NUMERICAL MATHEMATICS: Theory, Methods and Applications
Numer. Math. Theor. Meth. Appl., Vol. **xx**, No. x, pp. 1-26 (200x)

Domain Decomposition Preconditioners for Discontinuous Galerkin Discretizations of Compressible Fluid Flows

Stefano Giani[1] and Paul Houston[2,*]

[1] *School of Engineering and Computing Sciences, Durham University, South Road, Durham, DH1 3LE, UK.*
[2] *School of Mathematical Sciences, University of Nottingham, University Park, Nottingham NG7 2RD, UK.*

Abstract. In this article we consider the application of Schwarz-type domain decomposition preconditioners to the discontinuous Galerkin finite element approximation of the compressible Navier–Stokes equations. To discretize this system of conservation laws, we exploit the (adjoint consistent) symmetric version of the interior penalty discontinuous Galerkin finite element method. To define the necessary coarse-level solver required for the definition of the proposed preconditioner, we exploit ideas from composite finite element methods, which allow for the definition of finite element schemes on general meshes consisting of polygonal (agglomerated) elements. The practical performance of the proposed preconditioner is demonstrated for a series of viscous test cases in both two– and three–dimensions.

AMS subject classifications: 65F08, 65N12, 65N15, 65N30

Key words: Composite finite element methods, discontinuous Galerkin methods, domain decomposition, Schwarz preconditioners, compressible fluid flows.

1. Introduction

The application and development of discontinuous Galerkin finite element methods (DGFEMs) for the numerical approximation of the compressible Euler and Navier-Stokes equations has been considered extensively within the current literature; for example, see [12–15, 18, 21, 22, 25, 27–29, 33–35], and the references cited therein. DGFEMs have several important advantages over well established finite volume methods. The concept of higher-order discretization is inherent to the DGFEM. The stencil is minimal in the sense that each element communicates only with its direct neighbours. In particular, in contrast to the increasing stencil size needed to increase the accuracy of classical finite volume methods, the stencil of DGFEMs is the same for any order of accuracy, which has important

*Corresponding author. *Email addresses:* `stefano.giani@durham.ac.uk` (Stefano Giani), `Paul.Houston@nottingham.ac.uk` (Paul Houston) advantages for the implementation of boundary conditions and for the parallel efficiency of the method. Moreover, due to the simple communication at element interfaces, elements with so-called hanging nodes can be easily treated, a fact that simplifies local mesh refinement (*h*–refinement). Additionally, the communication at element interfaces is identical for any order of the method, which simplifies the use of methods with different polynomial orders p in adjacent elements. This allows for the variation of the order of polynomials over the computational domain (p–refinement), which in combination with h–refinement leads to *hp*–adaptivity. It should also be noted that the use of standard conforming methods for the discretization of compressible flows suffer from issues concerning numerical stability, which must be tackled with the introduction of suitable numerical dissipation terms in the form of SUPG stabilisation or artificial viscosity. Such terms can adversely affect the convergence of the underlying iterative solver used to compute the numerical solution.

Despite the advantages and capabilities of the DGFEM, the method is not yet mature and current implementations are subject to strong limitations for its application to large scale industrial problems. This situation is clearly reflected by the breadth of research activity and the increasing number of scientific articles concerning DGFEMs. In particular, one of the key issues is the design of efficient strategies for the solution of the system of equations generated by a DGFEM, which we should point out is typically larger than the corresponding matrix system generated when a conforming finite element method is employed. However, in the context of p–version finite element methods, it has been shown in the recent article [17] that DGFEMs can indeed outperform their conforming counterparts in the sense that the former class of methods may be more accurate for a given number of degrees of freedom as the polynomial degree is increased. For two–dimensional problems, parallel direct solvers such as MUMPS [1–3], for example, are generally applicable. However, for such problems, they still require very large amounts of memory in order to store the L and U factors. Moreover, for three-dimensional calculations, direct methods become impractical. Thereby, in this setting iterative solvers, such as GMRES, for example, must be exploited. Of course, the key to computing the solution in an efficient manner relies on the choice of the underlying preconditioning strategy employed. In order to exploit the parallel capabilities of modern high performance computing architectures, it is natural to consider multilevel techniques, which are based on exploiting some form of domain decomposition approach, such as additive and multiplicative Schwarz preconditioners, since they are naturally highly-parallelizable and scalable to a large number of processors.

In the context of DGFEMs, recent work on the design and analysis of multilevel preconditioners for DGFEMs has been undertaken; for example, we refer to [4, 5, 8, 10, 16, 20]. In particular, in [4, 5], cf., also, [8], it was demonstrated that Schwarz–type preconditioners are particularly suited to DGFEMs, in the sense that uniform scalability of the underlying iterative method may be established without the need to overlap the subdomain partition of the computational mesh. This is a particularly attractive property, since the absence of overlapping subdomains reduces communication between processors on parallel machines. By (uniform) scalability, we mean that the number of iterations needed to compute the solution of the underlying system of equations is uniform, as the mesh is refined, provided that an appropriate coarse–level solution is computed as part of the preconditioning strategy. In our recent article [7], we considered Schwarz–type preconditioners based on employing a composite DGFEM, which exploits general meshes consisting of polygonal (agglomerated) elements, cf. [6, 24], as the coarse-level solver. This class of methods allows for very coarse domain-conforming meshes to be employed; in the context of designing multilevel preconditioners, they provide a flexible mathematical and practical framework within which coarse level approximations may be computed. In this article we extend this work to consider the application of these multi-level preconditioners for application to compressible flows in both two– and three–dimensions. In particular, we employ a nonlinear (damped) Newton algorithm, whereby the inner linear solves are computed using a preconditioned GMRES solver, with an additive/multiplicative Schwarz preconditioner. We study the dependence of the convergence of the underlying iterative solver with respect to different mesh partitioning strategies employing both structured and general unstructured hybrid meshes. The numerical results presented in this article clearly highlight the efficiency of the proposed solution algorithm on general finite element meshes.

This article is structured as follows. In Section 2 we introduce the three–dimensional compressible Navier–Stokes equations. Then, in Section 3 we formulate its discontinuous Galerkin finite element approximation, based on employing the adjoint consistent symmetric interior penalty method introduced in [30]. Section 4 outlines a damped Newton–GMRES algorithm for the solution of the system of nonlinear equations arising from the DGFEM discretization of the underlying PDE system. Section 5 is devoted to defining the composite DGFEM (DGCFEM), which represents a natural extension of the standard DGFEM on (coarse) agglomerated meshes; based on exploiting the DGCFEM as a coarse–level solver, in Section 6 we construct the Schwarz preconditioners for application within the Newton–GMRES iteration. In Section 8 we present some numerical results obtained with the additive Schwarz preconditioner to highlight the practical performance of the proposed solver. Finally, in Section 9 we summarize the work presented in this paper.

2. Compressible Navier-Stokes equations

In this article, we consider both two– and three–dimensional laminar compressible flow problems. With this in mind, for generality, in this section we introduce the stationary compressible Navier-Stokes equations in three-dimensions:

$$\nabla \cdot (\mathscr{F}^c(\mathbf{u}) - \mathscr{F}^v(\mathbf{u}, \nabla \mathbf{u})) = 0 \quad \text{in } \Omega, \tag{2.1}$$

where Ω is an open bounded domain in $\mathbb{R}^d$ with boundary Γ; for the purposes of this section, we set $d = 3$. The vector of conservative variables $\mathbf{u}$ is given by $\mathbf{u} = (\rho, \rho v_1, \rho v_2, \rho v_3, \rho E)^\top$ and the convective flux $\mathscr{F}^c(\mathbf{u}) = \left(\mathbf{f}_1^c(\mathbf{u}), \mathbf{f}_2^c(\mathbf{u}), \mathbf{f}_3^c(\mathbf{u})\right)^\top$ is defined by $\mathbf{f}_1^c(\mathbf{u}) = (\rho v_1, \rho v_1^2 + p, \rho v_1 v_2, \rho v_1 v_3, \rho H v_1)^\top$, $\mathbf{f}_2^c(\mathbf{u}) = (\rho v_2, \rho v_2 v_1, \rho v_2^2 + p, \rho v_2 v_3, \rho H v_2)^\top$, and $\mathbf{f}_3^c(\mathbf{u}) = (\rho v_3, \rho v_3 v_1, \rho v_3 v_2, \rho v_3^2 + p, \rho H v_3)^\top$. Furthermore, writing $\mathscr{F}^v(\mathbf{u}) = \left(\mathbf{f}_1^v(\mathbf{u}), \mathbf{f}_2^v(\mathbf{u}), \mathbf{f}_3^v(\mathbf{u})\right)^\top$, we have $\mathbf{f}_k^v(\mathbf{u}, \nabla \mathbf{u}) = (0, \tau_{1k}, \tau_{2k}, \tau_{3k}, \tau_{kl} v_l + \mathscr{K} T_{x_k})^\top$, $k = 1, 2, 3$. Here, ρ, $\mathbf{v} = (v_1, v_2, v_3)^\top$, p, E and T denote the density, velocity vector, pressure, specific total energy, and temperature, respectively. Moreover, $\mathscr{K}$ is the thermal conductivity coefficient and H is the total enthalpy given by $H = E + \frac{p}{\rho} = e + \frac{1}{2}\mathbf{v}^2 + \frac{p}{\rho}$, where e is the specific static internal energy, and the pressure is determined by the equation of state of an ideal gas

$$p = (\gamma - 1)\rho e, \tag{2.2}$$

where γ is the ratio of specific heat capacities; for dry air, $\gamma = 1.4$. The viscous stress tensor is given by $\tau = \mu\left(\nabla\mathbf{v} + (\nabla\mathbf{v})^\top - \frac{2}{3}(\nabla\cdot\mathbf{v})I\right)$, where μ is the dynamic viscosity coefficient; T is given by $\mathscr{K}T = \mu\gamma\left(E - \frac{1}{2}\mathbf{v}^2\right)/\mathrm{Pr}$, where $\mathrm{Pr} = 0.72$ is the Prandtl number. We rewrite the Navier–Stokes equations (2.1) in the following form:

$$\nabla\cdot(\mathscr{F}^c(\mathbf{u}) - G(\mathbf{u})\nabla\mathbf{u}) \equiv \frac{\partial}{\partial x_k}\left(\mathbf{f}_k^c(\mathbf{u}) - G_{kl}(\mathbf{u})\frac{\partial\mathbf{u}}{\partial x_l}\right) = 0 \quad \text{in } \Omega.$$

Here, the matrices $G_{kl}(\mathbf{u}) = \partial\mathbf{f}_k^v(\mathbf{u},\nabla\mathbf{u})/\partial u_{x_l}$, for $k,l = 1,2,3$, are the homogeneity tensors defined by $\mathbf{f}_k^v(\mathbf{u},\nabla\mathbf{u}) = G_{kl}(\mathbf{u})\partial\mathbf{u}/\partial x_l$, $k = 1,2,3$.

