the HDG method exhibit equivalence to two different systems: the first globally couples the numerical trace of the solution on element boundaries, leading to a significant reduction in degrees of freedom. The second couples the conserved quantities and their gradients at the element level, allowing for an element-by-element solution. Due to these advantages, numerous research endeavors have extended the application of the HDG method to address a diverse array

of initial boundary value problems [7–15]. Nevertheless, in the face of extensive computations, relying solely on the hybridization approach may fall short in overcoming limitations related to memory and time-to-solution [16, 17]. Ongoing research efforts [18–20] highlight these challenges and help provide motivation for the current work.

Compared to the DG method, the classical finite element formulation, also known as the continuous Galerkin (CG) method [21, 22], leads to fewer unknowns when the same mesh is used. Nevertheless, unstructured mesh generators frequently produce meshes with high vertex valency, resulting in intricate communication patterns during parallel runs on distributed memory systems, thereby affecting scalability [23, 24]. The HDG method, although generating a global system with a higher rank than the traditional statically condensed system in CG, demonstrates significantly reduced bandwidth at moderate polynomial degrees.

The macro-element variant of the HDG method, recently motivated by the authors for advection–diffusion problems [25], investigates a discretization strategy that amalgamates elements from both the CG and HDG approaches, enabling several distinctive features that sets it apart from the CG and the standard HDG method. First and foremost, by incorporating continuous elements within macro-elements, the macro-element HDG method effectively tackles the issue of escalating the number of degrees of freedom in standard HDG methods for a given mesh. Second, it provides an additional layer of flexibility in terms of tailoring the macro-element discretization and adjusting the associated local problem size to match the specifications of the available compute system and its parallel architecture. Third, it introduces a direct approach to adaptive local refinement, enabling uniform simplicial subdivision. Fourth, all local operations are embarrassingly parallel and automatically balanced. This eliminates the necessity for resorting to load balancing procedures external to the numerical method. Consequently, the macro-element HDG method is highly suitable for a matrix-free solution approach.

In this article, we extend the macro-element variant of the HDG method [25] to solving steady and unsteady compressible flow problems given by the Navier–Stokes equations. On the one hand, unlike in standard HDG schemes, the time complexity of the local solver in the macro-element HDG method increases significantly as the size of local matrices grows. Therefore, effectively managing the local matrix is pivotal for achieving scalability in the macro-element HDG algorithm. On the other hand, the macro-element HDG method can alleviate the accelerated growth of degrees of freedom that stems from the duplication of degrees of freedom along element boundaries. Recent research, as documented in References [18, 26, 27], has explored alternative techniques to tackle this challenge within the standard HDG method. These techniques pivot toward Schur complement approaches for the augmented system, addressing both local and global unknowns rather than solely focusing on the numerical trace. In our study, we integrate and synthesize some of these techniques to present an efficient, computationally economical, and memory-friendly version of the macro-element HDG algorithm.

The paper is organized as follows. In Section 2, we delineate the differential equations and elucidate the spatial and temporal discretizations employed in this study, leveraging the macro-element HDG method. Section 3 delves into the development of parallel iterative methods designed to solve the nonlinear system of equations resulting from the discretization process. Within this section, we focus on an inexact variant of Newton's method, standard static condensation, and an alternative second-layer static condensation approach. In Section 4, we present the matrix-free implementation utilized in this investigation, alongside a discussion of global solver options and the corresponding choices of preconditioners. Section 5 is devoted to showcasing the outcomes of numerical experiments conducted with our macro-element HDG variant, juxtaposed against the standard HDG method. These experiments encompass several test cases, ranging from three-dimensional steady to unsteady flow scenarios. A comprehensive comparative analysis is undertaken, evaluating the methods in terms of accuracy, iteration counts, computational time, and the number of degrees of freedom in the local/global solver, with a particular focus on the parallel implementation.

2 | The HDG Method on Macro-Elements for Compressible Flow

We first present the Navier–Stokes equations for modeling compressible flows. We proceed with a summary of the notation necessary for the description of the macro-element HDG method, following the notation laid out in our earlier work [25]. Next, we briefly describe the macro-element DG discretization in space and the implicit Runge–Kutta discretization in time.

2.1 | Governing Equations

The time dependent compressible Navier–Stokes equations are a non-linear conservation law system that can be written as follows:

$$\partial_{t^*}\rho^* + \nabla^* \cdot (\rho^* \mathbf{v}) = 0 \tag{1a}$$

$$\partial_{t^*}(\rho^* \mathbf{v}) + \nabla^* \cdot (\rho^* \mathbf{v}^* \otimes \mathbf{v}^*) + \nabla^* P^* - \nabla^* \cdot \boldsymbol{\tau}^* = 0 \tag{1b}$$

$$\partial_{t^*}\left(\rho^* \boldsymbol{E}^*\right) + \nabla\cdot^*\left(\rho^* \boldsymbol{H}^* \mathbf{v}^*\right) - \nabla^* \cdot \left(\boldsymbol{\tau}^* \mathbf{v}^* - \boldsymbol{\phi}^*\right) = 0 \tag{1c}$$

where ρ^* is the density, $\mathbf{v}^*$ the velocity, and E^* the total specific energy, subject to the initial conditions $\rho^* = \rho_0^*, \mathbf{v}^* = \mathbf{v}_0^*, E^* = E_0^*$. Furthermore, P^* is the pressure, $H^* = E^* + P^*/\rho^*$ the total specific enthalpy, and the shear stress and heat flux are respectively given by:

$$\boldsymbol{\tau}^* = \mu^*\left(\nabla \mathbf{v}^* + (\nabla^* \mathbf{v}^*)^T + \lambda(\nabla^* \cdot \mathbf{v}^*)\mathbf{I}\right) \tag{2a}$$

$$\boldsymbol{\phi}^* = -\kappa^* \nabla^* \boldsymbol{T}^* \tag{2b}$$

Here, μ is the dynamic viscosity, $\lambda = -2/d$, with spatial dimension d, T^* is the temperature, and κ^* the thermal conductivity. The thermodynamical variables P^*, ρ^* and T^* are related through an equation of state $P^* = P^*(\rho^*, T^*)$. In this work, we employ the (calorically) perfect gas equation of state: $P^* = \rho^* R^* T^*$, where R^* is the specific gas constant. The temperature T^* is related to the internal energy via the constitutive relation $e^* = c_v^* T^*$, where γ

is the ratio of specific heats, and $c_v^* = R^*/(\gamma - 1)$ is the specific heat at constant volume. The total specific energy is given by $E^* = e^* + \mathbf{v}^* \cdot \mathbf{v}^*/2$. We introduce a rescaling of the system based on the following dimensionless variables:

$$\begin{aligned} \mathbf{x} &= \frac{\mathbf{x}^*}{X_0}, \quad t = \frac{t^* c_\infty}{X_0}, \quad \mathbf{v} = \frac{\mathbf{v}^*}{c_\infty}, \quad \rho = \frac{\rho^*}{\rho_\infty}, \\ P &= \frac{P^*}{\rho_\infty c_\infty^2}, \quad e = \frac{e^*}{c_\infty^2}, \quad E = \frac{E^*}{c_\infty^2} \\ T &= \frac{T^*}{T_\infty}, \quad \mu = \frac{\mu^*}{\mu_\infty}, \quad \kappa = \frac{\kappa^*}{\kappa_\infty}, \\ R &= \frac{R^*}{c_{p\infty}} \quad c_v = \frac{c_v^*}{c_{p\infty}} \quad c_p = \frac{c_p^*}{c_{p\infty}} \end{aligned} \tag{3}$$

where X_0 is a characteristic length, c_∞ the free-stream speed of sound, ρ_∞ the free-stream density, T_∞ the free-stream temperature and μ_∞ the free-stream dynamic viscosity. The non-dimensional system may be written in the compact form [10, 13]:

$$\frac{\partial \mathbf{u}}{\partial t} + \nabla \cdot (\mathbf{F}(\mathbf{u}) + \mathbf{G}(\mathbf{u}, \mathbf{q})) = 0 \tag{4a}$$

$$\mathbf{q} - \nabla \mathbf{u} = 0 \tag{4b}$$

subject to the initial condition $\mathbf{u} = \mathbf{u}_0$. Here, $\mathbf{u} = (\rho, \rho\mathbf{v}, \rho E)^T$ denotes the state vector of dimension n_s. It is of dimension $d + 2$, where d is the spatial dimension. The inviscid and viscous fluxes $\mathbf{F} = \mathbf{F}(\mathbf{u})$ and $\mathbf{G} = \mathbf{G}(\mathbf{u}, \mathbf{q})$, respectively, are given by:

$$\mathbf{F}(\mathbf{u}) = [\rho\mathbf{v}, \rho\mathbf{v} \otimes \mathbf{v} + P\mathbf{I}, \rho\mathbf{v}H]^T \tag{5a}$$

$$\mathbf{G}(\mathbf{u}, \mathbf{q}) = -[0, \tau, \tau\mathbf{v} - \boldsymbol{\phi}]^T \tag{5b}$$

The shear stress and heat flux take the form:

$$\tau = \frac{1}{Re_{c_\infty}} \left(\mu \left(\nabla\mathbf{v} + (\nabla\mathbf{v})^T + \lambda(\nabla \cdot \mathbf{v})\mathbf{I} \right) \right) \tag{6a}$$

$$\boldsymbol{\phi} = -\frac{1}{Re_{c_\infty} \, \mathrm{Pr}} \frac{\kappa}{R} \nabla T \tag{6b}$$

Since a mixed method is used, vectors $\nabla\mathbf{v}$ and ∇T are derived from $\mathbf{q}$. Also, the acoustic Reynolds number and Prandtl number are given by:

$$Re_{c_\infty} = \frac{\rho_\infty c_\infty X_0}{\mu_\infty} \tag{7a}$$

$$\mathrm{Pr} = \frac{c_{p\infty} \mu_\infty}{\kappa_\infty} \tag{7b}$$

Finally, the thermodynamical relations are in non-dimensional form as follows: $\gamma P = \rho T$, $e = c_v T$, $c_v = R/(\gamma - 1)$, and $E = e + \mathbf{v} \cdot \mathbf{v}/2$.

2.2 | Finite Element Mesh and Spaces on Macro-Element

We denote by $\mathcal{T}_h$ a collection of disjoint regular elements K that partition Ω, and set $\partial\mathcal{T}_h := \{\partial K : K \in \mathcal{T}_h\}$ to be the collection of the boundaries of the elements in $\mathcal{T}_h$. For an element K of the collection $\mathcal{T}_h$, $e = \partial K \cap \partial\Omega$ is a boundary face if its $d - 1$ Lebesgue measure is nonzero. For two elements K^+ and K^- of $\mathcal{T}_h$, $e = \partial K^+ \cap \partial K^-$ is the interior face between K^+ and K^- if its $d - 1$ Lebesgue measure is nonzero. We denote by $\varepsilon^{\mathrm{Int}}$ and ε^{∂} the set of interior and boundary faces, respectively, and we define $\varepsilon_h := \varepsilon^{\mathrm{Int}} \cup \varepsilon^{\partial}$ as the union of interior and boundary faces.

Figure 1 illustrates the degree of freedom structure of a macro-element. We first partition the domain in macro-elements (blue lines). Each macro-element is then further split into standard finite elements, which define standard C^0-continuous basis functions. From a global viewpoint, these basis functions are discontinuous across macro-element interfaces. The macro-elements are therefore coupled together in an HDG sense. Consequently, the trace variable is defined only on the macro-element interfaces. It is easy to see that the macro-element HDG method contains the standard HDG method as a special case, when each macro-element contains only one standard finite element. In this work, we focus on simplicial meshes obtained by splitting each macro-simplex into a regular number of triangles or tetrahedral elements. But we note that our methodology can directly be transferred to other element types such as quadrilaterals or hexahedra.

Let $\mathcal{P}_p(D)$ denote the set of polynomials of degree at most p on a domain D and let $L^2(D)$ be the space of square-integrable functions on D. Our macro-element variant of the HDG method uses patches of standard C^0 continuous elements that are discontinuous only across patch boundaries. Hence, on macro-elements, we use continuous piece-wise polynomials. We introduce the following finite element spaces:

$$\mathcal{V}_h^k = \{\boldsymbol{w} \in \left[C^0(K)\right]^{n_s} : \boldsymbol{w}\big|_K \in \left[\mathcal{P}_p(K)\right]^{n_s} \quad \forall K \in \mathcal{T}_h\} \tag{8a}$$

$$\mathcal{Q}_h^k = \{\boldsymbol{r} \in \left[C^0(K)\right]^{n_s \times d} : \boldsymbol{r}\big|_K \in \left[\mathcal{P}_p(K)\right]^{n_s \times d} \quad \forall K \in \mathcal{T}_h\} \tag{8b}$$

$$\boldsymbol{\mathcal{M}}_h^k = \{\boldsymbol{\mu} \in \left[L^2(\varepsilon_h)\right]^{n_s} : \boldsymbol{\mu}\big|_e \in \left[\mathcal{P}_p(e)\right]^{n_s} \quad \forall e \in \varepsilon_h\} \tag{8c}$$

$$\boldsymbol{\mathcal{M}}_h^k = \{\mu \in L^2(\varepsilon_h) : \mu\big|_e \in \mathcal{P}_p(e) \quad \forall e \in \varepsilon_h\} \tag{8d}$$

Next, we define several inner products associated with these finite element spaces. In particular, given $w, v \in \mathcal{V}_h^k$, $\boldsymbol{w}, \boldsymbol{v} \in \mathcal{V}_h^k$ and $\boldsymbol{W}, \boldsymbol{V} \in \mathcal{Q}_h^k$ we write:

$$\begin{aligned} (w, v)_{\mathcal{T}_h} &= \sum_{K \in \mathcal{T}_h} (w, v)_K = \sum_{K \in \mathcal{T}_h} \int_K wv_K, \quad (\boldsymbol{w}, \boldsymbol{v})_{\mathcal{T}_h} = \sum_{i=1}^{n_s} \left(w_i, v_i\right)_{\mathcal{T}_h}, \\ (\boldsymbol{W}, \boldsymbol{V})_{\mathcal{T}_h} &= \sum_{i=1}^{n_s} \sum_{j=1}^{d} \left(W_{ij}, V_{ij}\right)_{\mathcal{T}_h} \end{aligned} \tag{9}$$

The corresponding inner product for functions in the trace spaces are given by:

$$\langle \eta, \zeta \rangle_{\partial\mathcal{T}_h} = \sum_{K \in \mathcal{T}_h} \langle \eta, \zeta \rangle_{\partial K} = black \sum_{K \in \mathcal{T}_h} \int_{\partial K} \eta\zeta, \quad \langle \boldsymbol{\eta}, \boldsymbol{\zeta} \rangle_{\partial\mathcal{T}_h} = \sum_{i=1}^{n_s} \langle \eta_i, \zeta_i \rangle_{\partial\mathcal{T}_h} \tag{10}$$

for all $\eta, \zeta \in \mathcal{M}_h^k$ and $\boldsymbol{\eta}, \boldsymbol{\zeta} \in \boldsymbol{\mathcal{M}}_h^k$.

2.3 | Macro-Element Discretization

We seek an approximation $\left(\mathbf{q}_h(t), \mathbf{u}_h(t), \hat{\mathbf{u}}_h(t)\right) \in \mathcal{Q}_h^k \times \mathcal{V}_h^k \times \boldsymbol{\mathcal{M}}_h^k$ such that

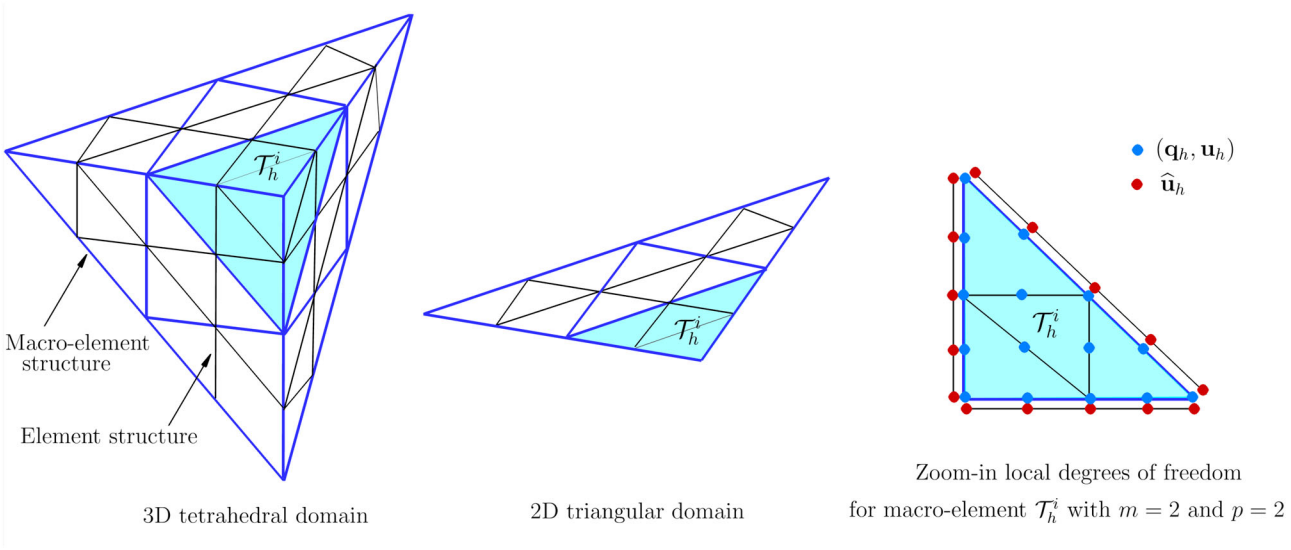


FIGURE 1 | Illustration of the macro-element HDG discretization of a 3D tetrahedral domain and a triangular domain in 2D for the case of $p = 2$. The blue lines represent the boundaries of macro-elements, while the black lines represent the boundaries of the C^0-continuous finite elements within each macro-element. We note that we refer to the set of standard finite elements within one macro-element as a (macro-element) patch. [Colour figure can be viewed at wileyonlinelibrary.com]

$$\left(\mathbf{q}_h, \boldsymbol{r}\right)_{\mathcal{T}_h} + \left(\mathbf{u}_h, \nabla \cdot \boldsymbol{r}\right)_{\mathcal{T}_h} - \left\langle \widehat{\mathbf{u}}_h, \boldsymbol{r}\boldsymbol{n}\right\rangle_{\partial\mathcal{T}_h} = 0 \tag{11a}$$

$$\left(\frac{\partial \mathbf{u}_h}{\partial t}, \boldsymbol{w}\right)_{\mathcal{T}_h} - \left(\mathbf{F}_h\left(\mathbf{u}_h\right) + \mathbf{G}_h\left(\mathbf{u}_h\mathbf{q}_h\right), \nabla \boldsymbol{w}\right)_{\mathcal{T}_h} + \left\langle \widehat{\mathbf{F}}_h + \widehat{\mathbf{G}}_h, \boldsymbol{w} \otimes \boldsymbol{n}\right\rangle_{\partial\mathcal{T}_h} + \sum_{e \in K} \left((\mathbf{A} \cdot \nabla)\boldsymbol{w}, \tau_{\text{SUPG}}\mathbf{R}\right)_{\mathcal{T}_h} = 0 \tag{11b}$$

$$\left\langle \widehat{\mathbf{F}}_h + \widehat{\mathbf{G}}_h, \boldsymbol{\mu} \otimes \boldsymbol{n}\right\rangle_{\partial\mathcal{T}_h \setminus \partial\Omega} + \left\langle \widehat{\mathbf{B}}_h\left(\widehat{\mathbf{u}}_h, \mathbf{u}_h, \mathbf{q}_h\right), \boldsymbol{\mu} \otimes \boldsymbol{n}\right\rangle_{\partial\Omega} = 0 \tag{11c}$$

for all $(\mathbf{r}, \mathbf{w}, \boldsymbol{\mu}) \in \mathcal{Q}_h^k \times \mathcal{V}_h^k \times \mathcal{M}_h^k$ and all $t \in (0, T]$. The boundary trace operator $\widehat{\mathbf{B}}_h\left(\widehat{\mathbf{u}}_h, \mathbf{u}_h, \mathbf{q}_h\right)$ imposes the boundary conditions along $\partial\Omega$ exploiting the hybrid variable [11]. We take the interior numerical fluxes of the form:

$$\left(\widehat{\mathbf{F}}_h + \widehat{\mathbf{G}}_h\right)\boldsymbol{n} = \left(\mathbf{F}_h\left(\widehat{\mathbf{u}}_h\right) + \mathbf{G}_h\left(\widehat{\mathbf{u}}_h, \mathbf{q}_h\right)\right)\boldsymbol{n} + \mathbf{S}\left(\mathbf{u}_h, \widehat{\mathbf{u}}_h\right)\left(\mathbf{u}_h - \widehat{\mathbf{u}}_h\right)\boldsymbol{n} \quad \text{on } \partial\mathcal{T}_h \tag{12}$$

where $\boldsymbol{n}$ denotes the outward unit normal vector. The latter member of (12) involves the stabilization tensor $\mathbf{S}$ that enhances the stability of the HDG method. The inviscid and viscous components of the system are separately stabilized by means of the decomposition [10]:

$$\mathbf{S} = \mathbf{S}^{\text{inv}} + \mathbf{S}^{\text{vis}} \tag{13a}$$

$$\mathbf{S}^{\text{inv}} = \lambda_{\max}\mathbf{I} \tag{13b}$$

$$\mathbf{S}^{\text{vis}} = \frac{1}{Re}\ \text{diag}\left(0, \Upsilon, 1/\left[(\gamma - 1)M_\infty^2\ \text{Pr}\right]\right) \tag{13c}$$

where $\mathbf{I}$ is the identity matrix, Υ is a $(n_s - 2)$-dimensional vector of ones, and M_∞ is the free stream Mach number. The inviscid stabilization tensor $\mathbf{S}^{\text{inv}}$ is a local Lax-Friedrich stabilization in which $\lambda_{\max}$ is the maximum absolute eigenvalue of the Jacobian matrix $\partial\mathbf{F}_h/\partial\widehat{\mathbf{u}}_h$. Finally, the latter term in (11b) is a standard residual-based streamline-upwind-Petrov Galerkin (SUPG) stabilization term [28, 29] where $\mathbf{R}$ is the local residual of the governing Equation (11b), $\mathbf{A}$ is the Jacobian of the inviscid flux and τ_{SUPG} is the stabilization matrix. The precise definitions of the $\mathbf{A}$ and τ_{SUPG} are described in References [30–33]. This stabilization term is active within the C^0-macro-elements, and further improves the stability for a wide range of Reynolds and Mach numbers.

2.4 | Temporal Integration

We adopt the s-stage diagonally implicit Runge–Kutta (DIRK) time-discretization scheme [34]. Due to their higher-order accuracy and wide stability range, DIRK methods are widely employed temporal integration schemes for stiff systems. We refer the interested reader to comprehensive reviews of higher-order time integration methods suitable for HDG methods [9, 15]. The Butcher's table associated with the DIRK method can be written as:

$$\begin{array}{c|cccc} c_1 & a_{11} & 0 & \cdots & 0 \\ c_2 & a_{21} & a_{22} & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ c_s & a_{s1} & a_{s2} & \cdots & a_{ss} \\ \hline & b_1 & b_2 & \cdots & b_s \end{array} \tag{14}$$

where we assume the matrix a_{ij} to be non-singular, c_i and b_i are numbers that depend on DIRK type [34]. Denoting the time level by n, we have $n = s(l-1) + i$, where s is the number of stages, l the current time step, and $i = 1, \ldots, s$ the current stage within the current time step. Let d_{ij} denote the inverse of a_{ij}, and let $\left(\mathbf{q}_h^{n,i}, \mathbf{u}_h^{n,i}, \widehat{\mathbf{u}}_h^{n,i}\right)$ be the intermediate solutions of $\left(\mathbf{q}_h\left(t^{n,i}\right), \mathbf{u}_h\left(t^{n,i}\right), \widehat{\mathbf{u}}_h\left(t^{n,i}\right)\right)$ at the discrete time $t^{n,i} = t_n + c_i\Delta t^n$, where $1 \le i \le s$. The numerical solution $\mathbf{u}_h^{n+1}$ at the time level

$n+1$ given by the DIRK method is computed as follows:

$$\mathbf{u}_h^{n+1} = \left(1 - \sum_{i=1}^{s} e_j\right)\mathbf{u}_h^n + \sum_{j=1}^{s} \mathbf{u}_h^{n,j} \tag{15}$$

where $e_j = \sum_{j=1}^{s} b_i d_{ij}$. The intermediate solutions are determined as follows: we search for $\left(\mathbf{q}_h^{n,i}, \mathbf{u}_h^{n,i}, \hat{\mathbf{u}}_h^{n,i}\right) \in \mathcal{Q}_h^k \times \mathcal{V}_h^k \times \mathcal{M}_h^k$ such that the following is satisfied:

$$\left(\mathbf{q}_h^{n,i}, \boldsymbol{r}\right)_{\mathcal{T}_h} + \left(\mathbf{u}_h^{n,i}, \nabla \cdot \boldsymbol{r}\right)_{\mathcal{T}_h} - \left\langle \hat{\mathbf{u}}_h^{n,i}, \boldsymbol{r}\boldsymbol{n}\right\rangle_{\partial\mathcal{T}_h} = 0 \tag{16a}$$

$$\left(\frac{\sum_{j=1}^{s} d_{ij}\left(\mathbf{u}_h^{n,j} - \mathbf{u}_h^n\right)}{\Delta t^n}, \boldsymbol{w}\right)_{\mathcal{T}_h} - \left(\mathbf{F}_h^{n,i} + \mathbf{G}_h^{n,i}, \nabla \boldsymbol{w}\right)_{\mathcal{T}_h} + \left\langle \hat{\mathbf{F}}_h^{n,i} + \hat{\mathbf{G}}_h^{n,i}, \boldsymbol{w} \otimes \mathbf{n}\right\rangle_{\partial\mathcal{T}_h} + \sum_{e \in K}\left(\left(\mathbf{A}^{n,i} \cdot \nabla\right)\mathbf{w}, \tau_{\text{SUPG}}^{n,i}\mathbf{R}^{n,i}\right)_{\mathcal{T}_h} = 0 \tag{16b}$$

$$\left\langle \hat{\mathbf{F}}_h^{n,i} + \hat{\mathbf{G}}_h^{n,i}, \boldsymbol{\mu} \otimes \mathbf{n}\right\rangle_{\partial\mathcal{T}_h \setminus \partial\Omega} + \left\langle \hat{\mathbf{B}}_h^{n,i}\left(\hat{\mathbf{u}}_h^{n,i}, \mathbf{u}_h^{n,i}, \mathbf{q}_h^{n,i}\right), \boldsymbol{\mu} \otimes \mathbf{n}\right\rangle_{\partial\Omega} = 0 \tag{16c}$$

for all $(\boldsymbol{r}, \boldsymbol{w}, \boldsymbol{\mu}) \in \mathcal{Q}_h^k \times \mathcal{V}_h^k \times \mathcal{M}_h^k$. Once $\mathbf{u}_h^{n+1}$ has been determined as above, we search for $\left(\mathbf{q}_h^{n+1}, \hat{\mathbf{u}}_h^{n+1}\right) \in \mathcal{Q}_h^k \times \mathcal{M}_h^k$ such that

$$\left(\mathbf{q}_h^{n+1}, \mathbf{r}\right)_{\mathcal{T}_h} + \left(\mathbf{u}_h^{n+1}, \nabla \cdot \mathbf{r}\right)_{\mathcal{T}_h} - \left\langle \hat{\mathbf{u}}_h^{n+1}, \mathbf{r}\mathbf{n}\right\rangle_{\partial\mathcal{T}_h} = 0 \tag{17a}$$

$$\left\langle \hat{\mathbf{F}}_h^{n+1} + \hat{\mathbf{G}}_h^{n+1}, \mu \otimes \mathbf{n}\right\rangle_{\partial\mathcal{T}_h \setminus \partial\Omega} + \left\langle \hat{\mathbf{B}}_h^{n+1}\left(\hat{\mathbf{u}}_h^{n+1}, \mathbf{u}_h^{n+1}, \mathbf{q}_h^{n+1}\right), \mu \otimes \mathbf{n}\right\rangle_{\partial\Omega} = 0 \tag{17b}$$

for all $(\mathbf{r}, \mathbf{w}, \boldsymbol{\mu}) \in \mathcal{Q}_h^k \times \mathcal{V}_h^k \times \mathcal{M}_h^k$.

Remark 1. The system (16) can be advanced in time without solving (17). Hence, in practice we only solve (17) at the time steps that we need $\left(\mathbf{q}_h^{n+1}, \hat{\mathbf{u}}_h^{n+1}\right)$ for post-processing purposes. Finally, it is worth mentioning that certain specific DIRK schemes, such as the strongly s-stable DIRK (2, 2) and DIRK (3, 3) schemes, have the unique property that $c_s = 1$. As a consequence, (17) becomes identical to (16) at final stage $i = s$. As a result, these particular DIRK schemes do not require the solution of Equation (17).

3 | Parallel Iterative Solvers

We construct parallel iterative methods for the solution of the nonlinear system of Equation (16). First, in Section 3.1, we linearize the nonlinear global problem by means of an inexact Newton method. Next, we propose two different options for the solution of the linear system; standard static condensation in Section 3.2, and an alternative second-layer static condensation approach in Section 3.3.

3.1 | Nonlinear Solver: Inexact Newton Method

At any given (sub) time step n, the nonlinear system of Equation (16) can be written as

$$R_Q\left(\mathbf{q}_h^n, \mathbf{u}_h^n, \hat{\mathbf{u}}_h^n\right) = 0 \tag{18a}$$

$$R_U\left(\mathbf{q}_h^n, \mathbf{u}_h^n, \hat{\mathbf{u}}_h^n\right) = 0 \tag{18b}$$

$$R_{\hat{U}}\left(\mathbf{q}_h^n, \mathbf{u}_h^n, \hat{\mathbf{u}}_h^n\right) = 0 \tag{18c}$$

where R_Q, R_U, and $R_{\hat{U}}$ are the discrete nonlinear residuals associated to Equations 11a, 11b, and 11c, respectively.

To address the nonlinear system (18), we use pseudo-transient continuation [35, 36], which is an inexact Newton method. The procedure requires an adaptation algorithm of the pseudo time step size to complete the method. In this study, we employ the successive evolution relaxation (SER) algorithm [37] with the following parameters:

$$\Delta\tau^0 = \tau_{\text{init}}, \quad \Delta\tau^{m+1} = \min\left(\Delta\tau^m \frac{\|R_U\|_{L^2}^{m+1}}{\|R_U\|_{L^2}^{m}}, \tau_{\max}\right) \tag{19}$$

Here, m is the iteration step for the pseudo-transient continuation. In this study, if not otherwise specified, $\tau_{\text{init}} = 1.0$ and $\tau_{\max} = 10^8$. By linearizing (18) with respect to the solution $\left(\mathbf{q}_h^{m,n}, \mathbf{u}_h^{m,n}, \hat{\mathbf{u}}_h^{m,n}\right)$ at the Newton step $m = 0, 1, \ldots$, we obtain the subsequent linear system:

$$\left[\begin{array}{cc|c} \mathbf{A}_{qq}^{m,n} & \mathbf{A}_{qu}^{m,n} & \mathbf{A}_{q\hat{u}}^{m,n} \\ \mathbf{A}_{uq}^{m,n} & \mathbf{A}_{uu}^{m,n} & \mathbf{A}_{u\hat{u}}^{m,n} \\ \hline \mathbf{A}_{\hat{u}q}^{m,n} & \mathbf{A}_{\hat{u}u}^{m,n} & \mathbf{A}_{\hat{u}\hat{u}}^{m,n} \end{array}\right]\left[\begin{array}{c} \Delta Q^{m,n} \\ \Delta U^{m,n} \\ \hline \Delta\hat{U}^{m,n} \end{array}\right] = -\left[\begin{array}{c} R_Q^{m,n} \\ R_U^{m,n} \\ \hline R_{\hat{U}}^{m,n} \end{array}\right] \tag{20}$$

where $\Delta Q^{m,n}$, $\Delta U^{m,n}$, and $\Delta\hat{U}^{m,n}$ are the update of the vector of degrees of freedom of the discrete field solutions $\mathbf{q}_h^{m,n}$, $\mathbf{u}_h^{m,n}$, and $\hat{\mathbf{u}}_h^{m,n}$, respectively. The next Newton update of these solution fields is defined as

$$\left(\mathbf{q}_h^{m+1,n}, \mathbf{u}_h^{m+1,n}, \hat{\mathbf{u}}_h^{m+1,n}\right) := \left(\mathbf{q}_h^{m,n}, \mathbf{u}_h^{m,n}, \hat{\mathbf{u}}_h^{m,n}\right) + \left(\Delta\mathbf{q}^{m,n}, \Delta\mathbf{u}^{m,n}, \Delta\hat{\mathbf{u}}^{m,n}\right) \tag{21}$$

Newton iterations are repeated until the norm of the full residual vector $R_F := \left(R_Q, R_U, R_{\hat{U}}\right)$ is smaller than a specified tolerance.