To prescribe boundary conditions on Γ, we assume that Γ may be decomposed as follows: $\Gamma = \Gamma_{\text{D,sup}} \cup \Gamma_{\text{D,sub-in}} \cup \Gamma_{\text{D,sub-out}} \cup \Gamma_{\text{W}} \cup \Gamma_{\text{sym}}$, where $\Gamma_{\text{D,sup}}$, $\Gamma_{\text{D,sub-in}}$, $\Gamma_{\text{D,sub-out}}$, Γ_{W}, and Γ_{sym} are distinct subsets of Γ representing Dirichlet (supersonic), Dirichlet (subsonic-inflow), Dirichlet (subsonic-outflow), solid wall boundaries, and symmetry boundaries, respectively, cf. [23, 28]. We specify the following boundary conditions:

$$\mathscr{B}(\mathbf{u}) = \mathscr{B}(\mathbf{g}) \;\text{ on }\; \Gamma_{\text{D,sup}} \cup \Gamma_{\text{D,sub-in}} \cup \Gamma_{\text{D,sub-out}},$$

where $\mathbf{g} = (g_1,\ldots,g_5)^\top$ is a prescribed Dirichlet condition. Here, $\mathscr{B}$ is a boundary operator employed to enforce appropriate Dirichlet conditions on $\Gamma_{\text{D,sup}} \cup \Gamma_{\text{D,sub-in}} \cup \Gamma_{\text{D,sub-out}}$. For simplicity of presentation, we assume that

$$\mathscr{B}(\mathbf{u}) = \begin{cases} \mathbf{u} & \text{on } \Gamma_{\text{D,sup}}, \\ (u_1,u_2,u_3,u_4,0)^\top & \text{on } \Gamma_{\text{D,sub-in}}, \\ \left(0,0,0,0,(\gamma-1)(u_5 - (u_2^2+u_3^2+u_4^2)/(2u_1))\right)^\top & \text{on } \Gamma_{\text{D,sub-out}}. \end{cases}$$

For solid wall boundaries, we consider *isothermal* and *adiabatic* conditions; to this end, decomposing $\Gamma_W = \Gamma_{\text{iso}} \cup \Gamma_{\text{adia}}$, we set $\mathbf{v} = \mathbf{0}$ on Γ_{W}, $T = T_{\text{wall}}$ on Γ_{iso}, $\mathbf{n}\cdot\nabla T = 0$ on Γ_{adia}, where T_{wall} is a given wall temperature; cf. [12, 14, 18, 19]. On the symmetry boundary, we simply impose that the normal component of the velocity is zero.

3. DGFEM Discretization

We introduce the adjoint-consistent interior penalty DGFEM discretization of the compressible Navier–Stokes equations (2.1), cf. [30]. First, we begin by introducing some notation. We assume that $\Omega \subset \mathbb{R}^d$, $d = 2,3$, can be subdivided into a mesh $\mathscr{T}_h = \{\kappa\}$ consisting of open element domains κ. For each $\kappa \in \mathscr{T}_h$, we denote by $\mathbf{n}_\kappa$, the unit outward normal vector to the boundary $\partial\kappa$. Given a polynomial approximation order $p \geq 1$, we introduce the finite element space $V(\mathscr{T}_h,p) = \{\mathbf{v} \in \left[L_2(\Omega)\right]^{d+2} : \mathbf{v}|_\kappa \in \left[\mathscr{S}_p(\kappa)\right]^{d+2} \quad \forall\kappa \in \mathscr{T}_h\}$, where $\mathscr{S}_p(\kappa)$, is either the space $\mathscr{P}_p(\kappa)$ of polynomials of degree at most p if κ is a simplex, or the space $\mathscr{Q}_p(\kappa)$ of all tensor product polynomials of degree at most p in each variable if κ is a hypercube.

An *interior face* of $\mathscr{T}_h$ is defined as the $(d-1)$–dimensional interior of $\partial\kappa^+ \cap \partial\kappa^-$, where κ^+ and κ^- are two adjacent elements of $\mathscr{T}_h$, not necessarily matching. A *boundary face* of $\mathscr{T}_h$ is defined as the (non-empty) $(d-1)$–dimensional interior of $\partial\kappa \cap \Gamma$, where κ is a boundary element of $\mathscr{T}_h$. We denote by $\Gamma_{\mathscr{I}_h}$ the union of all interior faces of $\mathscr{T}_h$. Let κ^+ and κ^- be two adjacent elements of $\mathscr{T}_h$, and $\mathbf{x}$ an arbitrary point on the interior face $f = \partial\kappa^+ \cap \partial\kappa^-$. Furthermore, let $\mathbf{v}$ and $\underline{\tau}$ be vector- and matrix-valued functions, respectively, that are smooth inside each element $\kappa^{\pm}$. By $(\mathbf{v}^{\pm}, \underline{\tau}^{\pm})$, we denote the traces of $(\mathbf{v}, \underline{\tau})$ on f taken from within the interior of $\kappa^{\pm}$, respectively. Then, the averages of $\mathbf{v}$ and $\underline{\tau}$ at $\mathbf{x} \in f$ are given by $\{\!\{\mathbf{v}\}\!\} = (\mathbf{v}^+ + \mathbf{v}^-)/2$ and $\{\!\{\underline{\tau}\}\!\} = (\underline{\tau}^+ + \underline{\tau}^-)/2$, respectively. Similarly, the jump of $\mathbf{v}$ at $\mathbf{x} \in f$ is given by $\underline{[\![\mathbf{v}]\!]} = \mathbf{v}^+ \otimes \mathbf{n}_{\kappa^+} + \mathbf{v}^- \otimes \mathbf{n}_{\kappa^-}$, where we denote by $\mathbf{n}_{\kappa^{\pm}}$ the unit outward normal vector of $\kappa^{\pm}$, respectively. On $f \subset \Gamma$, we set $\{\!\{\mathbf{v}\}\!\} = \mathbf{v}$, $\{\!\{\underline{\tau}\}\!\} = \underline{\tau}$ and $\underline{[\![\mathbf{v}]\!]} = \mathbf{v} \otimes \mathbf{n}$, where $\mathbf{n}$ denotes the unit outward normal vector to Γ.

The DGFEM discretization of (2.1) is given by: find $\mathbf{u}_h \in V(\mathscr{T}_h, p)$ such that

$$\begin{aligned}\mathscr{N}(\mathbf{u}_h, \mathbf{v}) \equiv & -\int_\Omega \mathscr{F}^c(\mathbf{u}_h) : \nabla_h \mathbf{v}\, \mathrm{d}\mathbf{x} + \sum_{\kappa \in \mathscr{T}_h} \int_{\partial\kappa \setminus \Gamma} \mathscr{H}(\mathbf{u}_h^+, \mathbf{u}_h^-, \mathbf{n}^+) \cdot \mathbf{v}^+ \, \mathrm{d}s \\ & + \int_\Omega \mathscr{F}^v(\mathbf{u}_h, \nabla_h \mathbf{u}_h) : \nabla_h \mathbf{v}\, \mathrm{d}\mathbf{x} - \int_{\Gamma_{\mathscr{I}_h}} \{\!\{\mathscr{F}^v(\mathbf{u}_h, \nabla_h \mathbf{u}_h)\}\!\} : \underline{[\![\mathbf{v}]\!]}\, \mathrm{d}s - \int_{\Gamma_{\mathscr{I}_h}} \{\!\{G^\top(\mathbf{u}_h)\nabla_h \mathbf{v}\}\!\} : \underline{[\![\mathbf{u}_h]\!]}\, \mathrm{d}s \\ & + \int_{\Gamma_{\mathscr{I}_h}} \underline{\delta}(\mathbf{u}_h) : \underline{[\![\mathbf{v}]\!]}\, \mathrm{d}s + \mathscr{N}_{\Gamma \setminus \Gamma_{\mathrm{sym}}}(\mathbf{u}_h, \mathbf{v}) + \mathscr{N}_{\Gamma_{\mathrm{sym}}}(\mathbf{u}_h, \mathbf{v}) = 0 \end{aligned} \tag{3.1}$$

for all $\mathbf{v}$ in $V(\mathscr{T}_h, p)$. Here, ∇_h denotes the elementwise gradient operator and $\mathscr{H}(\cdot,\cdot,\cdot)$ is the (convective) numerical flux function; here, we employ the Vijayasundaram flux.

In order to define the penalization function $\underline{\delta}(\cdot)$ arising in the DGFEM (3.1), we first introduce the local (anisotropic) mesh function $\mathtt{h}$. To this end, the function $\mathtt{h}$ in $L_\infty(\Gamma_{\mathscr{I}_h} \cup \Gamma)$ is defined as $\mathtt{h}(\mathbf{x}) = \min\{m_{\kappa^+}, m_{\kappa^-}\}/m_f$, if $\mathbf{x}$ is in the interior of $f = \partial\kappa^+ \cap \partial\kappa^-$ for two neighbouring elements in the mesh $\mathscr{T}_h$, and $\mathtt{h}(\mathbf{x}) = m_\kappa/m_f$, if $\mathbf{x}$ is in the interior of $f = \partial\kappa \cap \Gamma$. Here, for a given (open) bounded set $\omega \subset \mathbb{R}^s$, $s \geq 1$, we write m_ω to denote the s–dimensional measure (volume) of ω. We write

$$\underline{\delta}(\mathbf{u}_h) = C_{\mathrm{IP}} \frac{p^2}{\mathtt{h}} \{\!\{G(\mathbf{u}_h)\}\!\} \underline{[\![\mathbf{u}_h]\!]},$$

where C_{IP} is a (sufficiently large) positive constant. Finally, we define the boundary terms present in the forms $\mathscr{N}_{\Gamma\setminus\Gamma_{\mathrm{sym}}}(\cdot,\cdot)$ and $\mathscr{N}_{\Gamma_{\mathrm{sym}}}(\cdot,\cdot)$. To this end, we write

$$\begin{aligned}\mathscr{N}_{\Gamma\setminus\Gamma_{\mathrm{sym}}}(\mathbf{u}_h, \mathbf{v}) = & \int_{\Gamma\setminus\Gamma_{\mathrm{sym}}} \mathscr{H}_\Gamma(\mathbf{u}_h^+, \mathbf{u}_\Gamma(\mathbf{u}_h^+), \mathbf{n}^+) \cdot \mathbf{v}^+ \, \mathrm{d}s + \int_{\Gamma\setminus\Gamma_{\mathrm{sym}}} \underline{\delta}_\Gamma(\mathbf{u}_h^+) : \mathbf{v}^+ \otimes \mathbf{n}\, \mathrm{d}s \\ & - \int_{\Gamma\setminus\Gamma_{\mathrm{sym}}} \mathbf{n} \cdot \mathscr{F}_\Gamma^v(\mathbf{u}_\Gamma(\mathbf{u}_h^+), \nabla_h \mathbf{u}_h^+) \mathbf{v}^+ \, \mathrm{d}s - \int_{\Gamma\setminus\Gamma_{\mathrm{sym}}} \left(G_\Gamma^\top(\mathbf{u}_h^+)\nabla_h \mathbf{v}_h^+\right) : \left(\mathbf{u}_h^+ - \mathbf{u}_\Gamma(\mathbf{u}_h^+)\right) \otimes \mathbf{n}\, \mathrm{d}s,\end{aligned}$$

where

$$\underline{\delta}_{\Gamma}(\mathbf{u}_h) = C_{\text{IP}} \frac{p^2}{\mathrm{h}} G_{\Gamma}(\mathbf{u}_h^+) \left(\mathbf{u}_h^+ - \mathbf{u}_{\Gamma}(\mathbf{u}_h^+)\right) \otimes \mathbf{n}.$$

Here, the viscous flux $\mathscr{F}_{\Gamma}^{v}$ and corresponding homogeneity tensor G_{Γ} are defined by

$$\mathscr{F}_{\Gamma}^{v}(\mathbf{u}_h, \nabla \mathbf{u}_h) = \mathscr{F}^{v}(\mathbf{u}_{\Gamma}(\mathbf{u}_h), \nabla \mathbf{u}_h) = G_{\Gamma}(\mathbf{u}_h)\nabla \mathbf{u}_h = G(\mathbf{u}_{\Gamma}(\mathbf{u}_h))\nabla \mathbf{u}_h.$$