3.2 | Linear Solver: Static Condensation

The first method that we discuss for the solution of the linear system of Equation (20) is static condensation. First, we discuss the system of equations for each macro element, and subsequently the global system. Eliminating both ΔQ and ΔU in an element-by-element fashion, as mentioned earlier, can be achieved using the first two equations in (20). As a result, we compute for each macro-element $\mathcal{T}_i$ the solution updates $\Delta Q^{\mathcal{T}_i}$ and $\Delta U^{\mathcal{T}_i}$ as

$$\begin{bmatrix} \Delta Q^{\mathcal{T}_i} \\ \Delta U^{\mathcal{T}_i} \end{bmatrix} = \left(\mathbf{A}_{\text{local}}^{\mathcal{T}_i}\right)^{-1}\left(-\begin{bmatrix} R_Q^{\mathcal{T}_i} \\ R_U^{\mathcal{T}_i} \end{bmatrix} - \begin{bmatrix} \mathbf{A}_{q\hat{u}}^{\mathcal{T}_i} \\ \mathbf{A}_{u\hat{u}}^{\mathcal{T}_i} \end{bmatrix}\Delta\hat{U}^{\Gamma_i}\right) \tag{22}$$

where the block structured local matrices are given by:

$$\mathbf{A}_{\text{local}}^{\mathcal{T}_i} = \begin{bmatrix} \mathbf{A}_{qq}^{\mathcal{T}_i} & \mathbf{A}_{qu}^{\mathcal{T}_i} \\ \mathbf{A}_{uq}^{\mathcal{T}_i} & \mathbf{A}_{uu}^{\mathcal{T}_i}. \end{bmatrix} \tag{23}$$

Next, we consider the global system of equations. We note that the matrix $\left[\mathbf{A}_{qq}\mathbf{A}_{qu};\mathbf{A}_{uq}\mathbf{A}_{uu}\right]$ has a block-diagonal structure. This permits expressing ΔU and ΔQ in terms of $\Delta\widehat{U}$. By eliminating ΔQ and ΔU from (20), we obtain the globally coupled reduced system of linear equations:

$$\widehat{\mathbf{A}}\,\Delta\widehat{U} = \widehat{\mathbf{b}} \tag{24}$$

which has to be solved in every Newton iteration. The macro-element contributions to the global reduced system are given by:

$$\widehat{\mathbf{A}}^{\mathcal{T}_i} = \mathbf{A}^{\mathcal{T}_i}_{\hat{u}\hat{u}} - \begin{bmatrix}\mathbf{A}^{\mathcal{T}_i}_{\hat{u}q} & \mathbf{A}^{\mathcal{T}_i}_{\hat{u}u}\end{bmatrix}\left(\mathbf{A}^{\mathcal{T}_i}_{\text{local}}\right)^{-1}\begin{bmatrix}\mathbf{A}^{\mathcal{T}_i}_{q\hat{u}}\\ \mathbf{A}^{\mathcal{T}_i}_{u\hat{u}}\end{bmatrix} \tag{25a}$$

$$\widehat{\mathbf{b}}^{\mathcal{T}_i} = -R^{\mathcal{T}_i}_{\hat{U}} + \begin{bmatrix}\mathbf{A}^{\mathcal{T}_i}_{\hat{u}q} & \mathbf{A}^{\mathcal{T}_i}_{\hat{u}u}\end{bmatrix}\left(\mathbf{A}^{\mathcal{T}_i}_{\text{local}}\right)^{-1}\begin{bmatrix}R^{\mathcal{T}_i}_{Q}\\ R^{\mathcal{T}_i}_{U}\end{bmatrix} \tag{25b}$$

As a result of the single-valued trace quantities $\widehat{\mathbf{u}}_h$, the final matrix system of the HDG method is smaller than that of many other DG methods [10, 38, 39]. Moreover, the matrix $\widehat{\mathbf{A}}$ has a small bandwidth since solely the degrees of freedom between neighboring faces that share the same macro-element are connected [10].

Remark 2. We note that the local vector updates $\Delta Q^{\mathcal{T}_i}$ and $\Delta U^{\mathcal{T}_i}$, and the global reduced system (24) need to be stored for each macro-element.

3.3 | Linear Solver: Second-Layer Static Condensation

We discuss an alternative to the static condensation strategy described in Section 3.2, which we refer to as second-layer static condensation in the following. We note that our algorithm is motivated by the work of Kronbichler and co-authors [40] who proposed a very similar algorithm (see Algorithm 3 and the equations below on page 721 of their article). This approach considers an alternative implementation of the trace matrix–vector product. Instead of explicitly forming the Schur complement, the matrix system (25) is expanded in terms of all contributing matrices [17]. This method allows most matrix–vector products to be executed in a matrix-free manner, employing the scheme proposed in Kronbichler and co-authors [40] for the fast computation of HDG residuals.

We start by exploiting the structure of the local block matrices, which read:

$$\mathbf{A}^{\mathcal{T}_i}_{\text{local}} = \begin{bmatrix}\mathbf{A}^{\mathcal{T}_i}_{qq} & \mathbf{A}^{\mathcal{T}_i}_{qu}\\ \mathbf{A}^{\mathcal{T}_i}_{uq} & \mathbf{A}^{\mathcal{T}_i}_{uu}\end{bmatrix} = \left[\begin{array}{ccc|c}\mathbf{A}^{\mathcal{T}_i}_{q_xq_x} & 0 & 0 & \mathbf{A}^{\mathcal{T}_i}_{q_xu}\\ 0 & \mathbf{A}^{\mathcal{T}_i}_{q_yq_y} & 0 & \mathbf{A}^{\mathcal{T}_i}_{q_yu}\\ 0 & 0 & \mathbf{A}^{\mathcal{T}_i}_{q_zq_z} & \mathbf{A}^{\mathcal{T}_i}_{q_yu}\\ \hline \mathbf{A}^{\mathcal{T}_i}_{uq_x} & \mathbf{A}^{\mathcal{T}_i}_{uq_y} & \mathbf{A}^{\mathcal{T}_i}_{uq_z} & \mathbf{A}^{\mathcal{T}_i}_{uu}\end{array}\right] \tag{26}$$

The form of (26) permits an efficient storage and inversion strategy. Namely the inverse of the local matrix is given by:

$$\left(\mathbf{A}^{\mathcal{T}_i}_{\text{local}}\right)^{-1} = \begin{bmatrix}\left(\mathbf{A}^{\mathcal{T}_i}_{qq}\right)^{-1} + \left(\mathbf{A}^{\mathcal{T}_i}_{qq}\right)^{-1}\mathbf{A}^{\mathcal{T}_i}_{qu}\left(\mathbf{S}^{\mathcal{T}_i}\right)^{-1}\mathbf{A}^{\mathcal{T}_i}_{uq}\left(\mathbf{A}^{\mathcal{T}_i}_{qq}\right)^{-1} & -\left(\mathbf{A}^{\mathcal{T}_i}_{qq}\right)^{-1}\mathbf{A}^{\mathcal{T}_i}_{qu}\left(\mathbf{S}^{\mathcal{T}_i}\right)^{-1}\\ -\left(\mathbf{S}^{\mathcal{T}_i}\right)^{-1}\mathbf{A}^{\mathcal{T}_i}_{uq}\left(\mathbf{A}^{\mathcal{T}_i}_{qq}\right)^{-1} & \left(\mathbf{S}^{\mathcal{T}_i}\right)^{-1}\end{bmatrix} \tag{27}$$

where $\mathbf{S}^{\mathcal{T}_i} = \mathbf{A}^{\mathcal{T}_i}_{uu} - \mathbf{A}^{\mathcal{T}_i}_{uq}\left(\mathbf{A}^{\mathcal{T}_i}_{qq}\right)^{-1}\mathbf{A}^{\mathcal{T}_i}_{qu}$ is the Schur complement of $\mathbf{A}^{\mathcal{T}_i}_{\text{local}}$. It is needless store the complete dense inverse $\left(\mathbf{A}^{\mathcal{T}_i}_{\text{local}}\right)^{-1}$. We observe that the inverse contains the inverse of the Schur complement $\mathbf{S}^{\mathcal{T}_i}$ as well as the inverse of $\mathbf{A}^{\mathcal{T}_i}_{qq}$. We compute and store the Schur complement $\left(\mathbf{S}^{\mathcal{T}_i}\right)^{-1}$ on each macro-element, which is the same size as $\mathbf{A}^{\mathcal{T}_i}_{uu}$. The block matrix $\mathbf{A}^{\mathcal{T}_i}_{qq}$ consists of the components $\mathbf{A}^{\mathcal{T}_i}_{q_kq_k} = \text{diag}\left(J^{\mathcal{T}_i}\right)\mathbf{M}$, where k is the coordinate direction, $\mathbf{M}$ is the mass matrix over a reference macro-element, and $J^{\mathcal{T}_i}$ is the determinant of the Jacobian of the geometrical map. Its inverse is given by:

$$\left(\mathbf{A}^{\mathcal{T}_i}_{q_kq_k}\right)^{-1} = \text{diag}\left(1/J^{\mathcal{T}_i}\right)\mathbf{M}^{-1}$$

where $\mathbf{M}^{-1}$ is the inverse mass matrix on a reference macro-element. The inverse mass matrix may be precomputed and stored. Substituting $\mathbf{A}^{\mathcal{T}_i}_{\text{local}}$ from of (27) into Equation (25), the contributions to the global system take the form:

$$\begin{aligned}\widehat{\mathbf{A}}^{\mathcal{T}_i} = &\left(\mathbf{A}^{\mathcal{T}_i}_{\hat{u}u} - \mathbf{A}^{\mathcal{T}_i}_{\hat{u}q}\left(\mathbf{A}^{\mathcal{T}_i}_{qq}\right)^{-1}\mathbf{A}^{\mathcal{T}_i}_{qu}\right)\left(\mathbf{S}^{\mathcal{T}_i}\right)^{-1}\left(-\mathbf{A}^{\mathcal{T}_i}_{u\hat{u}} + \mathbf{A}^{\mathcal{T}_i}_{uq}\left(\mathbf{A}^{\mathcal{T}_i}_{qq}\right)^{-1}\mathbf{A}^{\mathcal{T}_i}_{q\hat{u}}\right)\\ &+\left(\mathbf{A}^{\mathcal{T}_i}_{\hat{u}\hat{u}} - \mathbf{A}^{\mathcal{T}_i}_{\hat{u}q}\left(\mathbf{A}^{\mathcal{T}_i}_{qq}\right)^{-1}\mathbf{A}^{\mathcal{T}_i}_{q\hat{u}}\right)\end{aligned} \tag{28a}$$

$$\begin{aligned}\widehat{\mathbf{b}}^{\mathcal{T}_i} = &\left(-\mathbf{A}^{\mathcal{T}_i}_{\hat{u}u} + \mathbf{A}^{\mathcal{T}_i}_{\hat{u}q}\left(\mathbf{A}^{\mathcal{T}_i}_{qq}\right)^{-1}\mathbf{A}^{\mathcal{T}_i}_{qu}\right)\left(\mathbf{S}^{\mathcal{T}_i}\right)^{-1}\left(-R^{\mathcal{T}_i}_{U} + \mathbf{A}^{\mathcal{T}_i}_{uq}\left(\mathbf{A}^{\mathcal{T}_i}_{qq}\right)^{-1}R^{\mathcal{T}_i}_{Q}\right)\\ &+\left(-R^{\mathcal{T}_i}_{\hat{U}} + \mathbf{A}^{\mathcal{T}_i}_{\hat{u}q}\left(\mathbf{A}^{\mathcal{T}_i}_{qq}\right)^{-1}R^{\mathcal{T}_i}_{Q}\right)\end{aligned} \tag{28b}$$

Finally, the local solution updates $\Delta Q^{\mathcal{T}_i}$ and $\Delta U^{\mathcal{T}_i}$ are given by:

$$\begin{aligned}\Delta U^{\mathcal{T}_i} = &\left(\mathbf{S}^{\mathcal{T}_i}\right)^{-1}\left[\left(\mathbf{A}^{\mathcal{T}_i}_{uq}\left(\mathbf{A}^{\mathcal{T}_i}_{qq}\right)^{-1}R^{\mathcal{T}_i}_{Q} - R^{\mathcal{T}_i}_{U}\right)\right.\\ &\left.+\left(-\mathbf{A}^{\mathcal{T}_i}_{u\hat{u}}\Delta\widehat{U}^{\Gamma_i} + \mathbf{A}^{\mathcal{T}_i}_{uq}\left(\mathbf{A}^{\mathcal{T}_i}_{qq}\right)^{-1}\mathbf{A}^{\mathcal{T}_i}_{q\hat{u}}\Delta\widehat{U}^{\Gamma_i}\right)\right]\end{aligned} \tag{29a}$$

$$\Delta Q^{\mathcal{T}_i} = \left(\mathbf{A}^{\mathcal{T}_i}_{qq}\right)^{-1}\left(-R^{\mathcal{T}_i}_{Q} - \mathbf{A}^{\mathcal{T}_i}_{qu}\Delta U^{\mathcal{T}_i} - \mathbf{A}^{\mathcal{T}_i}_{q\hat{u}}\Delta\widehat{U}^{\Gamma_i}\right) \tag{29b}$$

Remark 3. We do not explicitly compute the inverse of the Schur complement $\mathbf{S}^{-1}$, but instead compute an appropriate factorization that we store, and then apply the inverse to a vector $\mathbf{a}$ by solving the system $\mathbf{Sx} = \mathbf{a}$ via back-substitution.

Remark 4. We note that the optimizations between Equations (27) and (28) assume an affine mapping, which excludes mappings on curved elements required for the description of curved geometries. In this work, all elements in the interior of the domains, and hence the majority of the elements, are affine. All interior elements are therefore straight-sided, such that we can employ the quadrature-free approach. Curved elements appear only along curved domain boundaries. The curved elements at the boundaries are treated separately and with quadrature.

4 | Implementation Aspects

The concept of the matrix-free evaluation of high-order DG operators has been widely adopted and implemented within several software and research projects, including deal.II [41],

mfem [42], ExaDG [43], Exasim [44], and Nektar++ [45]. These fast evaluation techniques are directly applicable to graphics processors [46], and have been successfully applied to wave propagation [47, 48] and fluid flow problems [49, 50]. As part of the matrix-free evaluation of the macro-element HDG method, we have developed efficient implementations for the compressible Navier–Stokes equations. In the following, we provide details of a matrix-free implementation and a brief description of the preconditioning approach, which we will use in the computational study thereafter.

4.1 | Matrix-Free Implementation

We apply a straightforward matrix-free parallel implementation of the macro-element HDG method, for the both linear solvers presented in Sections 3.2 and 3.3. We provide the details for the second-layer static condensation of Section 3.3 in Algorithm 1, and note that the static condensation of Section 3.3 follows similarly. We provide a few core details of the algorithm for the sake of clarity. In Lines 2–5, the vector contributions of each macro-element local to process n are assembled in a global vector. Due to the discontinuous nature of macro-elements, this procedure requires only data from the macro-element local to each process, $\mathcal{T}_h^n$, and hence implies no communication between processes.

The global linear system solve in Line 6 is carried out via a matrix-free iterative procedure such as GMRES, relying on efficient matrix–vector products. In addition, matrix-based iterative solvers frequently face significant memory bandwidth constraints on modern processors in the high-order finite element context [51]. Methods that utilize less memory can be more efficient for matrix–vector products, even if they involve more arithmetic operations. Moreover, a relatively small memory sizes might enable the matrix inside the iterative solver to fit into caches, which might then help performance, because operations with the system matrix are usually limited by the memory bandwidth, not the arithmetic compute performance.

Finally, $\Delta Q^{\mathcal{T}i}$ and $\Delta U^{\mathcal{T}i}$ are obtained from $\Delta\widehat{U}^{\Gamma_i}$ in Lines 7–10, where $\Delta U^{\mathcal{T}i}$ is derived from Equation (29a), and $\Delta Q^{\mathcal{T}i}$ is derived from Equation (29b). We note that the procedures local to each macro-element can, but do not have to be implemented in a matrix-free fashion, as the corresponding matrices are comparatively small.

Algorithm 2 outlines an efficient matrix-free matrix–vector product, Equation (28a). The vector $\Delta\widehat{U}$ contains all degrees of freedom on the macro-element interfaces ε. The degrees of freedom on each interior interface $e \in \varepsilon$ will be operated on by $\widehat{\mathbf{A}}^{\mathcal{T}_+}$ and $\widehat{\mathbf{A}}^{\mathcal{T}_-}$. We adopt the notation $\mathcal{T}_\pm$ to denote the macro-elements on the left and right side of an interface. Lines 6–8 constitute the iterations of the matrix-free algorithm that are performed for each macro-element, $\mathcal{T}_i$, in parallel, in the following four steps:

1. $y_1 = \left(-\mathbf{A}^{\mathcal{T}_i}_{u\hat{u}}\Delta\widehat{U}^{\Gamma_i} + \mathbf{A}^{\mathcal{T}_i}_{uq}\left(\mathbf{A}^{\mathcal{T}_i}_{qq}\right)^{-1}\mathbf{A}^{\mathcal{T}_i}_{q\hat{u}}\Delta\widehat{U}^{\Gamma_i}\right)$
2. $y_2 = \left(\mathbf{S}^{\mathcal{T}_i}\right)^{-1} y_1$
3. $y_3 = \left(\mathbf{A}^{\mathcal{T}_i}_{\hat{u}u} - \mathbf{A}^{\mathcal{T}_i}_{\hat{u}q}\left(\mathbf{A}^{\mathcal{T}_i}_{qq}\right)^{-1}\mathbf{A}^{\mathcal{T}_i}_{qu}\right) y_2$
4. $y_4 = y_3 + \left(\mathbf{A}^{\mathcal{T}_i}_{\hat{u}\hat{u}}\Delta\widehat{U}^{\Gamma_i} - \mathbf{A}^{\mathcal{T}_i}_{\hat{u}q}\left(\mathbf{A}^{\mathcal{T}_i}_{qq}\right)^{-1}\mathbf{A}^{\mathcal{T}_i}_{q\hat{u}}\Delta\widehat{U}^{\Gamma_i}\right)$

Steps 1, 3, and 4 act on the global system, whereas Step 2 operates on the local system. Step 4 involves a reduction that relies on the data associated with the mesh skeleton faces and data from macro-elements, and thus requires communication among processors.

Algorithms 1 and 2 together constitute the complete algorithmic procedure for performing a macro-element HDG solve, storing only the reference-to-physical macro-element transformation data, the inverse Schur complements $\left(\mathbf{S}^{\mathcal{T}_i}\right)^{-1}$, and the right-hand-side vector $\widehat{\mathbf{b}}$.

We maintain the matrix-free nature of our approach by avoiding the storage of the local matrix $\mathbf{S}^{\mathcal{T}_i}$, even though local matrices are stored patch-wise on macro-elements rather than on individual elements [52]. This is a critical distinction from the "HDG compact matrix-free" approach [17], which uses an alternative

ALGORITHM 1 | Solution procedure on each macro-element associated with one parallel process.

1: $n \leftarrow$ Current process
2: **for** $\mathcal{T}_i \in \mathcal{T}_h^n$ **do**
3: $\quad\widehat{\mathbf{b}}^{\mathcal{T}_i} \leftarrow$ Vector contribution from $Q^{\mathcal{T}_i}, U^{\mathcal{T}_i}, \widehat{U}^{\Gamma_i}$
4: $\quad\widehat{\mathbf{b}} \leftarrow$ Assemble $\widehat{\mathbf{b}}^{\mathcal{T}_i}$ for all $\mathcal{T}_i \in \mathcal{T}_h^n$
5: **end for**
6: $\Delta\widehat{U} \leftarrow$ Matrix-free iterative solve $\widehat{\mathbf{A}}\ \Delta\widehat{U} = \widehat{\mathbf{b}}$
7: **for** $\mathcal{T}_i \in \mathcal{T}_h^n$ **do**
8: $\quad\Delta Q^{\mathcal{T}_i} \leftarrow \left(\mathbf{A}^{\mathcal{T}_i}_{qq}\right)^{-1}\left(-R^{\mathcal{T}_i}_Q - \mathbf{A}^{\mathcal{T}_i}_{qu}\Delta U^{\mathcal{T}_i} - \mathbf{A}^{\mathcal{T}_i}_{q\hat{u}}\Delta\widehat{U}^{\Gamma_i}\right).$
9: $\quad\Delta U^{\mathcal{T}_i} \leftarrow \left(\mathbf{S}^{\mathcal{T}_i}\right)^{-1}\left[\left(\mathbf{A}^{\mathcal{T}_i}_{uq}\left(\mathbf{A}^{\mathcal{T}_i}_{qq}\right)^{-1}R^{\mathcal{T}_i}_Q - R^{\mathcal{T}_i}_U\right) + \left(-\mathbf{A}^{\mathcal{T}_i}_{u\hat{u}}\Delta\widehat{U}^{\Gamma_i} + \mathbf{A}^{\mathcal{T}_i}_{uq}\left(\mathbf{A}^{\mathcal{T}_i}_{qq}\right)^{-1}\mathbf{A}^{\mathcal{T}_i}_{q\hat{u}}\Delta\widehat{U}^{\Gamma_i}\right)\right].$
10: **end for**

ALGORITHM 2 | Distributed matrix-free procedure for the matrix-vector product $\widehat{\mathbf{A}}\ \Delta\widehat{U}$.

1: $\mathbf{y} \leftarrow 0$
2: $n \leftarrow$ Current processor
3: **for** $\mathcal{T}_i \in \mathcal{T}_h^n$ **do**
4: $\quad\varepsilon \leftarrow$ Extract global DOF indices on $\partial\mathcal{T}_i$
5: $\quad\Delta\widehat{U}^{\Gamma_i} \leftarrow \Delta\widehat{U}[\varepsilon]$
6: $\quad\mathbf{y}[\varepsilon] \leftarrow \mathbf{y}[\varepsilon] + \left(\mathbf{A}^{\mathcal{T}_i}_{\hat{u}u} - \mathbf{A}^{\mathcal{T}_i}_{\hat{u}q}\left(\mathbf{A}^{\mathcal{T}_i}_{qq}\right)^{-1}\mathbf{A}^{\mathcal{T}_i}_{qu}\right)\left(\mathbf{S}^{\mathcal{T}_i}\right)^{-1}$
7: $\qquad\left(-\mathbf{A}^{\mathcal{T}_i}_{u\hat{u}}\Delta\widehat{U}^{\Gamma_i} + \mathbf{A}^{\mathcal{T}_i}_{uq}\left(\mathbf{A}^{\mathcal{T}_i}_{qq}\right)^{-1}\mathbf{A}^{\mathcal{T}_i}_{q\hat{u}}\Delta\widehat{U}^{\Gamma_i}\right)$
8: $\quad+\left(\mathbf{A}^{\mathcal{T}_i}_{\hat{u}\hat{u}}\Delta\widehat{U}^{\Gamma_i} - \mathbf{A}^{\mathcal{T}_i}_{\hat{u}q}\left(\mathbf{A}^{\mathcal{T}_i}_{qq}\right)^{-1}\mathbf{A}^{\mathcal{T}_i}_{q\hat{u}}\Delta\widehat{U}^{\Gamma_i}\right)$
9: **end for**
10: $\Delta\widehat{U} \leftarrow \mathbf{y}$

condensation technique on elements to enhance efficiency. As shown in analyses from References [25, 52] and results from matrix-free variants of CG/HDG methods [17], storing the global matrix is far less efficient than storing local matrices element-wise. This is why our method avoids using a global matrix. While our approach shares similarities with methods in frameworks such as Nektar++, deal.II, MFEM, and CEED, our key innovation lies in how we handle local matrices. Specifically, we avoid explicitly computing these matrices on elements. We refer readers interested in further details on the matrix-free implementation for the macro-element HDG method to our earlier paper [25] which specifically focuses on this aspect.

4.2 | Preconditioning Methods

In this work, we consider two options for solving the global linear system (24). The first and straightforward one solves the linear system in parallel using the restarted GMRES method [53] with iterative classical Gram-Schmidt (ICGS) orthogonalization. In order to accelerate convergence, a left preconditioner is used by the inverse of the block matrix $\mathbf{A}_{\widehat{u}\widehat{u}}$. We emphasize that the matrix $\left(\mathbf{A}_{\widehat{u}\widehat{u}}\right)^{-1}$ is never actually computed. Since all blocks $\mathbf{A}_{\widehat{u}\widehat{u}}^{\mathcal{T}_i}$ are positive definite and symmetric, we compute and store its Cholesky factorization in-place.

In the second option, we use an iterative solver strategy, where we use a flexible implementation of the GMRES (FGMRES) method for our spatial discretization scheme. One notable difference between FGMRES and the standard GMRES algorithm is that FGMRES can improve memory efficiency by fully utilizing vectors that are not actively used in a given iteration. These unused vectors can, for instance, be leveraged to compute a preconditioned vector through an additional GMRES run, thereby optimizing resource usage [54]. We note that within matrix-free approaches such as the one followed here, multilevel algorithms have emerged as a promising choice to precondition a flexible GMRES solver, see for example the reference [54]. This methodology has demonstrated success when applied for compressible flow problems [55–58] and incompressible flow problems [59–61]. In this regard, we use the FGMRES iteration, where GMRES itself is employed as a preconditioner, and a GMRES approach, where the inverse of the global matrix $\left(\mathbf{A}_{\widehat{u}\widehat{u}}\right)^{-1}$ is employed as a preconditioner. We note that we determine this inverse in the same way as above via a matrix-free Cholesky factorization. We note that preconditioned iterative solvers become more effective as the time step size of the discretization increases, which implies that the condition number increases.

For a macro-element HDG mesh consisting of N^{Mcr} macro-elements, the preconditioner is applied to the matrix $\mathbf{A}_{\widehat{u}\widehat{u}}^{\mathcal{T}_i}$ corresponding to each macro-element, which contains N^{Mcr} local matrices. It is important to note that while the global solver considers the entire domain, the preconditioning method is applied only to the macro-element entities. For further details, we refer to Section 3.3.2 of our previous paper [25].

4.3 | Computational Setup

We implement our methods in Julia,[1] by means of element formation routines. In the context of the local solver, the associated tasks involve the assembly of local matrices, the computation of local LU factorizations of $\mathbf{S}^{\mathcal{T}_i}$, and the extraction of local values from the global solution. We use FGMRES/GMRES iterative methods for the global solver provided by the open-source library PETSc.[2] Motivated by the very small size of the matrices on each element, we employ the dense linear algebra from the LAPACK package[3] for the standard HDG method. The matrices for the macro-element HDG method are larger and therefore we utilize sparse linear algebra provided by the UMFPACK library [62].

Our implementation has been adapted to the parallel computing environment Lichtenberg II (Phase 1), provided by the High-Performance Computing Center at the Technical University of Darmstadt. It was compiled using GCC (version 9.2.0), Portable Hardware Locality (version 2.7.1), and OpenMPI (version 4.1.2). Our computational experiments were conducted on this cluster system, utilizing multiple compute nodes. Each compute node features two Intel Xeon Platinum 9242 processors, each equipped with 48 cores running at a base clock frequency of 2.3 GHz. Additionally, each compute node provides a main memory capacity of up to 384 GB. For more detailed information about the available compute system, please refer to the Lichtenberg webpage.[4]

5 | Numerical Results

In the following, we will demonstrate the computational advantages of our macro-element HDG approach for compressible flow problems, in particular in comparison with the standard HDG method. Our numerical experiments encompass several three-dimensional test cases that include steady and unsteady flow scenarios. Here we consider the DIRK (3,3) scheme for the discretization in time. We conduct a comprehensive comparative analysis in terms of accuracy, number of iterations (all time steps), computing times (wall time for all time steps), and required number of degrees of freedom in the local/global solver, with a particular focus on the parallel implementation.

Remark 5. We will use the parameter m to denote the number of C^0-continuous elements along each macro-element edge. Therefore, the parameter m defines the size of the mesh of C^0-continuous elements in each tetrahedral macro-element. For $m = 1$, we obtain the standard HDG method.

5.1 | Compressible Couette Flow

To demonstrate our method for a simple example and to illustrate its optimal convergence, we consider the problem of steady-state compressible Couette flow with a source term [9, 63] on a three-dimensional cubic domain $\Omega = (x_1, x_2, x_3) = (0, 1)^3$. The analytical expression of the axis-aligned solution is:

$$v_1 = x_2 \log(1 + x_2) \tag{30a}$$

$$v_2 = 0 \tag{30b}$$

$$v_3 = 0 \tag{30c}$$

$$T = T_\infty \left(\alpha_c + x_2(\beta_c - \alpha_c) + \frac{\gamma - 1}{2\gamma\rho_\infty} Pr\, x_2(1 - x_2) \right) \tag{30d}$$

where we assume the following parameters: $\alpha_c = 0.8$, $\beta_c = 0.85$, $\gamma = 1.4$, $T_\infty = 1/(1-\gamma)M_\infty^2$, and $Pr = 0.71$. In addition, the Mach number is taken as $M_\infty = v_\infty/c_\infty = 0.15$, where c_∞ is the speed of sound corresponding to the temperature T_∞ and v_∞ is infinity velocity. The viscosity is assumed constant and the source term, $\mathbb{S}$, which is determined from the exact solution, is given by

$$\mathbb{S} = \frac{-1}{Re}\left\{0, \frac{2+x_2}{(1+x_2)^2}, 0, 0, \log^2(1+x_2) + \frac{x_2\log(1+x_2)}{1+x_2} + \frac{y(3+2x_2)\log(1+x_2) - 2x_2 - 1}{(1+x_2)^2}\right\}^t \tag{31}$$

While the exact solution is independent of the Reynolds number $Re = \rho_\infty v_\infty L/\mu_\infty = Re_{c_\infty} M_\infty$, we set it to 1 in order to replicate the case presented in References [9, 63], taking a characteristic length of $L = 1$.

Figure 2b plots the exact solution 30 in the $x_1 - x_2$ that corresponds to the source term 31. We consider polynomial orders $p = 1$ to $p = 5$ and consider meshes as shown in Figure 2a, where we choose the number m of elements along a macro-element edge in each direction to be two. In Figure 3a,b, we plot the error in the L^2 norm versus the element size under uniform refinement of the macro-elements for velocity and energy, respectively. We observe that we achieve the optimal convergence rate $p + 1$ in all cases.

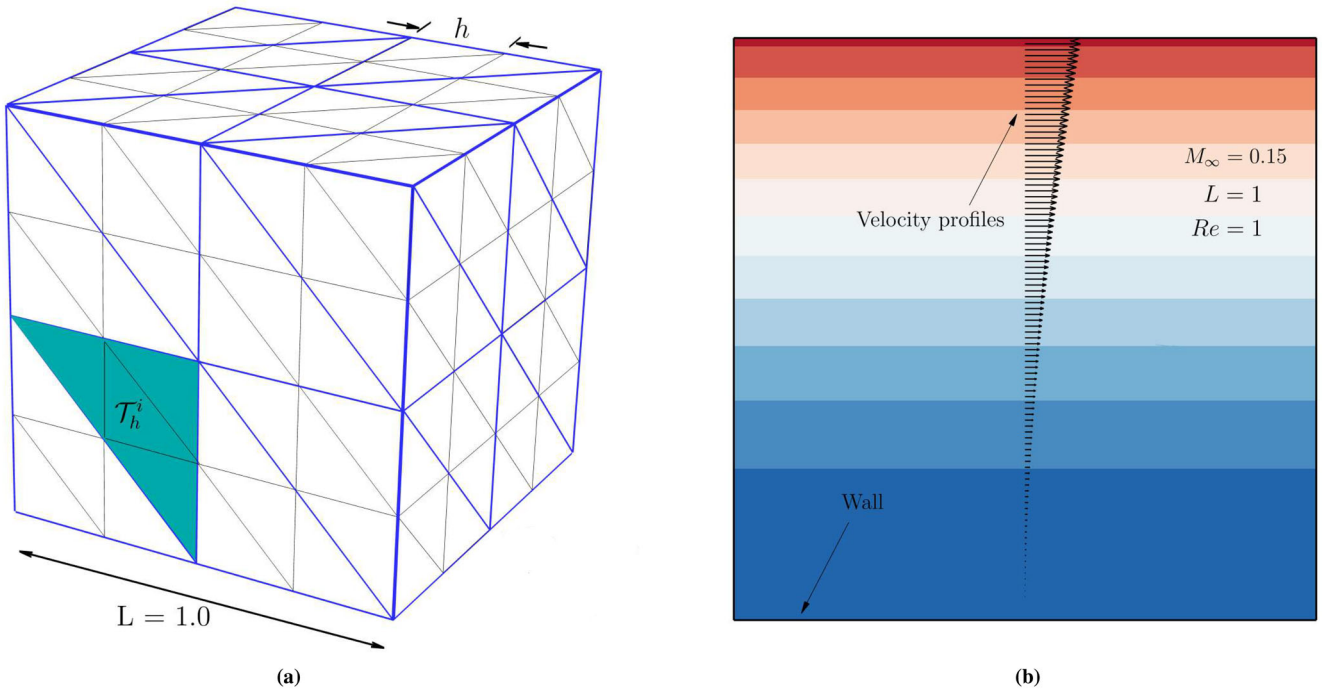


FIGURE 2 | Compressible Couette flow on a unit cube. (a) 3D structured mesh. (b) Exact solution in $x_1 - x_2$ plane. [Colour figure can be viewed at wileyonlinelibrary.com]

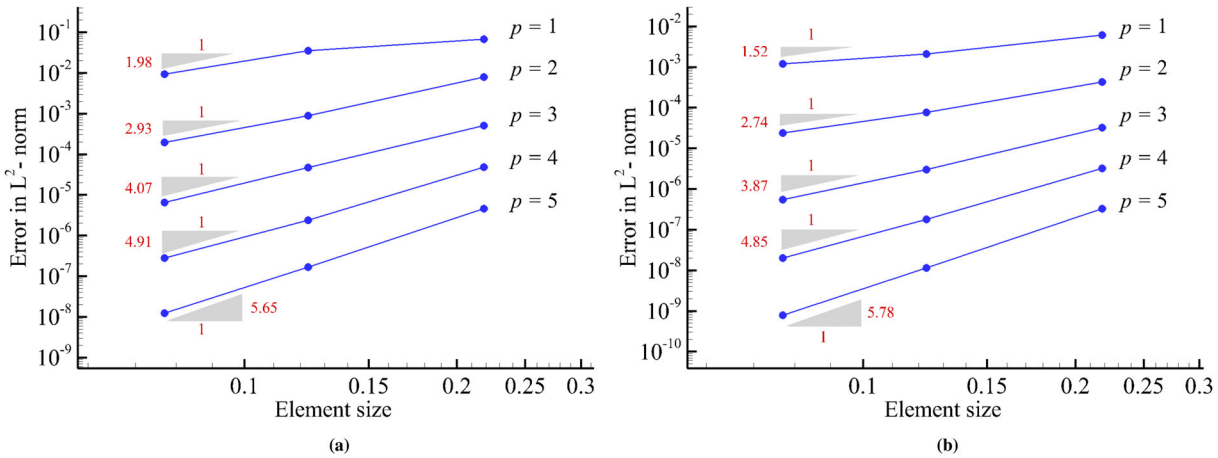


FIGURE 3 | Couette flow: Convergence of the macro-element HDG method with $m = 2$. (a) Velocity, $\|v_{1h} - v_1\|_{L2(\Omega)}$. (b) Energy, $\|\rho E_h - \rho E\|_{L2(\Omega)}$. [Colour figure can be viewed at wileyonlinelibrary.com]

5.1.1 | Computational Performance for a Fixed Global Mesh Size

To assess the performance of the local and global solvers with respect to a change in the number of elements m per macro-element, we consider a discretization of the cube with 768 elements with polynomial degree $p = 3$ that we apply to the current problem. We compute in parallel on 12 processors. We compare the standard HDG method ($m = 1$) versus the macro-element HDG method with eight elements per macro-element ($m = 2$) and 64 elements per macro-element ($m = 4$). Each HDG variant uses the same overall mesh of $N^{Elm} = 768$ elements to discretize the cube, only changing the number of macro-elements, N^{Mcr}. In Table 1, we report the number of degrees of freedom for the global problem for $m = 1$ (standard HDG method), $m = 2$ and $m = 4$ (macro-element HDG method). In Table 2, we report the time and number of iterations of the FGMRES method, where we use a drop tolerance of 10^{-12}. In addition, we compare for each HDG variant a GMRES linear solver approach versus a FGMRES linear solver approach, where both variants use the $\left(\mathbf{A}_{\hat{u}\hat{u}}\right)^{-1}$ preconditioner for the GMRES solver as described above.