Furthermore, on portions of the boundary Γ where adiabatic boundary conditions are imposed, $\mathscr{F}_{\Gamma}^{v}$ and G_{Γ} are modified such that $\mathbf{n} \cdot \nabla T = 0$. The convective boundary flux $\mathscr{H}_{\Gamma}$ is defined by $\mathscr{H}_{\Gamma}(\mathbf{u}_h^+, \mathbf{u}_{\Gamma}(\mathbf{u}_h^+), \mathbf{n}) = \mathbf{n} \cdot \mathscr{F}^{c}(\mathbf{u}_{\Gamma}(\mathbf{u}_h^+))$. The boundary function $\mathbf{u}_{\Gamma}(\mathbf{u})$ is given according to the type of boundary condition imposed. Here, we set

$$\mathbf{u}_{\Gamma}(\mathbf{u}) = \begin{cases} \mathbf{g} & \text{on } \Gamma_{\text{D,sup}}, \\ \left(g_1, g_2, g_3, g_4, \frac{p(\mathbf{u})}{\gamma-1} + (g_2^2 + g_3^2 + g_4^2)/(2g_1)\right)^{\top} & \text{on } \Gamma_{\text{D,sub-in}}, \\ \left(u_1, u_2, u_3, u_4, \frac{p_{\text{out}}}{\gamma-1} + (u_2^2 + u_3^2 + u_4^2)/(2u_1)\right)^{\top} & \text{on } \Gamma_{\text{D,sub-out}}. \end{cases}$$

Here, $p \equiv p(\mathbf{u})$ denotes the pressure evaluated using the equation of state (2.2). On Γ_{iso}, we set $\mathbf{u}_{\Gamma}(\mathbf{u}) = \left(u_1, 0, 0, 0, u_1 c_v T_{\text{wall}}\right)^{\top}$, while $\mathbf{u}_{\Gamma}(\mathbf{u}) = (u_1, 0, 0, 0, u_5)^{\top}$ on Γ_{adia}.

On Γ_{sym}, we employ the technique introduced in [31]. To this end, we define

$$\mathbf{u}_{\Gamma}(\mathbf{u}) = B_{\text{sym}} \mathbf{u} \quad \text{on} \quad \Gamma_{\text{sym}}, \tag{3.2}$$

where

$$B_{\text{sym}} = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1-2n_1^2 & -2n_1 n_2 & -2n_1 n_3 & 0 \\ 0 & -2n_1 n_2 & 1-2n_2^2 & -2n_2 n_3 & 0 \\ 0 & -2n_1 n_3 & -2n_2 n_3 & 1-2n_3^2 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

and $\mathbf{n} = (n_1, n_2, n_3)^{\top}$ is the unit outward normal vector to the boundary. Additionally, it is necessary to introduce a suitable approximation of $\nabla \mathbf{u}_h^-$; here, we write

$$(\nabla u)_{\Gamma, jl}(\mathbf{u}_h) = \partial_{u_m} u_{\Gamma}^{j}(\mathbf{u}_h) \partial_{x_k} u_h^{m} (\delta_{kl} - 2 n_k n_l).$$

With this notation, the form $\mathscr{N}_{\Gamma_{\text{sym}}}(\cdot,\cdot)$ is defined as follows

$$\begin{aligned} \mathscr{N}_{\Gamma_{\text{sym}}}(\mathbf{u}_h, \mathbf{v}) \;=\; & \int_{\Gamma_{\text{sym}}} \mathscr{H}_{\Gamma}(\mathbf{u}_h^+, \mathbf{u}_{\Gamma}(\mathbf{u}_h^+), \mathbf{n}^+) \cdot \mathbf{v}^+ \, \mathrm{d}s + \int_{\Gamma} \underline{\delta}_{\Gamma_{\text{sym}}}(\mathbf{u}_h^+) : \mathbf{v}^+ \otimes \mathbf{n} \, \mathrm{d}s \\ & - \frac{1}{2} \int_{\Gamma_{\text{sym}}} \left(\mathscr{F}^{v}(\mathbf{u}_h^+, \nabla_h \mathbf{u}_h^+) + \mathscr{F}^{v}(\mathbf{u}_{\Gamma}(\mathbf{u}_h^+), (\nabla \mathbf{u})_{\Gamma}(\mathbf{u}_h^+))\right) : \mathbf{v}^+ \otimes \mathbf{n} \, \mathrm{d}s \\ & - \frac{1}{2} \int_{\Gamma} \left(G^{\top}(\mathbf{u}_h^+) \nabla_h \mathbf{v}_h^+\right) : \left(\mathbf{u}_h^+ - \mathbf{u}_{\Gamma}(\mathbf{u}_h^+)\right) \otimes \mathbf{n} \, \mathrm{d}s, \end{aligned}$$

where

$$\underline{\delta}_{\Gamma_{\text{sym}}}(\mathbf{u}_h) = \frac{1}{2} C_{\text{IP}} \frac{p^2}{\mathrm{h}} \left(G(\mathbf{u}_h^+) + G(\mathbf{u}_{\Gamma}(\mathbf{u}_h^+))\right) \left(\mathbf{u}_h^+ - \mathbf{u}_{\Gamma}(\mathbf{u}_h^+)\right) \otimes \mathbf{n}.$$

4. Newton–GMRES algorithm

To determine the numerical solution $\mathbf{u}_h$ of the system of nonlinear equations (3.1), we employ a damped Newton method. This nonlinear iteration generates a sequence of approximations $\mathbf{u}_h^n$, $n = 0, 1, \ldots$, to the actual numerical solution $\mathbf{u}_h$, using the following algorithm. Given an iterate $\mathbf{u}_h^n$, the update $\mathbf{d}_h^n$ of $\mathbf{u}_h^n$ to get to the next iterate $\mathbf{u}_h^{n+1} = \mathbf{u}_h^n + \omega^n \mathbf{d}_h^n$ is defined by: find $\mathbf{d}_h^n \in V(\mathcal{T}_h, p)$ such that

$$\mathcal{N}'[\mathbf{u}_h^n](\mathbf{d}_h^n, \mathbf{v}_h) = R(\mathbf{u}_h^n, \mathbf{v}_h) \equiv -\mathcal{N}(\mathbf{u}_h^n, \mathbf{v}_h) \quad \forall \mathbf{v}_h \in V(\mathcal{T}_h, p). \tag{4.1}$$

Here, ω^n denotes a damping parameter, which is dynamically chosen to guarantee that the discrete l_2-norm of the residual computed with $\mathbf{u}_h^{n+1}$ is less than the same quantity computed with $\mathbf{u}_h^n$. Additionally, $\mathcal{N}'[\mathbf{w}](\cdot, \mathbf{v})$ denotes (an approximation to) the Fréchet derivative of $\mathbf{u} \rightarrow \mathcal{N}(\mathbf{u}, \mathbf{v})$, for $\mathbf{v} \in V(\mathcal{T}_h, p)$ fixed, at some $\mathbf{w}$ in $\mathbf{V}$, where $\mathbf{V}$ is some suitable chosen function space such that $V(\mathcal{T}_h, p) \in \mathbf{V}$. Here, we define

$$\begin{aligned}
\mathcal{N}'[\mathbf{w}](\boldsymbol{\phi}, \mathbf{v}) = &- \int_\Omega \left(\mathcal{F}^c_{\mathbf{u}}(\mathbf{w}) \boldsymbol{\phi} \right) : \nabla_h \mathbf{v} \, \mathrm{d}\mathbf{x} \\
&+ \sum_{\kappa \in \mathcal{T}_h} \int_{\partial\kappa \setminus \Gamma} \left(\hat{\mathcal{H}}_{\mathbf{u}^+}(\mathbf{w}^+, \mathbf{w}^-, \mathbf{n}_\kappa) \boldsymbol{\phi}^+ + \hat{\mathcal{H}}_{\mathbf{u}^-}(\mathbf{w}^+, \mathbf{w}^-, \mathbf{n}_\kappa) \boldsymbol{\phi}^- \right) \cdot \mathbf{v}^+ \, \mathrm{d}s \\
&+ \int_\Omega \left(\mathcal{F}^v_{\mathbf{u}}(\mathbf{w}, \nabla_h \mathbf{w}) \boldsymbol{\phi} \right) : \nabla_h \mathbf{v} \, \mathrm{d}\mathbf{x} + \int_\Omega \left(\mathcal{F}^v_{\nabla\mathbf{u}}(\mathbf{w}, \nabla_h \mathbf{w}) \nabla_h \boldsymbol{\phi} \right) : \nabla_h \mathbf{v} \, \mathrm{d}\mathbf{x} \\
&- \int_{\Gamma_{\mathcal{I}_h}} \{\!\{ \mathcal{F}^v_{\mathbf{u}}(\mathbf{w}, \nabla_h \mathbf{w}) \boldsymbol{\phi} \}\!\} : \underline{[\![\mathbf{v}]\!]} \, \mathrm{d}s - \int_{\Gamma_{\mathcal{I}_h}} \{\!\{ \mathcal{F}^v_{\nabla\mathbf{u}}(\mathbf{w}, \nabla_h \mathbf{w}) \nabla_h \boldsymbol{\phi} \}\!\} : \underline{[\![\mathbf{v}]\!]} \, \mathrm{d}s \\
&- \int_{\Gamma_{\mathcal{I}_h}} \{\!\{ G^\top_{\mathbf{u}}(\mathbf{w}) \boldsymbol{\phi} \, \nabla_h \mathbf{v} \}\!\} : \underline{[\![\mathbf{w}]\!]}) \, \mathrm{d}s - \int_{\Gamma_{\mathcal{I}_h}} \{\!\{ G^\top(\mathbf{w}) \nabla_h \mathbf{v} \}\!\} : \underline{[\![\boldsymbol{\phi}]\!]}) \, \mathrm{d}s \\
&+ \int_{\Gamma_{\mathcal{I}_h}} \underline{\delta}_{\mathbf{u}}[\mathbf{w}](\boldsymbol{\phi}) : \underline{[\![\mathbf{v}]\!]} \, \mathrm{d}s + \mathcal{N}'_{\Gamma \setminus \Gamma_{\mathrm{sym}}}[\mathbf{w}](\boldsymbol{\phi}, \mathbf{v}) + \mathcal{N}'_{\Gamma_{\mathrm{sym}}}[\mathbf{w}](\boldsymbol{\phi}, \mathbf{v}),
\end{aligned}$$

where $\hat{\mathcal{H}}_{\mathbf{u}^+}(\cdot, \cdot, \mathbf{n}_\kappa)$ and $\hat{\mathcal{H}}_{\mathbf{u}^-}(\cdot, \cdot, \mathbf{n}_\kappa)$ denote approximations to the derivatives of the flux function $\mathcal{H}(\cdot, \cdot, \cdot)$ with respect to its first and second arguments, respectively; for a detailed description of these derivatives for two specific choices of numerical fluxes, we refer to the article [26]. Additionally, we write $\mathcal{F}^c_{\mathbf{u}}(\cdot)$ to denote the derivative of the convective flux function, $\mathcal{F}^v_{\mathbf{u}}(\cdot, \cdot)$ and $\mathcal{F}^v_{\nabla\mathbf{u}}(\cdot, \cdot)$ to denote the derivative of $\mathcal{F}^v(\cdot, \cdot)$ with respect to its first and second arguments, respectively, and $G_{\mathbf{u}}$ to denote the derivative of the tensor G. The derivative of the penalization function $\underline{\delta}$ is given by

$$\underline{\delta}_{\mathbf{u}}[\mathbf{w}](\boldsymbol{\phi}) = C_{\mathrm{IP}} \frac{p^2}{\mathbf{h}} \left[\{\!\{ G_{\mathbf{u}}(\mathbf{w}) \boldsymbol{\phi} \}\!\} \underline{[\![\mathbf{w}]\!]} + \{\!\{ G(\mathbf{w}) \}\!\} \underline{[\![\boldsymbol{\phi}]\!]} \right].$$