As for the nonzero elements in the matrices, we note that the number of nonzeros in the local matrices increases with larger values of m. For further details, we refer to Section 5.1 of our previous paper [25]. From an operational standpoint, this suggests that the macro-element HDG method might be less efficient than the standard HDG method. However, this analysis does not account for several further factors that could be crucial. The standard HDG method requires more inter-processor communication due to a significantly higher number of local problems. Furthermore, the number of iterations needed to solve the global problem is generally much lower for the macro-element variant. Consequently, in Table 2, we focus solely on local operations, global operations, and the number of iterations to determine the optimal m.

We see that in both variants of the linear solver considered here, the computing time required for the local solver increases when m is increased. This increase is expected as the number of degrees of freedom in the local solver increases. We observe that the FGMRES approach is more efficient than the GMRES approach for all HDG variants. The decrease in the local and global computing time is even larger for the macro-element HDG methods than for the standard HDG method. We can also see that compared to the standard HDG method ($m = 1$), the global solver operations can be significantly reduced for the macro-element HDG variants with increasing m. We note that this is the opposite when the GMRES linear solver is employed. Based on this observation, we will focus on the FGMRES solver in the remainder of this study.

5.1.2 | Computational Performance for a Fixed Number of Local Unknowns Per Macro-Element

In the next step, we test the HDG variants for a fixed number of local unknowns that we choose as $\text{dof}_i^{local} = 3{,}300$. We focus on investigating the performance of one- and two-level static condensation in conjunction with FGMRES iterations. To this end, instead of keeping the number of processors fixed, we switch to a variable number of processors, but keep the ratio of the number of macro-elements to the number of processors fixed at one (macro-elements/proc.'s = 1). The macro-element HDG method combines patches of eight ($m = 2$), 64 ($m = 4$) or 512 ($m = 8$) C^0-continuous tetrahedral elements into one macro-element. To always obtain the same number of local unknowns dof_i^{local}, we adjust the polynomial degree p in such a way that the product of m and p is always $mp = 8$. Consequently, with an increase in the macro-element size m, the polynomial order p decreases. Table 3 reports the number of degrees of freedom for two of the meshes

TABLE 3 | Couette flow: Number of local and global unknowns at a constant number of local unknowns per macro-element.

	dof^{local}	dof^{global}	dof_i^{local}
$N^{Mcr} = 12$	39,600	6750	3300
$N^{Mcr} = 44$	145,200	25,200	3300

Note: We report values for two different meshes with 12 and 44 macro-elements, which hold for the pairs $(m, p) = (2, 4)$, $(m, p) = (4, 2)$, and $(m, p) = (8, 1)$.

TABLE 1 | Couette flow: Number of local and global unknowns for $p = 3$ and varying m on a fixed global mesh.

	dof^{local} (total)			dof^{global}			dof_i^{local} (per macro-element)		
	$m = 1$	$m = 2$	$m = 4$	$m = 1$	$m = 2$	$m = 4$	$m = 1$	$m = 2$	$m = 4$
$N^{Elm} = 768$	307,200	161,280	109,200	81,600	30,240	13,650	400	1680	9100

TABLE 2 | Couette flow: We compare the time for the local solver and the local part of the matrix-free global solver (Step 2) versus the time for the remaining parts of the global solver (residual drop 10^{-12}).

	Time local op's [s]			Time global op's [s]			# Newton iterations		
	$m = 1$	$m = 2$	$m = 4$	$m = 1$	$m = 2$	$m = 4$	$m = 1$	$m = 2$	$m = 4$
GMRES	11.1	15.4	120.1	27.3	15.1	58.9	199	160	160
FGMRES	8.6	5.2	26.6	20.6	4.5	10.1	56	50	49

Note: We use standard HDG ($m = 1$) versus macro-element HDG ($m = 2, 4$) (all with $p = 3$, GMRES preconditioner $\left(\mathbf{A}_{\hat{u}\hat{u}}\right)^{-1}$, proc.'s = 12).

with 12 and 44 macro-elements that we generated by uniformly refining the initial mesh given in Figure 7.

Table 4 reports the computing time for the local operations and the time for the global operations with increasing number of elements per macro-element, parametrized by m. As we use the same preconditioner, the number of Newton iterations in the static condensation and alternative second-layer static condensation approaches are practically the same. We note that the large number of iterations is not uncommon in the present case, as we solve a steady-state problem with a matrix-free method and a preconditioner that can only use the locally available part of the matrix [60, 64]. We observe that as m increases, the number of iterations decreases, despite the constant number of degrees of freedom in the local/global solver. This phenomenon can be attributed to the change in structure of the global matrices, leading to improved conditioning. Due to the reduced number of iterations, the time required for both local and global operations is reduced. When looking at the results of Table 4, it is important to keep in mind that a change in m also involves a change in p. The variation in polynomial order, however, strongly influences the accuracy of the approximate solution. Table 5 presents the approximation error corresponding to different values of p for the number of degrees of freedom reported in Table 3. We observe that with increasing the polynomial degree, we obtain a significantly better accuracy.

5.2 | Laminar Compressible Flow Past a Sphere

In the next step, we consider the test case of laminar compressible flow past a sphere at Mach number $M_\infty = 0.1$ and Reynolds number $Re = 100$. At these conditions, we expect a steady state solution with a large toroidal vortex formed just aft of the sphere. As we would like to arrive at the steady-state solution in a computational efficient manner, we would like to choose an implicit time integration scheme with a large time step, resulting in a large CFL number (in our case in the order of 10^5). The computational mesh, shown in Figure 4, consists of 139,272 tetrahedral elements at moderate polynomial order $p = 3$. We compare the standard HDG method, where all 139,272 elements are discontinuous, and the macro-element HDG method with $m = 2$ that uses the same elements grouped into 17,409 discontinuous macro-elements.

Figure 5 illustrates the solution characteristics of the problem achieved with the macro-element HDG method in terms of the streamlines and the Mach number. We expect that as both methods use the same mesh at the same polynomial degree, they also achieve practically the same accuracy. This assumption is supported by Table 6, where we report the values obtained with our matrix-free implementation of the two HDG variants for the length of the ring vortex, x_s, and the angle θ_s at which separation occurs. We observe that both HDG variants produce values that agree well with each other as well as commonly accepted values reported in the literature [65, 66].

Table 7 reports the number of degrees of freedom for the mesh shown in Figure 5 and polynomial degree $p = 3$. Both HDG variants use the same mesh, where the standard HDG method assumes fully discontinuous elements ($m = 1$) and the macro-element HDG method combines groups of eight tetrahedral elements ($m = 2$) into one C^0 continuous macro-element. We observe that the macro-element HDG variant, with increasing m, significantly reduces the total amount of degrees of freedom in both the local and global problems.

We observe that the macro-element HDG method exhibits a notable speed advantage, primarily reducing the time for the global solver, while the time for the parallelized local solver

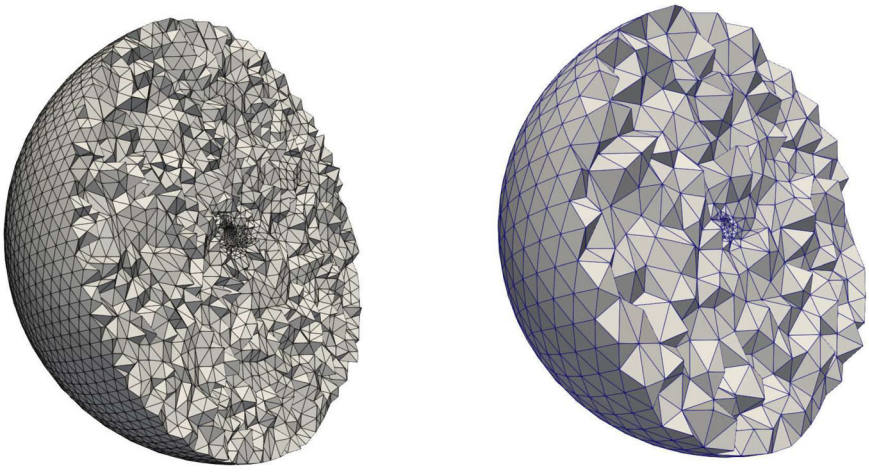
TABLE 4 | Couette flow: We compare the time for the local solver and the local part of the matrix-free global solver (Step 2) versus the time for the remaining parts of the global solver (residual drop 10^{-12}, number of macro-elements N^{Mcr}/proc.'s = 1).

	Time local op's [s]			Time global op's [s]			# iterations		
(m, p)	(2, 4)	(4, 2)	(8, 1)	(2, 4)	(4, 2)	(8, 1)	(2, 4)	(4, 2)	(8, 1)
Static condensation									
$N^{Mcr} = 12$	2.3	1.9	1.6	1.7	1.4	1.2	4481	3452	2984
$N^{Mcr} = 44$	4.4	3.4	3.3	4.3	3.2	3.4	10,314	8.137	7710
Second-layer static condensation									
$N^{Mcr} = 12$	0.6	0.5	0.4	1.3	0.9	0.7	4481	3452	2984
$N^{Mcr} = 44$	1.1	0.9	0.8	3.2	2.2	1.9	10,314	8.137	7710

TABLE 5 | Couette flow: Mass, velocity, and energy error in the L^2-norm for different (m, p) and number of macro-elements N^{Mcr}.

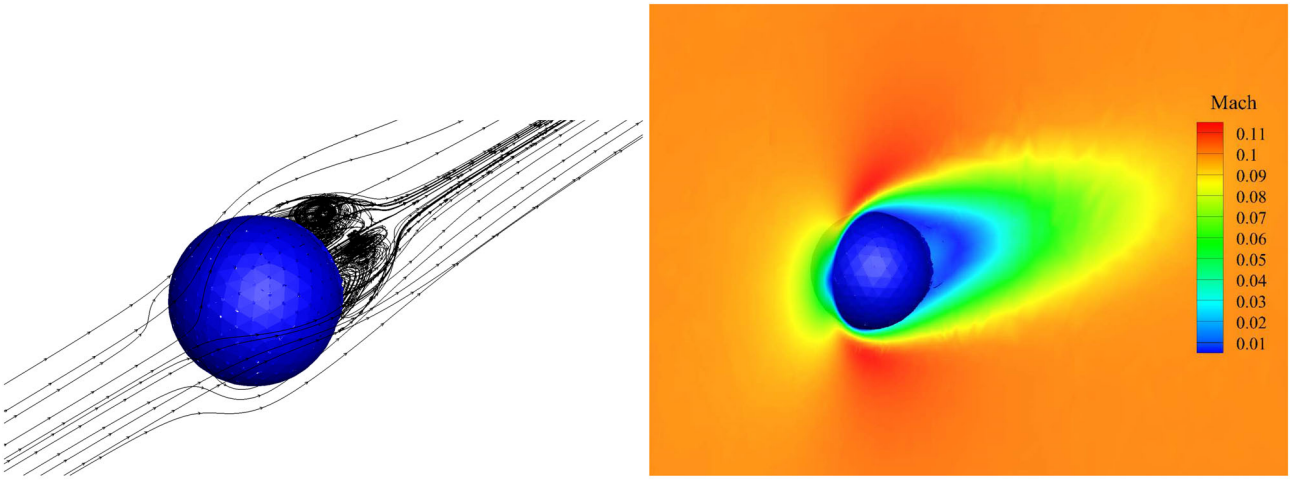
	$\| \rho_h - \rho \|_{L^2(\Omega)}$			$\| \boldsymbol{v}_{1h} - \boldsymbol{v}_1 \|_{L^2(\Omega)}$			$\| \rho E_h - \rho E \|_{L^2(\Omega)}$		
(m, p)	(2, 4)	(4, 2)	(8, 1)	(2, 4)	(4, 2)	(8, 1)	(2, 4)	(4, 2)	(8, 1)
$N^{Mcr} = 12$	1.72e-6	6.71e-5	6.41e-4	7.05e-6	2.21e-4	1.67e-3	1.13e-4	4.30e-3	4.15e-2
$N^{Mcr} = 44$	1.37e-7	1.52e-5	2.22e-4	8.90e-7	6.40e-5	9.15e-4	8.94e-6	9.43e-4	1.40e-2

Note: Subscript h denotes the numerical solution and no subscript denotes the exact solution.



(a) (b)

FIGURE 4 | Flow past a sphere: Cut halfway through the unstructured mesh, adaptively refined around the sphere in the center. (a) Element mesh. (b) Macro-element mesh. [Colour figure can be viewed at wileyonlinelibrary.com]



(a) (b)

FIGURE 5 | Flow past a sphere at $Re = 100$ and $M_\infty = 0.1$, computed on the mesh shown in Figure 4 at $p = 3$. (a) 3D streamlines, toroidal vortex aft of the sphere. (b) Mach number contours. [Colour figure can be viewed at wileyonlinelibrary.com]

TABLE 6 | Predicted length of the ring vortex and separation angle for flow past sphere at $M_\infty = 0.1$ and $Re = 100$.

Author(s)	Method	x_s	θ_s
Taneda [66]	Experimental	0.89	127.6
Johnson & Patel [65]	Finite volume method	0.88	126.6
Our matrix-free implementation	HDG	0.87	128.1
Our matrix-free implementation	Macro-element HDG	0.86	127.8

TABLE 7 | Flow past a sphere: We compare the time for the local solver and the local part of the matrix-free global solver (Step 2) versus the time for the remaining parts of the global solver for $p = 3$ and $Proc's = 2{,}048$ (on the mesh shown in Figure 4).

Time local op's [min]		Time global op's [min]		dof^{local}		dof^{global}	
$m = 1$	$m = 2$	$m = 1$	$m = 2$	$m = 1$	$m = 2$	$m = 1$	$m = 2$
5.38	6.34	83.6	26.6	55,708,800	29,247,120	14,123,600	5,012,000

remains at the same level. In the macro-element HDG method, computation time thus shifts from the global solver to the local solver, enhancing parallelization efficiency. Therefore, the time ratio between the local and global solver is thus reduced from approximately 16 in the standard HDG method to approximately four in the macro-element HDG method at $m = 2$. This example demonstrates that at moderate polynomial degrees such as $p = 3$, where the standard HDG method is typically not efficient, subdivision of one tetrahedral hybridized DG element into a macro-element with just eight C^0 continuous elements already leads to a practical advantage in terms of computational efficiency.

5.3 | Viscous Subsonic Flow Over NACA 0012 Airfoil

As the Mach numbers of the previous benchmark tests are relatively low for compressible flow, we now consider the test case of subsonic viscous flow around a NACA0012 airfoil. It clearly requires compressible flow equations, as the free-stream Mach and Reynolds numbers are $M_\infty = 0.5$ and $Re = 5{,}000$ with zero angle of attack. Zero heat flux (adiabatic) and no slip boundary conditions are imposed on the walls of the airfoil geometry. The flow is both steady and symmetric with respect to the stagnation stream line. However, flow separation occurs near the trailing edge, resulting in the formation of small re-circulation bubbles on the upper and lower surfaces of the airfoil that extend into the near-wake region and present numerical challenges for the accurate numerical prediction of the airfoil drag.

The computational mesh shown in Figure 6b consists of 15,360 triangle elements with polynomial order $p = 3$. We compare the standard HDG method, where all 15,360 elements are discontinuous, and the macro-element HDG method with $m = 2$ that uses the same elements grouped into 3840 discontinuous macro-elements shown in Figure 6a. Figure 6c depicts the Mach number contours that were computed with on the macro-element HDG mesh with $p = 3$ and $m = 2$. The re-circulation bubbles that appear as regions of low Mach number in the trailing-edge are plotted in Figure 6d. To evaluate accuracy, we compare our results with reference solutions for the pressure drag ($C_{D,p} = 0.0227$), the viscous drag ($C_{D,f} = 0.0327$), and the separation point location ($x_s/c = 0.81$), derived from a fourth-order accurate method utilizing a grid of 65,536 cells [67]. The relative errors of the results obtained from our macro-element HDG method with $p = 3$ and $m = 2$ ($C_{D,p} = 0.0225$, $C_{D,f} = 0.0331$, and $x_s/c = 0.814$) and the reference solution are 0.4%, 0.3%, and 0.09% for the pressure drag coefficient, the viscous drag coefficient, and the separation point, respectively. Since both the macro-element HDG and standard HDG methods use the same mesh and polynomial degree, they exhibit comparable accuracy.

Table 8 presents the number of degrees of freedom corresponding to the standard HDG mesh and the macro-element HDG mesh depicted in Figure 6a,b. Both HDG variants use the same elements. However, in the standard HDG method, the elements are fully discontinuous ($m = 1$), while in the macro-element HDG method, four triangular elements are grouped together to form one C^0 continuous macro-element ($m = 2$). It can be seen that the macro-element HDG approach results in a reduction in the total number of degrees of freedom in both the local and global problems. Additionally, the macro-element HDG method demonstrates a significant advantage in computational speed, particularly in reducing the time required for solving the global problem. For moderate polynomial degrees, such as $p = 3$, the global solver time for the macro-element HDG method with $m = 2$ is approximately an order of magnitude faster than that of the standard HDG method. This improvement is partly due to a reduction in the number of iterations, from 6592 to 1997. Consequently, in the macro-element HDG approach, the computational time shifts from the global solver to the local solver, increasing the efficiency of parallelization. As a result, the ratio of time spent between the local and global solvers is reduced from approximately 15 in the standard HDG method to around half in the macro-element HDG method with $m = 2$. This example illustrates that for moderate polynomial degrees such as $p = 3$, where the standard HDG method is typically inefficient, subdividing a single hybridized DG triangle element into a macro-element consisting of just four C^0-continuous elements offers a clear computational advantage.

5.4 | Compressible Taylor–Green Vortex

As the final benchmark, we consider the Taylor–Green vortex flow [68–70] on a cube $[-\pi L, \pi L]^3$. We prescribe periodic boundary conditions in all coordinate directions, assume $M_0 = 0.1$, and use the following initial conditions:

$$
\begin{aligned}
v_1(\mathbf{x}, t=0) &= V_0 \sin\left(\frac{x_1}{L}\right)\cos\left(\frac{x_2}{L}\right)\cos\left(\frac{x_3}{L}\right)\\
v_2(\mathbf{x}, t=0) &= -V_0 \cos\left(\frac{x_1}{L}\right)\sin\left(\frac{x_2}{L}\right)\cos\left(\frac{x_3}{L}\right)\\
v_3(\mathbf{x}, t=0) &= 0\\
P(\mathbf{x}, t=0) &= P_0 + \frac{\rho_0 V_0^2}{16}\left(\cos\left(\frac{2x_1}{L}\right) + \cos\left(\frac{2x_2}{L}\right)\right)\left(\cos\left(\frac{2x_3}{L}\right) + 2\right)
\end{aligned}
\tag{32}
$$

where $L = 1$, $\rho_0 = 1$, and $P_0 = 1/\gamma$. The flow is initialized to be isothermal, that is, $P/\rho = P_0/\rho_0$. The flow is computed at two Reynolds numbers, which are $Re = 100$ and $Re = 400$. The unsteady simulation is performed for a duration of $15t_c$, where $t_c = L/V_0$ is the characteristic convective time, $V_0 = M_0 c_0$, and c_0 is the speed of sound corresponding to P_0 and ρ_0, $c_0^2 = \gamma P_0/\rho_0$.

5.4.1 | Macro-Element HDG Discretization

We employ our macro-element HDG scheme with $mp = 8$, that is, the polynomial degrees chosen are $p = 4$ for $m = 2$ and $p = 2$ for $m = 4$. The corresponding effective resolutions of the cube are denoted as $N_{\text{eff}} = \left(N_{\text{ele,1d}} \cdot (mp+1)\right)^3$, where $N_{\text{ele,1d}}$ denotes the number of macro-elements in one spatial direction, see Figure 7 for an illustration. We adjust the effective resolution of the mesh according to the Reynolds number. In our case, we employ $N_{\text{eff}} = 54^3$ for $Re = 100$, and $N_{\text{eff}} = 99^3$ for 400. Figure 7 illustrates the two macro-element HDG meshes for $m = 2$, where the edges of the discontinuous macro-elements are plotted in black and the edges of the C^0-continuous elements within each macro-element

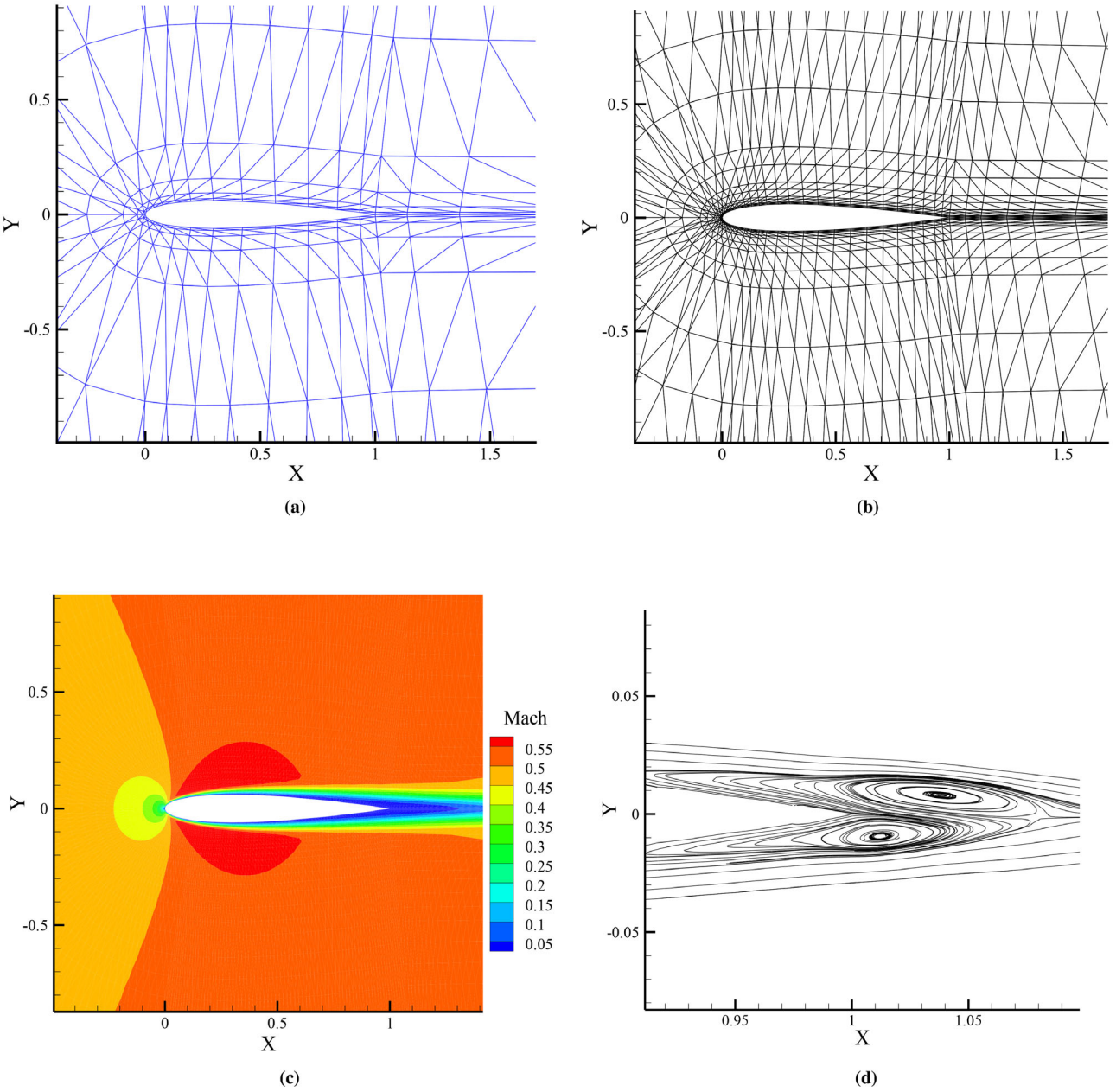


FIGURE 6 | Viscous subsonic flow around a NACA0012 airfoil. (a) Macro-element mesh. (b) Corresponding finite elements for $m = 2$ (discontinuous across macro-element interfaces). (c) Mach number contours. (d) Laminar separation bubble. [Colour figure can be viewed at wileyonlinelibrary.com]

TABLE 8 | Viscous subsonic flow over NACA 0012 airfoil: We compare the time for the local solver and the local part of the matrix-free global solver (Step 2) versus the time for the remaining parts of the global solver for 15,360 elements and $Proc's = 96$.

Time local op's [s]		Time global op's [s]		dof^{local}		dof^{global}		# iterations	
$m = 1$	$m = 2$	$m = 1$	$m = 2$	$m = 1$	$m = 2$	$m = 1$	$m = 2$	$m = 1$	$m = 2$
7.4	9.8	112.4	15.8	1,843,200	1,290,240	371,648	163,912	6592	1997

are plotted in black. In Table 9, we report the number of degrees of freedom for both local and global problems, along with the corresponding number of processes, for the two Reynolds numbers under consideration.

We set the time step such that a *CFL* number of the order of one is maintained, which represents the relation between the convective speed, the resolution length and the time scale. Specifically, we use a time step of

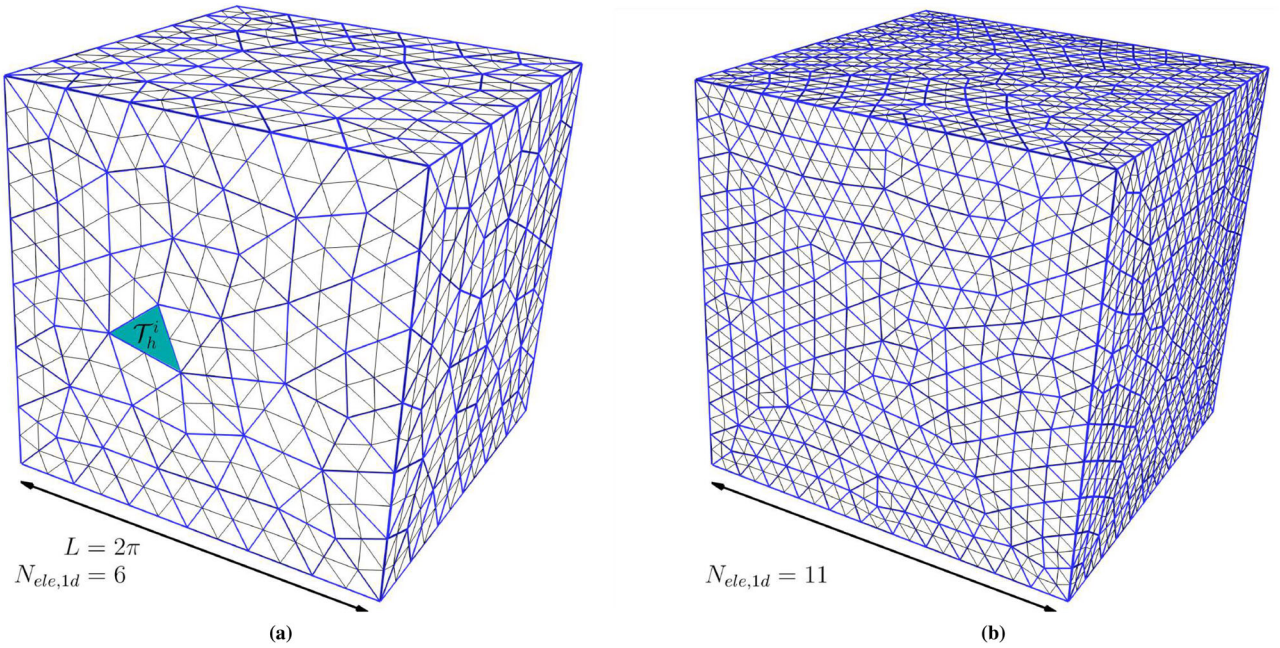


FIGURE 7 | Taylor–Green vortex: Unstructured meshes of the cube for $m = 2$. (a) Mesh for $Re = 100$. (b) Mesh for $Re = 400$. [Colour figure can be viewed at wileyonlinelibrary.com]

TABLE 9 | Taylor–Green vortex: Number of local and global unknowns for $mp = 8$ at two different mesh resolutions for $Re = 100$ and $Re = 400$.

	Mesh			#degrees of freedom	
	$N_{ele,1d}$	N_{eff}	N^{Mcr}	dof^{local}	dof^{global}
$Re = 100$	6	54^3	1592	5,253,600	716,400
$Re = 400$	11	99^3	10,143	33,471,900	4,564,350

$$\Delta t = CFL \frac{h}{V_0(p+1)} \tag{33}$$

where h is the characteristic element length and p is the polynomial of order. For this test case, our solution approach is used with absolute solver tolerance of 10^{-12} and relative solver tolerance of 10^{-6}.

5.4.2 | Verification of Accuracy

We first use the Taylor–Green vortex benchmark to investigate the accuracy that we can obtain with our macro-element HDG method. In terms of discretization accuracy and from a physical perspective, the focus lies on the kinetic energy dissipation rate, illustrated in Figure 8. In this context, the kinetic energy dissipation rate, ε, is exactly equal to

$$\varepsilon = 2\frac{\mu}{\Omega}\int_{\Omega} \rho \frac{\omega \cdot \omega}{2} d\Omega \tag{34}$$

for incompressible flow and approximately for compressible flow at low Mach number. The vorticity, ω, is defined as $\omega = \nabla \times V$ in (34). We consider the time range $0 \leq t/t_c \leq 15$ and the two Reynolds numbers $Re = 100$ and $Re = 400$. The obtained results will be compared against a reference incompressible flow solution [43].

The temporal evolution of the flow field is illustrated through iso-contours of vorticity magnitude, specifically $L/V_0 \cdot \mid \omega \mid = 1.0$, as shown in Figure 9 for the case of $Re = 100$, computed with the macro-element HDG method with the shown mesh and $(m, p) = (4, 2)$. In the early stages, corresponding to the initial time, the large-scale vortex structures initiate their evolution and exhibit a rolling-up phenomenon. Around the non-dimensional time instant $t/t_c = 6$, the smooth vertical structures give rise to more coherent formations, and by approximately $t/t_c = 9$, these coherent structures commence breaking down. Also, a snapshot of the vorticity magnitude $\mid \omega \mid$ on the periodic plane $x = \pi L$, computed with $(m, p) = (4, 2)$ at the non-dimensional time instants $t/t_c = 3.0, 9.0, 12.0$ and $Re = 100$ is plotted in Figure 10. We conclude from our results that for the benchmark at the chosen parameters, the macro-element HDG method delivers very good accuracy with the chosen mesh resolution and the polynomial degrees.

5.4.3 | Assessment of Computational Efficiency

In the next step, we investigate the computational efficiency of the macro-element HDG method for the chosen meshes and polynomial degrees. In Table 10, we report the time and number of iterations for the FGMRES linear solver with second-layer static condensation. In particular, we compare timings of

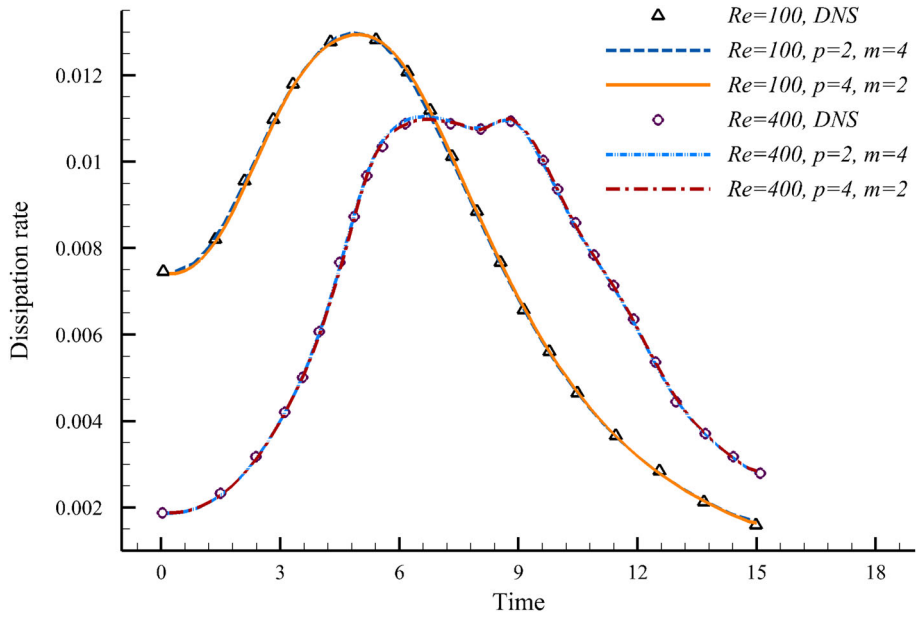


FIGURE 8 | Taylor–Green vortex: Time evolution of kinetic energy dissipation rates, computed for $Re = 100$ and $Re = 400$ on the two different meshes shown in Figure 7 with $(m, p) = (2, 4)$ and $(m, p) = (4, 2)$.

the macro-element HDG method with eight elements per macro-element ($m = 2$) and the macro-element HDG method with 64 elements per macro-element ($m = 4$). We observe that for both mesh resolutions considered here, the computational time required for the global solver decreases as m is increased. This reduction can be expected since the number of FGMRES iterations decreases and the structure of the global matrix changes. We note that this reduction becomes more pronounced with the increase of the mesh resolution for the higher Reynolds number $Re = 400$. This improvement can be attributed to an improved performance of our macro-element HDG implementation for larger mesh sizes and the deployment of a larger number of parallel processes.

We then compare the time evolution of kinetic energy dissipation rates of the two macro-element HDG variants with h refinement, see Figure 11. This comparison is based on computing time for both local and global operations, as well as the number of FGMRES iterations. We compute our example at $Re = 100$, and transition to a variable number of processors while maintaining a fixed ratio of the number of macro-elements to the number of processors at two (macro-elements/processors = 2). Table 11 reports the number of degrees of freedom for a sequence of three meshes generated by globally increasing the number of macro-elements, $N_{ele,1d}$, in each spatial direction. Both macro-element HDG variants, which use $(m, p) = (2, 4)$ and $(m, p) = (4, 2)$ utilize the same macro-element meshes, hence, the same number of macro-elements N^{Mcr}, and exhibit the same number of degrees of freedom. Figure 11 illustrates that with mesh refinement, the accuracy of both macro-element HDG variants with $(m, p) = (2, 4)$ and $(m, p) = (4, 2)$ improves and approaches the DNS reference solution. We observe a slight accuracy advantage of the macro-element HDG method that uses $(m, p) = (4, 2)$, in particular for the coarsest macro-element mesh (Mesh 1).