Furthermore, $\mathcal{N}'_{\Gamma \setminus \Gamma_{\mathrm{sym}}}[\mathbf{w}](\cdot, \mathbf{v})$ and $\mathcal{N}'_{\Gamma_{\mathrm{sym}}}[\mathbf{w}](\cdot, \mathbf{v})$ denote (approximations to) the Fréchet derivatives of the boundary terms, for $\mathbf{v} \in V(\mathcal{T}_h, p)$ fixed, at some $\mathbf{w}$ in $\mathbf{V}$. In particular,

$$\begin{aligned}
\mathcal{N}'_{\Gamma\setminus\Gamma_{\mathrm{sym}}}[\mathbf{w}](\boldsymbol{\phi},\mathbf{v}) &= \int_{\Gamma\setminus\Gamma_{\mathrm{sym}}} \mathbf{n}\cdot\mathcal{F}^c_{\mathbf{u}}(\mathbf{u}_\Gamma(\mathbf{w}^+))\mathbf{u}'_\Gamma(\mathbf{w}^+)\boldsymbol{\phi}^+\cdot\mathbf{v}^+\,\mathrm{d}s \\
&+ \int_{\Gamma\setminus\Gamma_{\mathrm{sym}}} C_{\mathrm{IP}}\frac{p^2}{\mathrm{h}}\Big[G_{\mathbf{u}}(\mathbf{u}_\Gamma(\mathbf{w}^+))\boldsymbol{\phi}^+(\mathbf{w}^+-\mathbf{u}_\Gamma(\mathbf{w}^+))\otimes\mathbf{n} \\
&\qquad + G(\mathbf{u}_\Gamma(\mathbf{w}^+))(\boldsymbol{\phi}^+-\mathbf{u}'_\Gamma(\mathbf{w}^+)\boldsymbol{\phi}^+)\otimes\mathbf{n}\Big] : \mathbf{v}^+\otimes\mathbf{n}\,\mathrm{d}s, \\
&- \int_{\Gamma\setminus\Gamma_{\mathrm{sym}}} \Big(\mathcal{F}^v_{\mathbf{u}}(\mathbf{u}_\Gamma(\mathbf{w}^+),\nabla_h\mathbf{w}^+)\mathbf{u}'_\Gamma(\mathbf{w}^+)\boldsymbol{\phi}^+ + \mathcal{F}^v_{\nabla\mathbf{u}}(\mathbf{u}_\Gamma(\mathbf{w}^+),\nabla_h\mathbf{w}^+)\nabla_h\boldsymbol{\phi}^+\Big) : \mathbf{v}^+\otimes\mathbf{n}\,\mathrm{d}s \\
&- \int_{\Gamma\setminus\Gamma_{\mathrm{sym}}} \Big(G^\top_{\mathbf{u}}(\mathbf{u}_\Gamma(\mathbf{w}^+))\mathbf{u}'_\Gamma(\mathbf{w}^+)\boldsymbol{\phi}^+\nabla_h\mathbf{v}^+\Big) : \Big(\mathbf{w}^+-\mathbf{u}_\Gamma(\mathbf{w}^+)\Big)\otimes\mathbf{n}\,\mathrm{d}s \\
&- \int_{\Gamma\setminus\Gamma_{\mathrm{sym}}} \Big(G^\top_\Gamma(\mathbf{w}^+)\nabla_h\mathbf{v}^+\Big) : \Big(\boldsymbol{\phi}^+-\mathbf{u}'_\Gamma(\mathbf{w}^+)\boldsymbol{\phi}^+\Big)\otimes\mathbf{n}\,\mathrm{d}s.
\end{aligned}$$

Similarly,

$$\begin{aligned}
\mathcal{N}'_{\Gamma_{\mathrm{sym}}}[\mathbf{w}](\boldsymbol{\phi},\mathbf{v}) &= \int_{\Gamma_{\mathrm{sym}}} \mathbf{n}\cdot\mathcal{F}^c_{\mathbf{u}}(\mathbf{u}_\Gamma(\mathbf{w}^+))\mathbf{u}'_\Gamma(\mathbf{w}^+)\boldsymbol{\phi}^+\cdot\mathbf{v}^+\,\mathrm{d}s \\
&+ \frac{1}{2}\int_{\Gamma_{\mathrm{sym}}} C_{\mathrm{IP}}\frac{p^2}{\mathrm{h}}\Big[\Big(G_{\mathbf{u}}(\mathbf{w}^+)+G_{\mathbf{u}}(\mathbf{u}_\Gamma(\mathbf{w}^+))\mathbf{u}'_\Gamma(\mathbf{w}^+)\Big)\,\boldsymbol{\phi}^+(\mathbf{w}^+-\mathbf{u}_\Gamma(\mathbf{w}^+))\otimes\mathbf{n} \\
&\qquad + \Big(G(\mathbf{w}^+)+G(\mathbf{u}_\Gamma(\mathbf{w}^+))\Big)\,(\boldsymbol{\phi}^+-\mathbf{u}'_\Gamma(\mathbf{w}^+)\boldsymbol{\phi}^+)\otimes\mathbf{n}\Big] : \mathbf{v}^+\otimes\mathbf{n}\,\mathrm{d}s \\
&- \frac{1}{2}\int_{\Gamma_{\mathrm{sym}}} \Big(\mathcal{F}^v_{\mathbf{u}}(\mathbf{w}^+,\nabla_h\mathbf{w}^+)\boldsymbol{\phi}^+ + \mathcal{F}^v_{\nabla\mathbf{u}}(\mathbf{w}^+,\nabla_h\mathbf{w}^+)\nabla_h\boldsymbol{\phi}^+\Big) : \mathbf{v}^+\otimes\mathbf{n}\,\mathrm{d}s \\
&- \frac{1}{2}\int_{\Gamma_{\mathrm{sym}}} \Big(\mathcal{F}^v_{\mathbf{u}}(\mathbf{u}_\Gamma(\mathbf{w}^+),(\nabla\mathbf{u})_\Gamma(\mathbf{w}^+))\mathbf{u}'_\Gamma(\mathbf{w}^+)\boldsymbol{\phi}^+ \\
&\qquad + \mathcal{F}^v_{\nabla\mathbf{u}}(\mathbf{u}_\Gamma(\mathbf{w}^+),(\nabla\mathbf{u})_\Gamma(\mathbf{w}^+))(\nabla\mathbf{u})'_\Gamma(\mathbf{w}^+)\nabla_h\boldsymbol{\phi}^+\Big) : \mathbf{v}^+\otimes\mathbf{n}\,\mathrm{d}s \\
&- \frac{1}{2}\int_{\Gamma_{\mathrm{sym}}} \Big(G^\top_{\mathbf{u}}(\mathbf{w}^+)\boldsymbol{\phi}^+\nabla_h\mathbf{v}^+\Big) : \Big(\mathbf{w}^+-\mathbf{u}_\Gamma(\mathbf{w}^+)\Big)\otimes\mathbf{n}\,\mathrm{d}s \\
&- \frac{1}{2}\int_{\Gamma_{\mathrm{sym}}} \Big(G^\top(\mathbf{w}^+)\nabla_h\mathbf{v}^+\Big) : \Big(\boldsymbol{\phi}^+-\mathbf{u}'_\Gamma(\mathbf{w}^+)\boldsymbol{\phi}^+\Big)\otimes\mathbf{n}\,\mathrm{d}s.
\end{aligned}$$

Here, $\mathbf{u}'_\Gamma(\mathbf{u})$ denotes the derivative of the boundary function $\mathbf{u}_\Gamma(\mathbf{u})$ with respect to the conservative variables $\mathbf{u}$. On the supersonic parts of the boundary, we have $\mathbf{u}'_\Gamma(\mathbf{u}) = 0$ on

$\Gamma_{\text{D,sup}}$; on $\Gamma_{\text{D,sub-in}}$ and $\Gamma_{\text{D,sub-out}}$, $\mathbf{u}'_\Gamma(\mathbf{u})$ is given by

$$\mathbf{u}'_\Gamma(\mathbf{u}) = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ \frac{1}{2}|\mathbf{v}|^2 & -v_1 & -v_2 & -v_3 & 1 \end{pmatrix} \quad \text{and} \quad \mathbf{u}'_\Gamma(\mathbf{u}) = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ -\frac{1}{2}|\mathbf{v}|^2 & v_1 & v_2 & v_3 & 0 \end{pmatrix},$$

respectively. On the isothermal and adiabatic no-slip boundaries,

$$\mathbf{u}'_\Gamma(\mathbf{u}) = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ c_v T_{\text{wall}} & 0 & 0 & 0 & 0 \end{pmatrix} \quad \text{and} \quad \mathbf{u}'_\Gamma(\mathbf{u}) = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix},$$

respectively. On Γ_{sym}, $\mathbf{u}'_\Gamma(\mathbf{u}) = B_{\text{sym}}$, cf. (3.2). Finally, $(\nabla\mathbf{u})'_\Gamma$ denotes the derivative of the (boundary) gradient operator $(\nabla\mathbf{u})_\Gamma$.

The linear system of equations defined by (4.1) potentially may be very large, particularly for three–dimensional problems. In this setting, direct methods may not be appropriate due to the large memory requirements needed to store the corresponding L and U factors, for example. With this mind, in Section 6, we consider the construction of two-level Schwarz–type preconditioners for application within the GMRES iterative method. First, however, in the following section, we introduce the so–called composite DGFEM (DGCFEM), which is based on employing arbitrarily shaped (agglomerated) elements; this scheme will then form the basis of the coarse grid solver defined in Section 6.

5. Construction of composite DGFEMs

In this section we briefly introduce the composite version of the (interior penalty) DGFEM; for further details, we refer to our recent article [6]. For the purposes of defining a coarse level solver for application within a Schwarz–type preconditioner, we consider a simpler construction to that presented in [6], following the ideas developed in the article [11], cf. [7].

Given the (fine) mesh $\mathcal{T}_h$, we consider a coarsened mesh $\mathcal{T}_H$ which is constructed based on agglomerating elements from $\mathcal{T}_h$. Thereby, $\mathcal{T}_H$ represents a partition of Ω into disjoint elements κ_H, such that (i) $\overline{\Omega} = \cup_{\kappa_H \in \mathcal{T}_H} \overline{\kappa}_H$; (ii) we may write

$$\overline{\kappa}_H = \cup_{\kappa \in \mathcal{R}_{\kappa_H}} \overline{\kappa},$$

where $\mathcal{R}_{\kappa_H}$ denotes the set of elements from the fine mesh $\mathcal{T}_h$, which are employed to construct κ_H. For the purposes of this article, we assume for simplicity, that the elements κ_H are connected; we remark that this restriction may be relaxed, cf. [6]. We denote by $\Gamma_{\mathcal{I}_H}$ the union of all interior faces of the partition $\mathcal{T}_H$ of Ω.