Table 12 compares the time for the local parts of the solver and the remaining parts of the global solver, and also reports the number of FGMRES iterations for each case. We observe that at a fixed number of degrees of freedom, using 64 C^0-continuous elements within each macro-element at $p = 2$ is results in a significant reduction in computing time in comparison to using just 8 C^0-continuous elements per macro-element at a higher polynomial degree of $p = 4$. An essential factor contributing to this reduction is the improved conditioning of the smaller system, leading to a substantial decrease in the required number of iterations for the FGMRES solver. For the largest mesh with an effective resolution of $N_{\text{eff}} = 54^3$, the time for the global operations in the macro-element HDG method with $m = 4$ is approximately a factor of two smaller than that of the macro-element HDG method with $m = 2$, primarily due to the reduction in the number of iterations from 3863 to just 2314.

Moreover, we observe that for $m = 4$, the ratio between the computing times for local and global operations is closer to the desired optimum of one. Consequently, we conclude that the macro-element HDG method with a larger number of C^0-continuous elements per macro-element, owing to its flexibility in adjusting the computational load per macro-element, in general achieves a better balance between local and global operations. This observation indicates that for very large systems, larger macro-elements (at a lower polynomial degree) are preferable to balance local and global operations, leading to the fastest computing times for the overall problem.

6 | Summary and Conclusions

In this paper, we focused on the macro-element variant of the HDG method and investigated its performance in the analysis of steady and unsteady compressible flow problems at

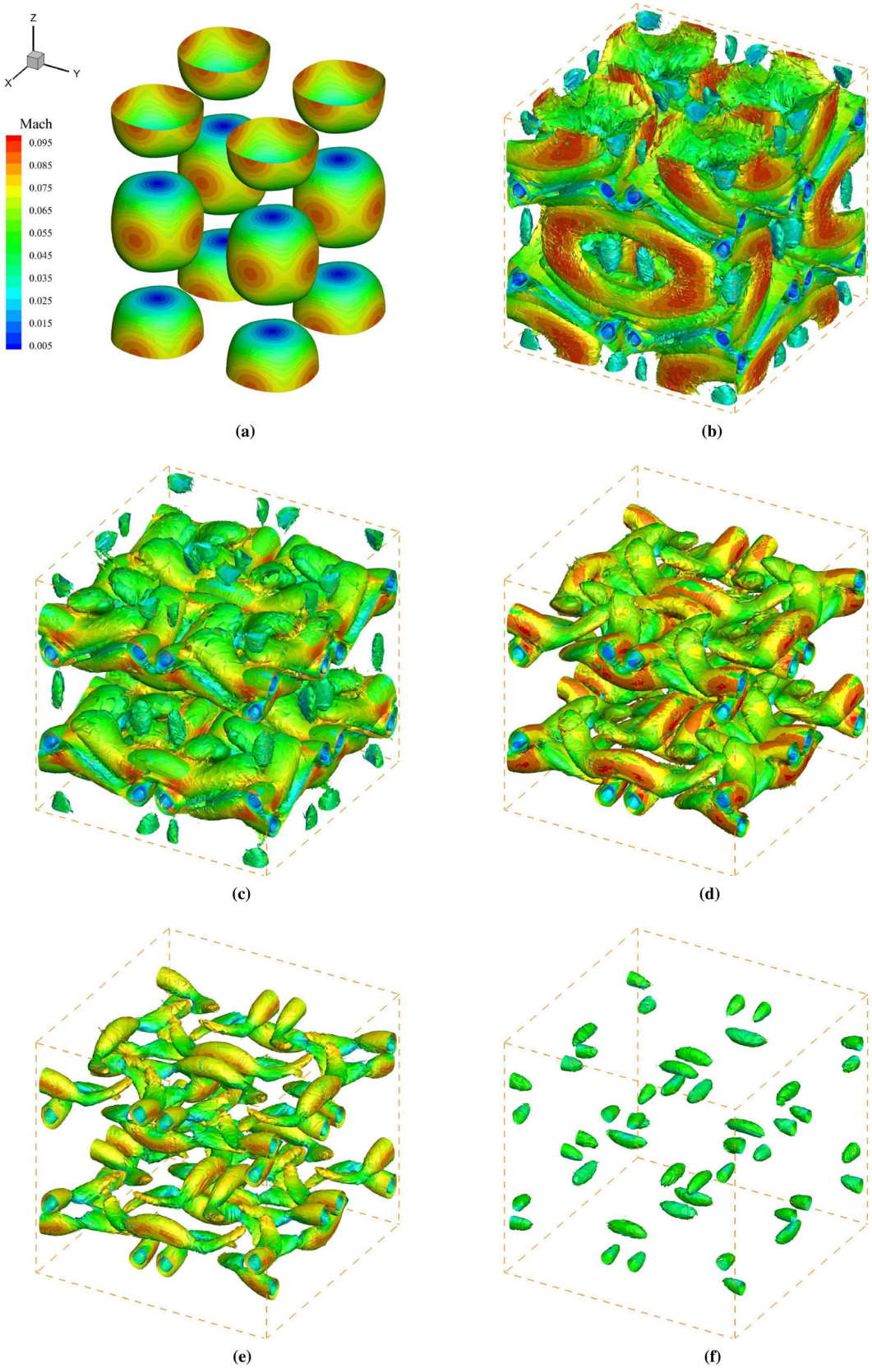


FIGURE 9 | Taylor–Green vortex: Isocontours of the vorticity magnitude $L/V_0 \cdot |\omega| = 1.0$ at $Re = 100$, computed on the mesh shown in Figure 7a with $(m, p) = (4, 2)$. (a) $t/t_c = 0.0$. (b) $t/t_c = 3.0$. (c) $t/t_c = 6.0$. (d) $t/t_c = 9.0$. (e) $t/t_c = 12.0$. (f) $t/t_c = 15.0$. [Colour figure can be viewed at wileyonlinelibrary.com]

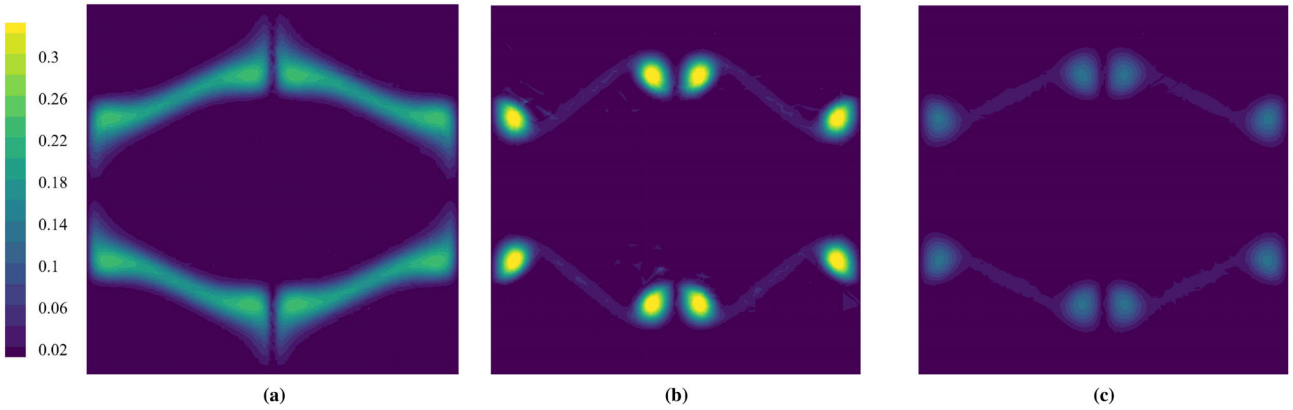


FIGURE 10 | Taylor–Green vortex: Snapshot of the vorticity magnitude $|\omega|$ at $Re = 100$, plotted on the periodic plane $x = \pi L$ for three non-dimensional time instants $t/t_c = 3.0, 9.0, 12.0$, computed on the mesh shown in Figure 7a with $(m, p) = (4, 2)$. (a) $t/t_c = 3.0$. (b) $t/t_c = 9.0$. (c) $t/t_c = 15.0$. [Colour figure can be viewed at wileyonlinelibrary.com]

TABLE 10 | Taylor–Green vortex: We compare the time for the local solver and the local part of the matrix-free global solver (Step 2) versus the time for the remaining parts of the global solver for $Re = 100$ and $Re = 400$.

	# Proc's	Time local op's [min]		Time global op's [min]		# iterations	
		$(m, p) = (2, 4)$	$(m, p) = (4, 2)$	$(m, p) = (2, 4)$	$(m, p) = (4, 2)$	$(m, p) = (2, 4)$	$(m, p) = (4, 2)$
$Re = 100$	796	7.7	4.1	18.3	8.5	3863	2314
$Re = 400$	5088	29.9	21.2	114.1	66.3	9604	8513

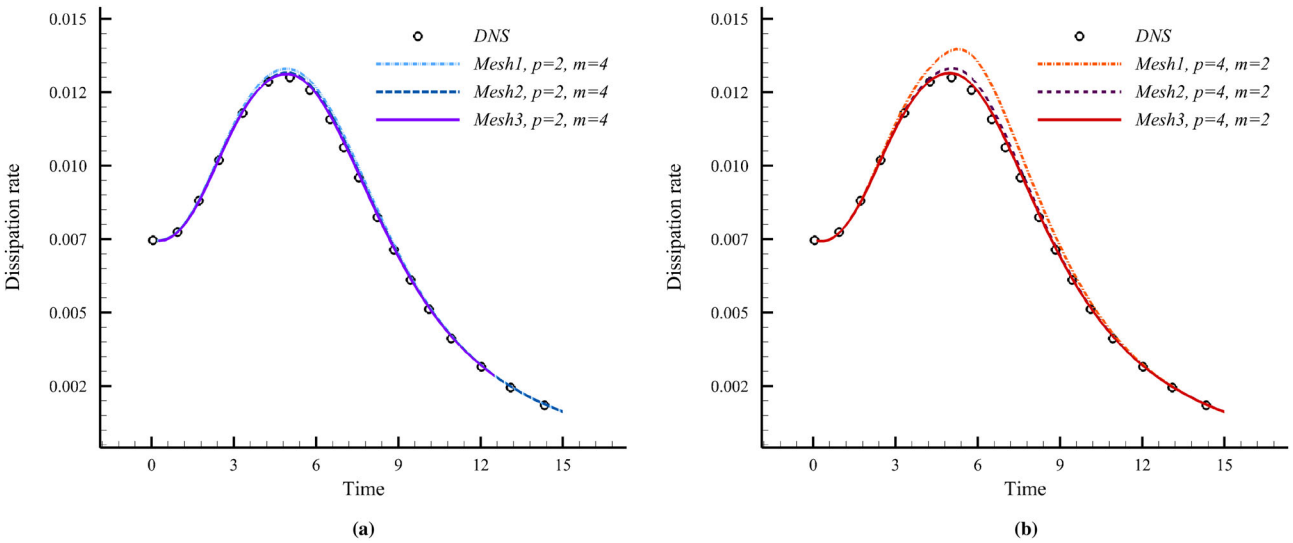


FIGURE 11 | Taylor–Green vortex: Time evolution of kinetic energy dissipation rates for $Re = 100$, obtained with the macro-element HDG method and the two different (m, p) pairs on the three different macro-element meshes defined in Table 11. (a) Kinetic energy dissipation rates for $(m, p) = (4, 2)$. (b) Kinetic energy dissipation rates for $(m, p) = (2, 4)$. [Colour figure can be viewed at wileyonlinelibrary.com]

moderate Reynolds numbers. Combining aspects of continuous and hybridized discontinuous finite element discretization, the macro-element HDG variant offers a number of advantages compared to the standard HDG method, such as the mitigation of the proliferation of degrees of freedom, the preservation of the unique domain decomposition mechanism, and automatic load-balancing inherent to the numerical method. In addition, the macro-element HDG method is particularly well suited for a matrix-free solution approach.

In comparison to our earlier work on scalar advection–diffusion problems, compressible flow problems lead to significantly larger systems of equations, due to the requirement of fine meshes with many elements as well as due to the $(d + 1) \times (d + 2)$ degrees

TABLE 11 | Taylor–Green vortex: Number of local and global unknowns for a sequence of three meshes, where the last one corresponds to the mesh shown in Figure 7a.

	Mesh			#degrees of freedom	
	$N_{ele,1d}$	N_{eff}	N^{Mcr}	dof^{local}	dof^{global}
Mesh 1	4	36^3	495	1,633,500	222,750
Mesh 2	5	45^3	955	3,151,500	429,750
Mesh 3	6	54^3	1592	5,253,600	716,400

Note: These values hold for both $(m, p) = (2, 4)$ and $(m, p) = (4, 2)$.

TABLE 12 | Taylor–Green vortex at $Re = 100$: We compare the time for the local solver and the local part of the matrix-free global solver (Step 2) versus the time for the remaining parts of the global solver.

	# Proc's	Time local op's [min]		Time global op's [min]		# iterations	
		$(m, p) = (2, 4)$	$(m, p) = (4, 2)$	$(m, p) = (2, 4)$	$(m, p) = (4, 2)$	$(m, p) = (2, 4)$	$(m, p) = (4, 2)$
Mesh 1	248	5.0	2.8	8.3	4.2	2520	1595
Mesh 2	478	6.6	3.6	11.9	5.9	3339	1993
Mesh 3	796	7.7	4.1	18.3	8.5	3863	2314

Note: We use the macro-element HDG method with $(m, p) = (2, 4)$ and $(m, p) = (4, 2)$ (at constant ratio macro-elements/proc.'s = 2).

of freedom per basis function. We therefore explored several computational strategies at the level of the local and the global solver, with the aim at enhancing computational efficiency and reducing memory requirements. For solving the local system per macro-element efficiently, we devised a second-layer static condensation approach that reduces the size of the local system matrix in each macro-element and hence the factorization time of the local solver. For solving the global system efficiently within a matrix-free implementation, we explored the use of a multi-level preconditioner based on the FGMRES method. Based on the multi-level FGMRES implementation available in PETSc, we used the iterative GMRES solver as a preconditioner, and in the second level, we again employed the inverse of the global system matrix computed via a matrix-free Cholesky factorization as a preconditioner for the GMRES solver.

We demonstrated the performance of our developments via parallel implementation of our macro-element HDG variant in Julia and C^{++} that we ported on a modern heterogeneous compute system (Lichtenberg II Phase 1 at the Technical University of Darmstadt, at position 253 in the TOP500 list 11/2023). Our computational results showed that the multi-level FGMRES iterative solver in conjunction with the second-layer static condensation approach indeed enhance the computational efficiency of the macro-element HDG method, benefitting from the reduction in degrees of freedom and communication across compute nodes. Our results also confirmed that unlike standard HDG, the macro-element HDG method is efficient for moderate polynomial degrees such as quadratics, as it is possible to increase the local computational load per macro-element irrespective of the polynomial degree. In the context of compressible flow simulations, we observed a shift in computational workload from the global solver to the local solver. This shift helps achieve a balance of local and global operations, enhancing parallelization efficiency as compared to our earlier work on scalar advection–diffusion problems. We demonstrated that—due to this balance, the reduction in degrees of freedom, and the reduction of the global problem size and the number of iterations for its solution—the macro-element HDG method delivers faster computing times than the standard HDG method, particularly for higher Reynolds numbers and mesh resolutions. In particular, we observed that for large-scale compressible flow computations, our macro-element variant of the HDG method is more efficient, when it employs macro-elements with more C^0-continuous elements, but at a lower polynomial degree, rather than using macro-elements with fewer C^0-continuous elements, but at a higher polynomial degree.

It remains to be seen how these properties demonstrated here for moderate Reynolds numbers transfer to high-Reynolds-number flow problems that involve turbulence. We plan to investigate this question in the future by applying the macro-element HDG method for the direct numerical simulation of turbulent flows modeled via the compressible Navier–Stokes equations.

Acknowledgments

The authors gratefully acknowledge financial support from the German Research Foundation (Deutsche Forschungsgemeinschaft) through the DFG Emmy Noether Grant SCH 1249/2-1. The authors also gratefully acknowledge the computing time provided to them on the high-performance computer Lichtenberg at the NHR Centers NHR4CES at TU Darmstadt. This is funded by the Federal Ministry of Education and Research and the State of Hesse. Open Access funding enabled and organized by Projekt DEAL.

Conflicts of Interest

The authors declare no conflicts of interest.

Data Availability Statement

The data that support the findings of this study are available from the corresponding author upon reasonable request.

Endnotes

[1] The Julia Programming Language, https://julialang.org/https://julialang.org/.

[2] Portable, Extensible Toolkit for Scientific Computation, https://petsc.org/https://petsc.org/.

[3] Linear Algebra PACKage, https://netlib.org/lapack/https://netlib.org/lapack/.

[4] High performance computing at TU Darmstadt, https://www.hrz.tu-darmstadt.de/hlr/hochleistungsrechnen/index.en.jsphttps://www.hrz.tu-darmstadt.de/hlr/hochleistungsrechnen/index.en.jsp.