With this notation, we make the following key assumption:

(A1) For all elements $\kappa \in \mathscr{T}_H$, we define

$$C_\kappa = \mathrm{card}\left\{F \subset \Gamma \cup \Gamma_{\mathscr{I}_H} : F \subset \partial\kappa\right\}.$$

In the following we assume that there exists a positive constant C_F such that

$$\max_{\kappa \in \mathscr{T}_H} C_\kappa \leq C_F,$$

uniformly with respect to the mesh size.

To each composite/agglomerated element $\kappa_H \in \mathscr{T}_H$, we assign the polynomial degree $0 \leq q \leq p$. The construction of the composite finite element space $V(\mathscr{T}_H, q)$, cf. below, may be undertaken based on employing a suitable prolongation operator $R_0^\top$, cf. [6, 24]. We point out that the choice of $R_0^\top$ employed in [24] leads to finite element basis functions, defined on each composite element domain κ_H, which are piecewise polynomials. In contrast, we follow the approach developed in [6], whereby the restriction of a function from the underlying finite element space to an element $\kappa_H \in \mathscr{T}_H$ is a polynomial of degree q. Thereby, we write

$$V(\mathscr{T}_H, q) = \{\mathbf{v} \in [L_2(\Omega)]^{d+2} : \mathbf{v}|_{\kappa_H} \in [\mathscr{P}_q(\kappa_H)]^{d+2} \quad \forall \kappa_H \in \mathscr{T}_H\},$$

cf., also, [11]. Following [6], the classical prolongation (injection) operator from $V(\mathscr{T}_H, q)$ to $V(\mathscr{T}_h, p)$ is denoted by $R_0^\top : V(\mathscr{T}_H, q) \to V(\mathscr{T}_h, p)$. With this notation, we may write $V(\mathscr{T}_H, q)$ in the following alternative form

$$V(\mathscr{T}_H, q) = \{\mathbf{v} \in [L_2(\Omega)]^{d+2} : \mathbf{v} = R_0 \boldsymbol{\phi},\ \boldsymbol{\phi} \in V(\mathscr{T}_h, p)\}, \tag{5.1}$$

where the restriction operator R_0 is defined as the transpose of $R_0^\top$ with respect to the $L_2(\Omega)$ inner product.

The DGCFEM discretization of the problem (2.1) is defined as follows: find $\mathbf{u}_H \in V(\mathscr{T}_H, q)$ such that

$$\mathscr{N}_{\mathtt{CDG}}(\mathbf{u}_H, \mathbf{v}) = 0 \tag{5.2}$$

for all $\mathbf{v} \in V(\mathscr{T}_H, q)$, where $\mathscr{N}_{\mathtt{CDG}}(\cdot, \cdot)$ is defined in an analogous manner to $\mathscr{N}(\cdot, \cdot)$, relative to a penalization function $\underline{\delta}_H(\cdot)$. We remark that $\underline{\delta}_H(\cdot)$ is defined in a similar fashion to $\underline{\delta}(\cdot)$, subject to a change in the definition of the *representative* face length $\mathbb{h}$ employed and of the, possibly different, polynomial approximation degree employed on the coarse grid, cf. [6] for details.

6. Non-overlapping Schwarz preconditioners

In this section we introduce two level non-overlapping Schwarz preconditioners in order to compute the Newton solver update $\mathbf{d}_h^n$, $n = 0, 1, 2, \ldots$, defined on the fine space $V(\mathscr{T}_h, p)$ given by (4.1). To this end, we denote by $\mathscr{T}_{\mathscr{S}} = \{\Omega_i\}_{i=1}^N$ a family of partitions of Ω into N non-overlapping domains, such that $\overline{\Omega} = \cup_{i=1}^N \overline{\Omega}_i$. With this (user–defined) partition, we consider two families of fine and coarse meshes $\mathscr{T}_h$ and $\mathscr{T}_H$, respectively, constructed as in the previous sections, respectively. In particular, we assume that $\mathscr{T}_h$, $\mathscr{T}_H$ and $\mathscr{T}_{\mathscr{S}}$ are nested, $\mathscr{T}_{\mathscr{S}} \subseteq \mathscr{T}_H \subseteq \mathscr{T}_h$, i.e., the subdomain partition does not cut any element of $\mathscr{T}_H$ and thereby of $\mathscr{T}_h$.

With this notation, we now introduce the local and coarse level solvers.

Local solvers For $i = 1, \ldots, N$, the local DGFEM finite element spaces are defined on Ω_i, respectively, in the following manner:

$$V(\mathscr{T}_{h_i}, p) = \{\mathbf{v} \in [L_2(\Omega_i)]^{d+2} : \mathbf{v}|_\kappa \in [\mathscr{S}_p(\kappa)]^{d+2} \quad \forall \kappa \in \mathscr{T}_{h_i}\},$$

where $\mathscr{T}_{h_i} = \{\kappa \in \mathscr{T}_h : \kappa \subset \Omega_i\}$. The classical prolongation (injection) operator from $V(\mathscr{T}_{h_i}, p)$ to $V(\mathscr{T}_h, p)$ is denoted by $R_i^\top : V(\mathscr{T}_{h_i}, p) \to V(\mathscr{T}_h, p)$. The restriction operator R_i is defined as the transpose of $R_i^\top$ with respect to the $L_2(\Omega_i)$ inner product, cf. above. In particular, we note that

$$V(\mathscr{T}_h, p) = R_1^\top V(\mathscr{T}_{h_1}, p) \oplus \ldots \oplus R_N^\top V(\mathscr{T}_{h_N}, p).$$

The local solvers $\mathscr{N}_i'[\mathbf{u}_h^n](\cdot,\cdot) : V(\mathscr{T}_{h_i}, p) \times V(\mathscr{T}_{h_i}, p) \to \mathbb{R}$ are defined as follows:

$$\mathscr{N}_i'[\mathbf{u}_h^n](\mathbf{u}_i, \mathbf{v}_i) := \mathscr{N}'[\mathbf{u}_h^n](R_i^\top \mathbf{u}_i, R_i^\top \mathbf{v}_i) \qquad \forall \mathbf{u}_i, \mathbf{v}_i \in V(\mathscr{T}_{h_i}, p), \; i = 1, \ldots, N.$$

Coarse solver Employing the composite discontinuous Galerkin finite element space $V(\mathscr{T}_H, q) \subset V(\mathscr{T}_h, p)$, the coarse solver $\mathscr{N}_0'[\mathbf{u}_h^n](\cdot,\cdot) : V(\mathscr{T}_H, q) \times V(\mathscr{T}_H, q) \to \mathbb{R}$ is defined by

$$\mathscr{N}_0'[\mathbf{u}_h^n](\mathbf{u}_0, \mathbf{v}_0) := \mathscr{N}_{\mathtt{CDG}}'[\mathbf{u}_h^n](\mathbf{u}_0, \mathbf{v}_0) \qquad \forall \mathbf{u}_0, \mathbf{v}_0 \in V(\mathscr{T}_H, q),$$

where $\mathscr{N}_{\mathtt{CDG}}'[\mathbf{w}](\cdot, \mathbf{v})$ denotes (an approximation to) the Fréchet derivative of $\mathbf{u} \to \mathscr{N}_{\mathtt{CDG}}(\mathbf{u}, \mathbf{v})$, for $\mathbf{v} \in V(\mathscr{T}_h, p)$ fixed, at some $\mathbf{w}$.

Local projection operators For $i = 1, \ldots, N$, the local projection operators $\tilde{P}_i : V(\mathscr{T}_h, p) \to V(\mathscr{T}_{h_i}, p)$ are defined by:

$$\mathscr{N}_i'[\mathbf{u}_h^n](\tilde{P}_i \mathbf{u}, \mathbf{v}_i) := \mathscr{N}'[\mathbf{u}_h^n](\mathbf{u}, R_i^\top \mathbf{v}_i) \quad \forall \mathbf{v}_i \in V(\mathscr{T}_{h_i}, p).$$

Analogously, on the coarse space $V(\mathscr{T}_H, q)$, we let $\tilde{P}_0 : V(\mathscr{T}_h, p) \to V(\mathscr{T}_H, q)$ be given by

$$\mathscr{N}_0'[\mathbf{u}_h^n](\tilde{P}_0 \mathbf{u}, \mathbf{v}_0) := \mathscr{N}'[\mathbf{u}_h^n](\mathbf{u}, R_0^\top \mathbf{v}_0) \quad \forall \mathbf{v}_0 \in V(\mathscr{T}_H, q).$$

With this notation, we define the projection operators

$$P_i := R_i^\top \tilde{P}_i : V(\mathscr{T}_h, p) \to V(\mathscr{T}_h, p),$$

for $i = 0, 1, \ldots, N$.

Thereby, the additive and multiplicative Schwarz operators are defined, respectively, by

$$P_{\mathtt{ad}} := \sum_{i=0}^{N} P_i, \quad P_{\mathtt{mu}} := I - (I - P_N)(I - P_{N-1}) \cdots (I - P_0).$$

The convergence analysis for the proposed preconditioner for simple elliptic PDEs has been undertaken in our recent article [7] based on the ideas developed in [8].

7. Implementation issues

In this section we briefly outline some of the implementation aspects of the above multi-level preconditioners; for further details, and in particular details concerning the construction of the projection operators P_i, $i = 0, 1, \dots, N$, we refer to [7]. Moreover, the implementation of the coarse grid (composite) DGCFEM is discussed in detail in [6].

Generation of the coarse level mesh In the case when the coarse level mesh $\mathscr{T}_H$ is not provided by the user, we instead construct $\mathscr{T}_H$ from the fine level mesh $\mathscr{T}_h$ based on an agglomeration algorithm. More precisely, we first create a partition of the fine level mesh $\mathscr{T}_h$ into N_H regions or macro-elements based on exploiting the graph partitioning package METIS [32]. In order for METIS to partition the mesh $\mathscr{T}_h$, the logical structure of the mesh is first stored in a graph, where each node represents an element domain of $\mathscr{T}_h$, and each link between two nodes represents a face shared by the two elements represented by the graph nodes. The partition of $\mathscr{T}_h$ constructed by METIS is produced with the objective of minimizing the number of neighbours among each of the resulting partitions. Once the partition of $\mathscr{T}_h$ into N_H regions has been computed, the coarse mesh $\mathscr{T}_H$ may be easily constructed; in fact, here we simply identify each subpartition region to be a composite/agglomerated element $\kappa_H \in \mathscr{T}_H$. Thereby, $\mathscr{T}_H$ is formed by N_H elements which are arbitrarily shaped polygons resulting from the aggregation of fine level elements.

Construction of the finite element basis functions in the DGCFEM space $V(\mathscr{T}_H, q)$ The composite finite element space $V(\mathscr{T}_H, q)$ may be constructed in a number of different ways. In the case when the coarse level mesh is $\mathscr{T}_H$ is provided by the user, we construct the basis for the composite finite element space based on directly exploiting the prolongation operator $R_0^\top$ employed in the definition of $V(\mathscr{T}_H, q)$, cf. (5.1); for details, we refer to the algorithm outlined in [6]. An alternative approach is to construct a basis for $V(\mathscr{T}_H, q)$ by working directly on the agglomerated elements $\kappa_H \in \mathscr{T}_H$. Indeed, the recent article [11] proposed an algorithm to construct the elemental basis functions based on employing a Gram-Schmidt orthogonalization process applied to a given set of polynomial functions. In this article, we propose an alternative approach based on employing polynomial spaces defined over the bounding box of each element, cf. [17]. More precisely, given an element $\kappa_H \in \mathscr{T}_H$, we first construct the Cartesian bounding box B_{κ_H}, such that $\bar{\kappa}_H \subseteq \bar{B}_{\kappa_H}$. On the bounding box B_{κ_H} we may define a standard polynomial space $\mathscr{S}_q(B_{\kappa_H})$ spanned by a set of basis functions $\{b_{i,\kappa_H}\}$, $i = 1, \dots, \dim(\mathscr{S}_q(B_{\kappa_H}))$. Finally, a polynomial basis over the composite/agglomerated element κ_H may be defined by simply restricting the support of $\{b_{i,\kappa_H}\}$, $i = 1, \dots, \dim(\mathscr{S}_q(B_{\kappa_H}))$, to κ_H; i.e., the polynomial basis defined over κ_H is given by $\{b_{i,\kappa_H}|_{\kappa_H}\}$, $i = 1, \dots, \dim(\mathscr{S}_q(B_{\kappa_H}))$.

8. Numerical experiments

In this section we present a series of computational examples to highlight the practical performance of the non-overlapping Schwarz preconditioners proposed in this article.

$\sqrt{2}h^{-1}$	2	4	8	16	32
GMRES	122 (812, 9)	300 (1759, 8)	660 (3119, 6)	1850 (8730, 6)	* (*,*)
GMRES+ILU	12 (80,9)	20 (115,8)	33 (174,6)	89 (449,6)	292 (1373,6)

Table 1: Example 1: GMRES and GMRES+ILU iteration counts, when $p = 1$.

$\sqrt{2}h^{-1}$	2	4	8	16	32
GMRES	562 (3140, 7)	921 (5182, 7)	1791 (8780, 6)	3150 (15620, 6)	7977 (24154,6)
GMRES+ILU	19 (113,7)	27 (153,7)	38 (200,6)	75 (365,6)	201 (930,5)

Table 2: Example 1: GMRES and GMRES+ILU iteration counts, when $p = 2$.

For simplicity, we restrict ourselves to the additive Schwarz operator $P_{\mathtt{ad}}$. All the numerical examples presented in this section have been computed using the AptoFEM package (`www.aptofem.com`).

8.1. Example 1: Two–dimensional viscous flow in a square domain

In this first example, we consider a simple model problem in order to study the performance of the proposed preconditioner on sequences of uniform meshes. To this end, we let Ω be the square $(-1,1)^2$, and supplement the compressible Navier–Stokes equations (2.1) with an inhomogeneous forcing function $\mathbf{f}$, which is chosen so that the analytical solution to (2.1) is given by

$$\rho := \sin(2(x+y))+4\,, \quad \rho v_1 = \rho v_2 := 0.2\sin(2(x+y))+4\,, \quad \rho E := (\sin(2(x+y))+4)^2\,,$$

where the dynamic viscosity coefficient μ is set to $1/10$; cf. [30].

In all of the simulations presented in this section, we partition the fine level mesh $\mathcal{T}_h$ into four subdomains, i.e., so that $N = 4$. Furthermore, we set a tolerance for the Newton–GMRES algorithm outlined in Section 4 equal to 10^{-8}, as the termination condition, and the absolute tolerance for the underlying GMRES inner–solver to 10^{-8} with a restart every 50 iterations. In the following tables, we report the maximum number of GMRES iterations required to compute the solution to each inner (linear) problem within the Newton–GMRES algorithm and, in brackets, the total number of all GMRES iterations required by the Newton–GMRES method to attain convergence and the number of Newton iterations. Throughout this section we employ uniform meshes consisting of quadrilateral elements for both the fine and coarse level meshes $\mathcal{T}_h$ and $\mathcal{T}_H$, respectively, with granularity h and H, respectively.

For reference, in Tables 1 & 2 we first show the number of iterations required for convergence with polynomial orders $p = 1$ and $p = 2$, respectively, when no preconditioner is applied within the Newton–GMRES algorithm and when GMRES with an ILU preconditioner is employed based on using the PETSc software package [9]. Here, we observe that for each polynomial degree, the number of iterations increases as the mesh is uniformly refined at each step; we point out that for $h = \sqrt{2}/32$ and $p = 1$, the Newton–GMRES

$\sqrt{2}h^{-1}$ \ $\sqrt{2}H^{-1}$	1	2	4	8	16
2	21 (150,9)	-	-	-	-
4	27 (174,8)	26 (164,8)	-	-	-
8	39 (206,6)	38 (199,6)	33 (169,6)	-	-
16	75 (353,6)	66 (311,6)	50 (251,6)	37 (187,6)	-
32	96 (477,6)	96 (472,6)	83 (406,6)	60 (291,6)	34 (167,6)

Table 3: Example 1: GMRES iteration counts with the preconditioner, when $p = q = 1$.

$\sqrt{2}h^{-1}$ \ $\sqrt{2}H^{-1}$	1	2	4	8	16
2	37 (221,7)	-	-	-	-
4	68 (351,7)	46 (260,7)	-	-	-
8	122 (572,6)	75 (363,6)	46 (226,6)	-	-
16	128 (606,6)	100 (476,6)	60 (278,6)	41 (192,6)	-
32	140 (562,5)	135 (511,5)	67 (270,5)	39 (159,5)	32 (136,5)

Table 4: Example 1: GMRES iteration counts with the preconditioner, when $p = q = 2$.

$\sqrt{2}h^{-1}$ \ $\sqrt{2}H^{-1}$	1	2	4	8	16
2	39 (230,7)	-	-	-	-
4	76 (414,7)	56 (301,7)	-	-	-
8	99 (501,6)	182 (856,6)	66 (320,6)	-	-
16	127 (567,6)	141 (647,6)	90 (431,6)	55 (255,6)	-
32	94 (373,5)	142 (577,5)	126 (529,5)	50 (204,5)	35 (146,5)

Table 5: Example 1: GMRES iteration counts with the preconditioner, when $p = 2$ and $q = 1$.

algorithm failed to converge, in the sense that the maximum number of GMRES iterations (set to 10000) was exceeded within one of the (linear) inner-solves. We note that for finer meshes the ILU preconditioner actually leads to a smaller number of iterations required to attain convergence when the polynomial degree is increased from 1 to 2.

In Tables 3, 4, & 5 we now consider the performance of the proposed additive Schwarz preconditioner when $p = q = 1$, $p = q = 2$, and $p = 2$, $q = 1$, respectively. Here we observe that the maximum number of GMRES iterations remains roughly constant when the ratio of the coarse and fine mesh sizes is kept fixed. Moreover, we note that, overall, the number of iterations required for convergence when $p = q = 2$ is typically smaller than when $p = 2$ and $q = 1$; indeed, although a richer coarse space $V(\mathscr{T}_H, q)$ leads to a more computationally expensive preconditioner, it generally leads to a reduction in iteration

# Eles $\mathcal{T}_H$ / Mesh $\mathcal{T}_h$	500	1000	2000	4000	8000
Mesh 2	124 (936,10)	-	-	-	-
Mesh 3	186 (1303,9)	121 (800,9)	-	-	-
Mesh 4	310 (1957,9)	168 (1150,9)	116 (700,9)	-	-
Mesh 5	519 (3136,9)	278 (1796,9)	151 (1034,9)	95 (646,9)	-
Mesh 6	933 (5604,9)	492 (3034,9)	276 (1785,9)	162 (1090,9)	103 (687,9)

Table 6: Example 2: GMRES iteration counts for the preconditioner, with $p = q = 1$ on unstructured meshes using coarse meshes created with METIS and a subdomain partition generated with METIS consisting of $N = 250$ domains.

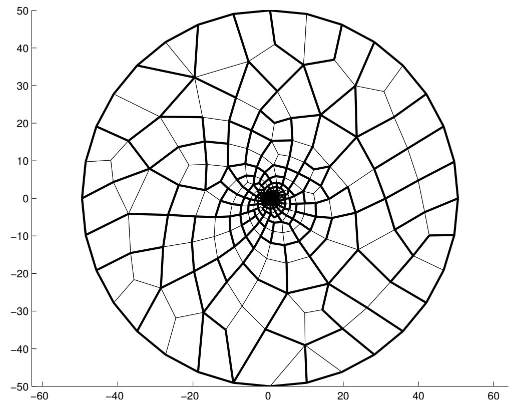
counts. The behaviour observed in this numerical experiment is in agreement with the analysis and numerical experiments presented for the Poisson equation in the articles [4, 7, 8], for example. Comparing these results with those attained with the standard ILU preconditioner, cf. Tables 1 & 2, we note that on very coarse meshes the ILU preconditioner requires less iterations than the proposed additive Schwarz preconditioner. However, as the mesh is refined, the scalability of the latter preconditioner naturally leads to a significant reduction in the number of iterations needed to attain convergence when compared to the ILU approach.

8.2. Example 2: Laminar flow around a NACA0012

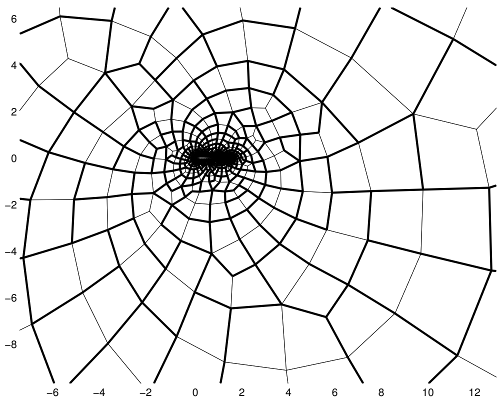
In this second example, we consider the subsonic viscous flow around a NACA0012 airfoil. At the farfield (inflow) boundary we specify a Mach 0.5 flow at an angle of attack $\alpha = 2^\circ$, with Reynolds number $\mathrm{Re} = 5000$; on the walls of the airfoil geometry, we impose a zero heat flux (adiabatic) no-slip boundary condition. As in the previous section we set a tolerance for termination of the Newton–GMRES algorithm equal to 10^{-8}, and the absolute tolerance for the underlying GMRES inner–solver to 10^{-8}; though here, we restart every 500 iterations.

In this example, we consider the performance of the proposed preconditioner on a sequence of unstructured quadrilateral–dominant hybrid meshes consisting of both quadrilateral and triangular elements. The coarsest mesh, denoted as mesh 1, consists of 578 elements; the subsequent meshes, meshes 2–6, each contain 1134, 2113, 4246, 8946, and 20229 elements, respectively. Here, we shall investigate the dependence of the proposed additive Schwarz preconditioner with respect to different subdomain partition strategies. From Section 6, we recall that the construction of the preconditioner requires that the computational domain is first partitioned into N non-overlapping subdomains; this partition is denoted by $\mathcal{T}_{\mathcal{S}}$. The coarse and fine mesh partitions $\mathcal{T}_H$ and $\mathcal{T}_h$, respectively, are then constructed in such a manner that $\mathcal{T}_h$, $\mathcal{T}_H$ and $\mathcal{T}_{\mathcal{S}}$ are nested, i.e., $\mathcal{T}_{\mathcal{S}} \subseteq \mathcal{T}_H \subseteq \mathcal{T}_h$.