References

1. D. N. Arnold, F. Brezzi, B. Cockburn, and L. D. Marini, "Unified Analysis of Discontinuous Galerkin Methods for Elliptic Problems," *SIAM Journal on Numerical Analysis* 39, no. 5 (2002): 1749–1779.

2. F. Bassi and S. Rebay, "A High-Order Accurate Discontinuous Finite Element Method for the Numerical Solution of the Compressible Navier–Stokes Equations," *Journal of Computational Physics* 131, no. 2 (1997): 267–279.

3. B. Cockburn, "Discontinuous Galerkin Methods for Computational Fluid Dynamics," *Encyclopedia of Computational Mechanics Second Edition* 5, (2018): 1–63.

4. J. S. Hesthaven and T. Warburton, *Nodal Discontinuous Galerkin Methods: Algorithms, Analysis, and Applications* (New York: Springer Science & Business Media, 2007).

5. J. Peraire and P. O. Persson, "The Compact Discontinuous Galerkin (CDG) Method for Elliptic Problems," *SIAM Journal on Scientific Computing* 30, no. 4 (2008): 1806–1824.

6. B. Cockburn, J. Gopalakrishnan, and R. Lazarov, "Unified Hybridization of Discontinuous Galerkin, Mixed, and Continuous Galerkin Methods for Second Order Elliptic Problems," *SIAM Journal on Numerical Analysis* 47, no. 2 (2009): 1319–1365.

7. N. C. Nguyen, J. Peraire, and B. Cockburn, "An Implicit High-Order Hybridizable Discontinuous Galerkin Method for Linear Convection–Diffusion Equations," *Journal of Computational Physics* 228, no. 9 (2009): 3232–3254.

8. B. Cockburn, J. Guzmán, and H. Wang, "Superconvergent Discontinuous Galerkin Methods for Second-Order Elliptic Problems," *Mathematics of Computation* 78, no. 265 (2009): 1–24.

9. N. C. Nguyen and J. Peraire, "Hybridizable Discontinuous Galerkin Methods for Partial Differential Equations in Continuum Mechanics," *Journal of Computational Physics* 231, no. 18 (2012): 5955–5988.

10. J. Peraire, N. Nguyen, and B. Cockburn, "A Hybridizable Discontinuous Galerkin Method for the Compressible Euler and Navier-Stokes Equations," in *48th AIAA Aerospace Sciences Meeting Including the New Horizons Forum and Aerospace Exposition* (Reston, Virginia: American Institute of Aeronautics and Astronautics (AIAA), 2010), 363.

11. B. Cockburn, B. Dong, J. Guzmán, M. Restelli, and R. Sacco, "A Hybridizable Discontinuous Galerkin Method for Steady-State Convection-Diffusion-Reaction Problems," *SIAM Journal on Scientific Computing* 31, no. 5 (2009): 3827–3846.

12. N. C. Nguyen, J. Peraire, and B. Cockburn, "An Implicit High-Order Hybridizable Discontinuous Galerkin Method for the Incompressible Navier–Stokes Equations," *Journal of Computational Physics* 230, no. 4 (2011): 1147–1170.

13. P. Fernandez, N. C. Nguyen, and J. Peraire, "The Hybridized Discontinuous Galerkin Method for Implicit Large-Eddy Simulation of Transitional Turbulent Flows," *Journal of Computational Physics* 336 (2017): 308–329.

14. J. Vila-Pérez, M. Giacomini, R. Sevilla, and A. Huerta, "Hybridisable Discontinuous Galerkin Formulation of Compressible Flows," *Archives of Computational Methods in Engineering* 28, no. 2 (2021): 753–784.

15. N. C. Nguyen, J. Peraire, and B. Cockburn, "High-Order Implicit Hybridizable Discontinuous Galerkin Methods for Acoustics and Elastodynamics," *Journal of Computational Physics* 230, no. 10 (2011): 3695–3718.

16. W. Pazner and P. O. Persson, "Stage-Parallel Fully Implicit Runge–Kutta Solvers for Discontinuous Galerkin Fluid Simulations," *Journal of Computational Physics* 335 (2017): 700–717.

17. M. Kronbichler and W. A. Wall, "A Performance Comparison of Continuous and Discontinuous Galerkin Methods With Fast Multigrid Solvers," *SIAM Journal on Scientific Computing* 40, no. 5 (2018): A3423–A3448.

18. M. S. Fabien, M. G. Knepley, R. T. Mills, and B. M. Rivière, "Manycore Parallel Computing for a Hybridizable Discontinuous Galerkin Nested Multigrid Method," *SIAM Journal on Scientific Computing* 41, no. 2 (2019): C73–C96.

19. X. Roca, C. Nguyen, and J. Peraire, "Scalable Parallelization of the Hybridized Discontinuous Galerkin Method for Compressible Flow," in *21st AIAA Computational Fluid Dynamics Conference* (Reston, Virginia: American Institute of Aeronautics and Astronautics (AIAA), 2013), 2939.

20. X. Roca, N. C. Nguyen, and J. Peraire, "GPU-Accelerated Sparse Matrix-Vector Product for a Hybridizable Discontinuous Galerkin Method," in *49th AIAA Aerospace Sciences Meeting Including the New Horizons Forum and Aerospace Exposition* (Orlando, Florida: American Institute of Aeronautics and Astronautics (AIAA), 2011), 687.

21. T. J. Hughes, *The Finite Element Method: Linear Static and Dynamic Finite Element Analysis* (Mineola, New York: Courier Corporation, 2012).

22. M. Paipuri, C. Tiago, and S. Fernández-Méndez, "Coupling of Continuous and Hybridizable Discontinuous Galerkin Methods: Application to Conjugate Heat Transfer Problem," *Journal of Scientific Computing* 78, no. 1 (2019): 321–350.

23. R. M. Kirby, S. J. Sherwin, and B. Cockburn, "To CG or to HDG: A Comparative Study," *Journal of Scientific Computing* 51 (2012): 183–212.

24. S. Yakovlev, D. Moxey, R. M. Kirby, and S. J. Sherwin, "To CG or to HDG: A Comparative Study in 3D," *Journal of Scientific Computing* 67, no. 1 (2016): 192–220.

25. V. Badrkhani, R. R. Hiemstra, M. Mika, and D. Schillinger, "A Matrix-Free Macro-Element Variant of the Hybridized Discontinuous Galerkin Method," *International Journal for Numerical Methods in Engineering* 124, no. 20 (2023): 4427–4452.

26. M. Kronbichler, K. Kormann, and W. A. Wall, "Fast Matrix-Free Evaluation of Hybridizable Discontinuous Galerkin Operators," in *Numerical Mathematics and Advanced Applications ENUMATH 2017* (Cham, Switzerland: Springer, 2019), 581–589.

27. P. F. Lermusiaux, P. J. Haley, Jr., and N. K. Yilmaz, "Environmental Prediction, Path Planning and Adaptive Sampling-Sensing and Modeling for Efficient Ocean Monitoring, Management and Pollution Control," *Sea Technology* 48, no. 9 (2007): 35–38.

28. A. N. Brooks and T. J. Hughes, "Streamline Upwind/Petrov-Galerkin Formulations for Convection Dominated Flows With Particular Emphasis on the Incompressible Navier-Stokes Equations," *Computer Methods in Applied Mechanics and Engineering* 32, no. 1–3 (1982): 199–259.

29. J. Donea and A. Huerta, *Finite Element Methods for Flow Problems* (Hoboken, NJ: John Wiley & Sons, 2003).

30. F. Xu, G. Moutsanidis, D. Kamensky, et al., "Compressible Flows on Moving Domains: Stabilized Methods, Weakly Enforced Essential Boundary Conditions, Sliding Interfaces, and Application to Gas-Turbine Modeling," *Computers & Fluids* 158 (2017): 201–220.

31. F. Shakib, T. J. Hughes, and Z. Johan, "A New Finite Element Formulation for Computational Fluid Dynamics: X. The Compressible Euler and Navier-Stokes Equations," *Computer Methods in Applied Mechanics and Engineering* 89, no. 1–3 (1991): 141–219.

32. T. E. Tezduyar and M. Senga, "Stabilization and Shock-Capturing Parameters in SUPG Formulation of Compressible Flows," *Computer Methods in Applied Mechanics and Engineering* 195, no. 13–16 (2006): 1621–1632.

33. T. E. Tezduyar, M. Senga, and D. Vicker, "Computation of Inviscid Supersonic Flows Around Cylinders and Spheres With the SUPG Formulation and YZ β Shock-Capturing," *Computational Mechanics* 38 (2006): 469–481.

34. R. Alexander, "Diagonally Implicit Runge–Kutta Methods for Stiff ODE's," *SIAM Journal on Numerical Analysis* 14, no. 6 (1977): 1006–1021.

35. H. Bijl, M. H. Carpenter, V. N. Vatsa, and C. A. Kennedy, "Implicit Time Integration Schemes for the Unsteady Compressible Navier–Stokes Equations: Laminar Flow," *Journal of Computational Physics* 179, no. 1 (2002): 313–329.

36. A. Jameson, "Time Dependent Calculations Using Multigrid, With Applications to Unsteady Flows Past Airfoils and Wings," in *10th Computational Fluid Dynamics Conference* (Reston, Virginia: American Institute of Aeronautics and Astronautics (AIAA), 1991), 1596.

37. W. A. Mulder and B. Van Leer, "Experiments With Implicit Upwind Methods for the Euler Equations," *Journal of Computational Physics* 59, no. 2 (1985): 232–246.

38. B. Cockburn, "Static Condensation, Hybridization, and the Devising of the HDG Methods," in *Building Bridges: Connections and Challenges in Modern Approaches to Numerical Partial Differential Equations* (Cham, Switzerland: Springer, 2016), 129–177.

39. N. C. Nguyen, J. Peraire, and B. Cockburn, "A Class of Embedded Discontinuous Galerkin Methods for Computational Fluid Dynamics," *Journal of Computational Physics* 302 (2015): 674–692.

40. M. Kronbichler, S. Schoeder, C. Müller, and W. A. Wall, "Comparison of Implicit and Explicit Hybridizable Discontinuous Galerkin Methods for the Acoustic Wave Equation," *International Journal for Numerical Methods in Engineering* 106, no. 9 (2016): 712–739.

41. W. Bangerth, T. Heister, L. Heltai, et al., "The Dealii Library, Version 8.2," *Archive of Numerical Software* 3, no. 100 (2015): 19–24.

42. R. Anderson, J. Andrej, A. Barker, et al., "MFEM: A Modular Finite Element Methods Library," *Computers & Mathematics with Applications* 81 (2021): 42–74.

43. D. Arndt, N. Fehn, G. Kanschat, et al., "ExaDG: High-Order Discontinuous Galerkin for the Exa-Scale," in *Software for Exascale Computing-SPPEXA 2016-2019* (Cham, Switzerland: Springer International Publishing, 2020), 189–224.

44. J. Vila-Pérez, R. L. Van Heyningen, N. C. Nguyen, and J. Peraire, "Exasim: Generating Discontinuous Galerkin Codes for Numerical Solutions of Partial Differential Equations on Graphics Processors," *SoftwareX* 20 (2022): 101212.

45. C. D. Cantwell, D. Moxey, A. Comerford, et al., "Nektar++: An Open-Source Spectral/Hp Element Framework," *Computer Physics Communications* 192 (2015): 205–219.

46. M. Kronbichler and K. Ljungkvist, "Multigrid for Matrix-Free High-Order Finite Element Computations on Graphics Processors," *ACM Transactions on Parallel Computing (TOPC)* 6, no. 1 (2019): 1–32.

47. S. Schoeder, K. Kormann, W. A. Wall, and M. Kronbichler, "Efficient Explicit Time Stepping of High Order Discontinuous Galerkin Schemes for Waves," *SIAM Journal on Scientific Computing* 40, no. 6 (2018): C803–C826.

48. S. Schoeder, S. Sticko, G. Kreiss, and M. Kronbichler, "High-Order Cut Discontinuous Galerkin Methods With Local Time Stepping for Acoustics," *International Journal for Numerical Methods in Engineering* 121, no. 13 (2020): 2979–3003.

49. N. Fehn, W. A. Wall, and M. Kronbichler, "A Matrix-Free High-Order Discontinuous Galerkin Compressible Navier-Stokes Solver: A Performance Comparison of Compressible and Incompressible Formulations for Turbulent Incompressible Flows," *International Journal for Numerical Methods in Fluids* 89, no. 3 (2019): 71–102.

50. K. R. Wichmann, M. Kronbichler, R. Löhner, and W. A. Wall, "Practical Applicability of Optimizations and Performance Models to Complex Stencil-Based Loop Kernels in CFD," *International Journal of High Performance Computing Applications* 33, no. 4 (2019): 602–618.

51. D. A. Patterson and J. L. Hennessy, *Computer Organization and Design ARM Edition: The Hardware Software Interface* (Burlington, MA: Morgan kaufmann, 2016).

52. C. D. Cantwell, S. J. Sherwin, R. M. Kirby, and P. H. Kelly, "From h to p Efficiently: Strategy Selection for Operator Evaluation on Hexahedral and Tetrahedral Elements," *Computers & Fluids* 43, no. 1 (2011): 23–28.

53. Y. Saad and M. H. Schultz, "GMRES: A Generalized Minimal Residual Algorithm for Solving Nonsymmetric Linear Systems," *SIAM Journal on Scientific and Statistical Computing* 7, no. 3 (1986): 856–869.

54. Y. Saad, "A Flexible Inner-Outer Preconditioned GMRES Algorithm," *SIAM Journal on Scientific Computing* 14, no. 2 (1993): 461–469.

55. L. Wang and D. J. Mavriplis, "Implicit Solution of the Unsteady Euler Equations for High-Order Accurate Discontinuous Galerkin Discretizations," *Journal of Computational Physics* 225, no. 2 (2007): 1994–2015.

56. L. T. Diosady and D. L. Darmofal, "Preconditioning Methods for Discontinuous Galerkin Solutions of the Navier–Stokes Equations," *Journal of Computational Physics* 228, no. 11 (2009): 3917–3935.

57. K. J. Fidkowski, T. A. Oliver, J. Lu, and D. L. Darmofal, "p-Multigrid Solution of High-Order Discontinuous Galerkin Discretizations of the Compressible Navier–Stokes Equations," *Journal of Computational Physics* 207, no. 1 (2005): 92–113.

58. V. Badrkhani, A. Nejat, and M. Tahani, "Development of Implicit Algorithm for High-Order Discontinuous Galerkin Methods to Solve Compressible Flows Using Newton-Krylov Methods," *Modares Mechanical Engineering* 17, no. 3 (2017): 281–292.

59. L. Botti, A. Colombo, and F. Bassi, "h-Multigrid Agglomeration Based Solution Strategies for Discontinuous Galerkin Discretizations of Incompressible Flow Problems," *Journal of Computational Physics* 347 (2017): 382–415.

60. M. Franciolini, L. Botti, A. Colombo, and A. Crivellini, "p-Multigrid Matrix-Free Discontinuous Galerkin Solution Strategies for the Under-Resolved Simulation of Incompressible Turbulent Flows," *Computers & Fluids* 206 (2020): 104558.

61. S. Vakilipour, M. Mohammadi, V. Badrkhani, and S. Ormiston, "Developing a Physical Influence Upwind Scheme for Pressure-Based Cell-Centered Finite Volume Methods," *International Journal for Numerical Methods in Fluids* 89, no. 1–2 (2019): 43–70.

62. T. A. Davis, "Algorithm 832: UMFPACK V4. 3—An Unsymmetric-Pattern Multifrontal Method," *ACM Transactions on Mathematical Software (TOMS)* 30, no. 2 (2004): 196–199.

63. J. Schütz, M. Woopen, and G. May, "A Hybridized DG/Mixed Scheme for Nonlinear Advection-Diffusion Systems, Including the Compressible Navier-Stokes Equations," in *50th AIAA Aerospace Sciences Meeting Including the New Horizons Forum and Aerospace Exposition* (Reston, Virginia: American Institute of Aeronautics and Astronautics (AIAA), 2012), 729.

64. M. Franciolini, K. J. Fidkowski, and A. Crivellini, "Efficient Discontinuous Galerkin Implementations and Preconditioners for Implicit Unsteady Compressible Flow Simulations," *Computers & Fluids* 203 (2020): 104542.

65. T. Johnson and V. Patel, "Flow Past a Sphere up to a Reynolds Number of 300," *Journal of Fluid Mechanics* 378 (1999): 19–70.

66. S. Taneda, "Experimental Investigation of the Wake Behind a Sphere at Low Reynolds Numbers," *Journal of the Physical Society of Japan* 11, no. 10 (1956): 1104–1108.

67. R. Radespiel, "A Cell-Vertex Multigrid Method for the Navier-Stokes Equations," *Technical Report* (1989).

68. G. I. Taylor and A. E. Green, "Mechanism of the Production of Small Eddies From Large Ones," *Proceedings of the Royal Society of London. Series A-Mathematical and Physical Sciences* 158, no. 895 (1937): 499–521.

69. M. Brachet, "Direct Simulation of Three-Dimensional Turbulence in the Taylor–Green Vortex," *Fluid Dynamics Research* 8, no. 1–4 (1991): 1.

70. N. Fehn, W. A. Wall, and M. Kronbichler, "Efficiency of High-Performance Discontinuous Galerkin Spectral Element Methods for Under-Resolved Turbulent Incompressible Flows," *International Journal for Numerical Methods in Fluids* 88, no. 1 (2018): 32–54.