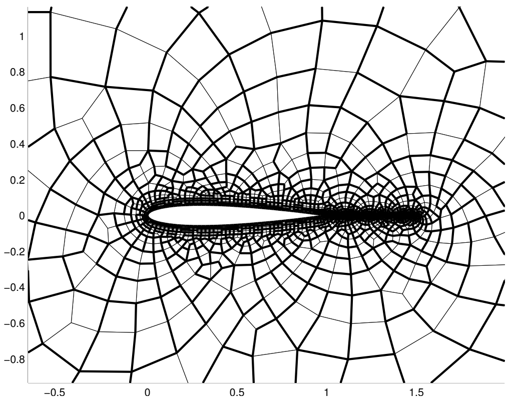
Firstly, we consider the performance of the proposed multilevel preconditioner based on exploiting METIS for the construction of both the subdomain partition $\mathcal{T}_{\mathcal{S}}$ and the composite elements in the coarse mesh $\mathcal{T}_H$. We stress that both $\mathcal{T}_{\mathcal{S}}$ and $\mathcal{T}_H$ are constructed in such a manner that the inclusion $\mathcal{T}_{\mathcal{S}} \subseteq \mathcal{T}_H \subseteq \mathcal{T}_h$ holds. To this end, the number of



(a)

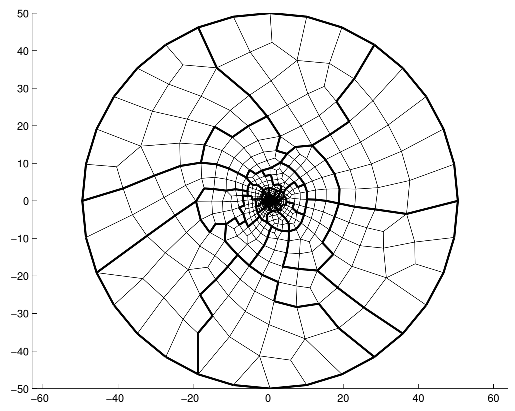


(b)

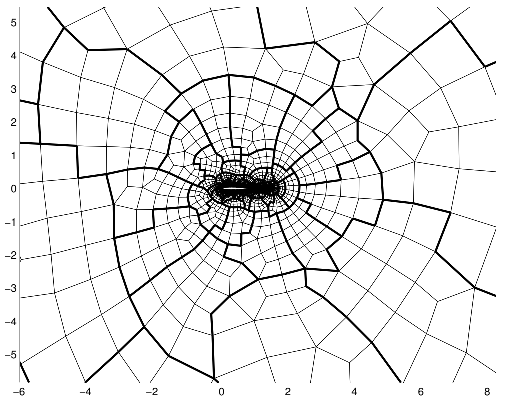


(c)

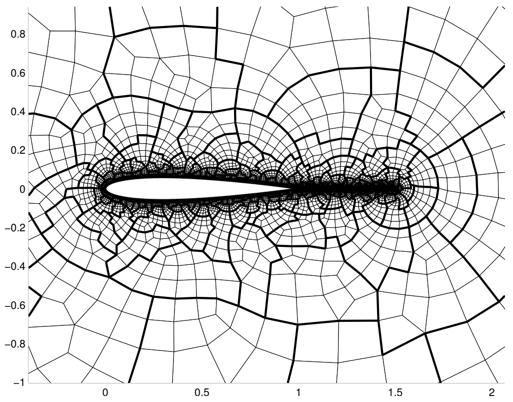
Figure 1: Example 2: (a) Mesh 3 partitioned into 1000 regions using METIS; (b) Zoom of (a); (c) Further zoom of (a).



(a)



(b)



(c)

Figure 2: Example 2: (a) Mesh 5 partitioned into 500 regions using METIS; (b) Zoom of (a); (c) Further zoom of (a)

# Eles $\mathcal{T}_H$ / Mesh $\mathcal{T}_h$	500	1000	2000	4000	8000
Mesh 2	139 (1043,10)	-	-	-	-
Mesh 3	208 (1434,9)	175 (1037,9)	-	-	-
Mesh 4	336 (2238,9)	228 (1519,9)	203 (1075,9)	-	-
Mesh 5	629 (3644,9)	343 (2250,9)	283 (1589,9)	172 (1004,9)	-
Mesh 6	1150 (6827,9)	665 (4150,9)	435 (2831,9)	342 (2046,9)	342 (1783,9)

Table 7: Example 2: GMRES iteration counts for the preconditioner, with $p = q = 1$ on unstructured meshes using coarse meshes created with METIS *without* a subdomain partition.

iterations required to attain convergence of the Newton–GMRES algorithm for $p = q = 1$ are presented in Table 6. Here, as in the previous example, we again observe that the maximum number of GMRES iterations required to attain convergence remains roughly constant when the ratio of the coarse and fine mesh sizes is kept fixed; we remark that while the number of elements in the coarse mesh (denoted by # Eles $\mathcal{T}_H$ in Table 6) is doubled at each step, the number of elements contained within each fine mesh $\mathcal{T}_h$ is only approximately doubled for the set of meshes studied here. For reference, in Figures 1 & 2 we show meshes 3 and 5, respectively, together with the corresponding coarse mesh partition $\mathcal{T}_H$ (denoted with bold lines) generated using METIS. In particular, we observe that the resulting composite elements employed within the coarse mesh solver may contain quite complex element shapes, which typically are not even convex. Indeed, the composite finite element method outlined in Section 5, cf. [6], can naturally handle such element domains. The practical performance of this class of methods on a range of element domains which includes general simply–connected polygons, and even elements which consist of disconnected subdomains is illustrated in [6].

We now turn our attention to study the effect of the choice of the subdomain partition and coarse solver. To this end, in Table 7 we present the number of iterations required to attain convergence of the Newton–GMRES algorithm when METIS is employed to construct the coarse mesh $\mathcal{T}_H$; however, here we do not employ a fixed subdomain partition of the computational domain Ω. Instead, we solve local fine mesh problems on each of the coarse mesh elements present in $\mathcal{T}_H$; more precisely, we set $\mathcal{T}_{\mathcal{S}} = \mathcal{T}_H$, so that N, the number of subdomain partitions present in $\mathcal{T}_{\mathcal{S}}$, grows as $\mathcal{T}_H$ is refined. In essence, in this setting the preconditioner corresponds to a block-Jacobi preconditioner with coarse mesh correction. As expected, in Table 7 we now observe that the number of iterations required to attain convergence of the Newton–GMRES algorithm is not only greater than the corresponding quantity when N is kept fixed, cf. Table 6, but also that the iteration counts (in general) increase as the fine mesh is enriched, even when the ratio of the coarse and fine mesh sizes is kept fixed.

Finally, in this section we consider employing non-nested meshes for the coarse grid solver; thereby, $\mathcal{T}_H \not\subseteq \mathcal{T}_h$. To this end, in Table 8 we present the iteration counts for the case when METIS is employed to generate a subdomain partition consisting of $N = 250$ domains, but the original unstructured meshes are exploited for the coarse solver.

Mesh $\mathcal{T}_h$ \ Mesh $\mathcal{T}_H$	Mesh 2	Mesh 3	Mesh 4	Mesh 5
Mesh 3	101 (654,9)	-	-	-
Mesh 4	142 (1300,9)	100 (614,9)	-	-
Mesh 5	245 (1614,9)	129 (882,9)	90 (565,9)	-
Mesh 6	106 (1698,9)	235 (1547,9)	149 (936,9)	81 (531,9)

Table 8: Example 2: GMRES iteration counts for the preconditioner, with $p = q = 1$ on unstructured meshes using a subdomain partition generated with METIS consisting of $N = 250$ domains.

Mesh $\mathcal{T}_h$ \ Mesh $\mathcal{T}_H$	Mesh 2	Mesh 3	Mesh 4	Mesh 5
Mesh 3	118 (1038,9)	-	-	-
Mesh 4	169 (1067,9)	94 (631,9)	-	-
Mesh 5	256 (1690,9)	140 (943,9)	87 (549,9)	-
Mesh 6	432 (2970,9)	245 (1610,9)	151 (902,9)	105 (576,9)

Table 9: Example 2: GMRES iteration counts for the preconditioner, with $p = q = 1$ on unstructured meshes using Mesh 1 as a subdomain partition.

Mesh $\mathcal{T}_h$ \ Mesh $\mathcal{T}_H$	Mesh 1	Mesh 2	Mesh 3	Mesh 4	Mesh 5
Mesh 2	137 (1012,10)	-	-	-	-
Mesh 3	199 (1340,9)	153 (924,9)	-	-	-
Mesh 4	322 (2091,9)	241 (1382,9)	162 (947,9)	-	-
Mesh 5	531 (3176,9)	337 (3947,9)	271 (1378,9)	173 (929,9)	-
Mesh 6	909 (5610,9)	649 (3663,9)	339 (2166,9)	261 (1421,9)	250 (1137,9)

Table 10: Example 2: GMRES iteration counts for the preconditioner, with $p = q = 1$ on unstructured meshes *without* a subdomain partition.

	Mesh 2	Mesh 3	Mesh 4	Mesh 5	Mesh 6
GMRES+ILU	555 (1619,9)	638 (2805,9)	792 (4502,9)	1862 (10221,9)	* (*,*)

Table 11: Example 2: GMRES+ILU iteration counts, when $p = 1$.

More precisely, METIS is employed to compute a subdomain partition of the coarse mesh $\mathcal{T}_H$; the local problems are then constructed based on grouping the fine level elements whose centroids lie within a given subdomain. Even though the coarse and fine meshes are no longer nested, we observe that the preconditioner performs extremely well, again leading to roughly a constant number of iterations required to attain convergence when the ratio of the coarse and fine mesh size is kept (roughly) constant. Indeed, comparing the iteration counts reported in Table 8 with the corresponding numbers presented in

$\sqrt{3}h^{-1}$	4	8	16	32
GMRES	145 (859,8)	210 (1126,7)	433 (2097,6)	947 (4699,6)
GMRES+ILU	16 (94,8)	29 (159,7)	66 (336,6)	170 (830,6)

Table 12: Example 3: GMRES and GMRES+ILU iteration counts, when $p = 1$.

Table 6, we observe that exploiting the unstructured meshes for the coarse grid solver, as apposed to the coarse meshes generated by METIS, leads to a slight reduction in the number of iterations needed to attain convergence. Table 9 considers the performance of the preconditioner when Mesh 1 is employed as the subdomain partition; thereby, $\mathcal{T}_{\mathcal{S}} \not\subseteq \mathcal{T}_H \not\subseteq \mathcal{T}_h$. The number of iterations to attain convergence are comparable to those reported in both Tables 8 & 6. In Table 10 we again consider employing the unstructured meshes for the coarse mesh $\mathcal{T}_H$, though now we do not employ a fixed subdomain partition of the computational domain Ω, cf. above. Indeed, as in Table 7, from Table 10 we again observe that by increasing the number of subdomain partitions as the coarse mesh is enriched leads to an increase in the number of iterations required to attain convergence.

For comparison Table 11 presents results using GMRES with the ILU preconditioning for $p = 1$. As noted in the previous example, the number of iterations required to attain convergence increases as the number of elements is increased; moreover, this approach again fails to converge on the finest mesh. In this setting, we see that when the additive Schwarz preconditioner is applied and METIS is used for the construction of both the subdomain partition $\mathcal{T}_{\mathcal{S}}$ and the composite elements in the coarse mesh $\mathcal{T}_H$, cf. Table 6, that the number of iterations required to attain convergence is always less than the corresponding quantity for the ILU preconditioner.

8.3. Example 3: Three–dimensional viscous flow in a cube domain

In the following two examples, we now turn our attention to three–dimensional laminar flows. To this end, as in the two–dimensional setting, we first study the performance of the proposed preconditioner on sequences of uniform meshes for a simple model problem. Here, we set Ω to be the unit cube $(0,1)^3$ and supplement the compressible Navier–Stokes equations (2.1) with an inhomogeneous forcing function $\mathbf{f}$, which is chosen so that the analytical solution is given by

$$\rho := \sin(2(x+y+z)) + 4\,, \qquad \rho v_1 = \rho v_2 = \rho v_3 := 0.2\sin(2(x+y+z)) + 4\,,$$
$$\rho E := (\sin(2(x+y+z)) + 4)^2\,.$$

As in Example 1, we set the dynamic viscosity coefficient μ to 1/10.

In this section we set $N = 8$; i.e., the fine level mesh $\mathcal{T}_h$ is partitioned into eight subdomains. As in the previous examples, the termination tolerance for the Newton–GMRES algorithm is set to 10^{-8}; similarly, the absolute tolerance for the underlying GMRES inner–solver is 10^{-8} with a restart every 50 iterations. Throughout this section we exploit uniform hexahedral elements for both the fine and the coarse level meshes $\mathcal{T}_h$ and $\mathcal{T}_H$, respectively, with granularity h and H, respectively.

$\sqrt{3}h^{-1}$ \ $\sqrt{3}H^{-1}$	2	4	8	16
4	27 (167,8)	-	-	-
8	37 (202,7)	34 (190,7)	-	-
16	47 (240,6)	47 (235,6)	40 (198,6)	-
32	68 (356,6)	68 (328,6)	63 (304,6)	44 (215,6)

Table 13: Example 3: GMRES iteration counts with the preconditioner, when $p = q = 1$.

Mesh $\mathcal{T}_h$ \ # Eles $\mathcal{T}_H$	350	2800
Mesh 1	230 (1397, 7)	-
Mesh 2	629 (3547,7)	304 (1880,7)
Mesh 3	1835(9184,7)	993 (5844,7)

Table 14: Example 4: GMRES iteration counts for the preconditioner, with $p = q = 1$ on unstructured meshes using coarse meshes created with METIS and a subdomain partition generated with METIS consisting of $N = 300$ domains.

For reference, in Table 12 we first show the number of iterations required for convergence with polynomial order $p = 1$, when no preconditioner is employed and when an ILU preconditioner is applied within the Newton–GMRES algorithm. Here, we observe that the number of iterations increases as the mesh size is halved at each step. Secondly, in Table 13 we consider the performance of the proposed additive Schwarz preconditioner when $p = q = 1$. As for the two–dimensional case, we observe that the maximum number of GMRES iterations remains roughly constant when the ratio of the coarse and fine mesh sizes is kept fixed, cf. [4, 7, 8]. Analogous behaviour is also observed for the cases when $p = q = 2$ and $p = 2$ and $q = 1$; though, as before, the latter case leads to a slight increase in the number of iterations required to attain convergence. For brevity, these numerics have been omitted. As in Example 1, on very coarse meshes, we note that the number of iterations required to attain convergence is typically smaller when the ILU preconditioner is employed when compared to the proposed additive Schwarz preconditioner; however, as before, this behaviour is reversed as the mesh is refined.

8.4. Laminar flow around a streamlined body

In this final example we consider laminar flow past a streamlined three–dimensional body. Here, the geometry of the body is based on a 10 percent thick airfoil with boundaries constructed by a surface of revolution. More precisely, the (half) geometry is given by the

Mesh $\mathcal{T}_h$ \ # Eles $\mathcal{T}_H$	350	2800
Mesh 1	158 (927, 7)	-
Mesh 2	458 (2732,7)	262 (1633,7)
Mesh 3	1053(5857,7)	715 (4413,7)

Table 15: Example 4: GMRES iteration counts for the preconditioner, with $p = q = 1$ on unstructured meshes using coarse meshes created with METIS and a subdomain partition generated with METIS consisting of $N = 150$ domains.

Mesh $\mathcal{T}_h$ \ # Eles $\mathcal{T}_H$	350	2800
Mesh 1	252 (1472, 7)	-
Mesh 2	662 (3822,7)	431 (2624,7)
Mesh 3	1890(9546,7)	1902 (9739,7)

Table 16: Example 4: GMRES iteration counts for the preconditioner, with $p = q = 1$ on unstructured meshes using coarse meshes created with METIS *without* a subdomain partition.

following expression

$$\begin{array}{ll}
16(x - 1/4)^2 + 400z^2 = 1, & 0 \le x \le 1/3,\ 0 \le y \le 1/100, \\
z = 1/(10\sqrt{2})(1 - x), & 1/3 < x \le 1,\ 0 \le y \le 1/100,\ z > 0, \\
z = -1/(10\sqrt{2})(1 - x), & 1/3 < x \le 1,\ 0 \le y \le 1/100,\ z < 0, \\
16(x - 1/4)^2 + 400(z^2 + (y - 1/100)^2) = 1, & 0 \le x \le 1/3,\ y > 1/100, \\
200(z^2 + (y - 1/100)^2) - (1 - x)^2 = 0, & 1/3 \le x \le 1,\ y > 1/100,
\end{array}$$

cf. [23]. This geometry is considered at laminar conditions with inflow Mach number equal to 0.5, at an angle of attack $\alpha = 1^\circ$, and Reynolds number $\mathrm{Re} = 5000$ with adiabatic no-slip wall boundary condition imposed. Throughout this section, we set a tolerance for termination of the Newton–GMRES algorithm equal to 10^{-8}, and the absolute tolerance for the underlying GMRES inner–solver to 10^{-8}; though here, we restart every 500 iterations.

In this example, we consider the performance of the proposed preconditioner on a sequence of unstructured hexahedral meshes. Here, the initial coarse mesh consists of 768 elements; two subsequent meshes, consisting of 6144 and 49152 elements, respectively, are constructed, based on uniformly refining this initial mesh. As in Example 2, we consider different strategies for constructing the subdomain mesh partition. Firstly, in Tables 14 & 15 we consider the performance of the proposed multilevel preconditioner based on exploiting METIS for the construction of both the subdomain partition $\mathcal{T}_{\mathcal{S}}$ and the composite elements in the coarse mesh $\mathcal{T}_H$ with $p = q = 1$ consisting of $N = 300$ and $N = 150$ domains, respectively. Here, we observe that the maximum number of GMRES iterations required to attain convergence grows slightly when the ratio of the coarse and fine mesh sizes is kept fixed. We remark that this may be simply pre-asymptotic behaviour;

Mesh $\mathcal{T}_h$ \ Mesh $\mathcal{T}_H$	Mesh 1	Mesh 2
Mesh 2	427 (2458,7)	-
Mesh 3	1151 (6459,7)	554 (3185,7)

Table 17: Example 4: GMRES iteration counts for the preconditioner, with $p = q = 1$ on unstructured meshes using a subdomain partition generated with METIS consisting of $N = 250$ domains.

Mesh $\mathcal{T}_h$ \ Mesh $\mathcal{T}_H$	Mesh 1	Mesh 2
Mesh 2	329 (1999,7)	-
Mesh 3	857 (4951,7)	453 (2626,7)

Table 18: Example 4: GMRES iteration counts for the preconditioner, with $p = q = 1$ on unstructured meshes using a subdomain partition generated with METIS consisting of $N = 150$ domains.

indeed, a slight growth of the number of iterations on relatively coarse meshes has been observed in our previous examples. We also point out that a reduction in the number of subdomains in the partition $\mathcal{T}_{\mathcal{S}}$ leads to a decrease in the number of iterations required to attain convergence, though at the expense of solving larger systems of local problems. As in Example 2, we now consider the case when $\mathcal{T}_{\mathcal{S}} = \mathcal{T}_H$, so that N, the number of subdomain partitions present in $\mathcal{T}_{\mathcal{S}}$, grows as $\mathcal{T}_H$ is refined, cf. Table 16; as before, we again observe that the number of iterations required to attain convergence grows by about a factor of 2 as the fine mesh is enriched, even when the ratio of the coarse and fine mesh sizes is kept fixed.

Finally, Tables 17, 18 & 19 present the number of iterations required to attain convergence when the unstructured hexahedral meshes are employed for the coarse mesh solver. In particular, here Tables 17 & 18 considers the case when $\mathcal{T}_{\mathcal{S}}$ is constructed based on employing METIS with $N = 250$ and $N = 150$, respectively. As in Example 2, we again observe that the number of iterations required to attain convergence is reduced when $\mathcal{T}_H$ is selected to be one of the user–defined unstructured meshes, as apposed to a coarse mesh generated by METIS. Moreover, we again notice that a reduction in N leads to a reduction in the number of iterations needed to attain convergence. Table 19 presents the case when $\mathcal{T}_H = \mathcal{T}_{\mathcal{S}}$; as before, we again observe that the number of iterations required to attain convergence grows as the fine mesh is enriched, even when the ratio of the coarse and fine mesh sizes is kept fixed.

For comparison in Table 20 the results using GMRES with ILU preconditioner are reported for $p = 1$. As we have previously observed, on very coarse meshes, the number of iterations required to attain convergence is typically smaller when the ILU preconditioner is employed when compared to the proposed additive Schwarz preconditioner; this behaviour is reversed as the mesh is enriched, cf. Table 14.

Mesh $\mathcal{T}_H$ / Mesh $\mathcal{T}_h$	Mesh 1	Mesh 2
Mesh 2	518 (3015,7)	-
Mesh 3	1430 (7942,7)	941 (5498,7)

Table 19: Example 4: GMRES iteration counts for the preconditioner, with $p = q = 1$ on unstructured meshes *without* a subdomain partition.

	Mesh 1	Mesh 2	Mesh 3
GMRES+ILU	59 (358,7)	264 (1515,7)	914 (5229,7)

Table 20: Example 4: GMRES+ILU iteration counts, when $p = 1$.

9. Concluding remarks

In this article we have considered the application of Schwarz-type domain decomposition preconditioners for discontinuous Galerkin finite element approximations of compressible fluid flows. Here, the coarse level solver employed within the proposed preconditioning strategy is based on exploiting ideas from so-called composite discontinuous Galerkin methods; this class of methods can easily handle general polygonal element domains consisting of agglomerated 'standard' elements. In this way, extremely coarse meshes may be defined, even for computational domains which contain small geometric details. The application of the preconditioner to viscous compressible flows in two– and three–dimensions clearly highlights the efficiency and robustness of the proposed solution strategy.

Acknowledgements

SG and PH acknowledge the financial support of the EPSRC under the grant EP/H005498. PH also acknowledges the support of the Leverhulme Trust.

References

[1] P. R. Amestoy, I. S. Duff, J. Koster, and J.-Y. L'Excellent. A fully asynchronous multifrontal solver using distributed dynamic scheduling. *SIAM J. Matrix Anal. Appl.*, 23(1):15–41, 2001.

[2] P. R. Amestoy, I. S. Duff, and J.-Y. L'Excellent. Multifrontal parallel distributed symmetricand unsymmetric solvers. *Comput. Methods Appl. Mech. Eng.*, 184:501–520, 2000.

[3] P. R. Amestoy, A. Guermouche, J.-Y. L'Excellent, and S. Pralet. Hybrid scheduling for the parallel solution of linear systems. *Parallel Computing*, 32(2):136–156, 2006.

[4] P.F. Antonietti and B. Ayuso. Schwarz domain decomposition preconditioners for discontinuous Galerkin approximations of elliptic problems: non-overlapping case. *M2AN Math. Model. Numer. Anal.*, 41(1):21–54, 2007.

[5] P.F. Antonietti and B. Ayuso. Multiplicative Schwarz methods for discontinuous Galerkin approximations of elliptic problems. *M2AN Math. Model. Numer. Anal.*, 42(3):443–469, 2008.

[6] P.F. Antonietti, S. Giani, and P. Houston. *hp*–Version composite discontinuous Galerkin methods for elliptic problems on complicated domains. *SIAM J. Sci. Comput.*, 35(3):A1417–A1439, 2013.

[7] P.F. Antonietti, S. Giani, and P. Houston. Domain decomposition preconditioners for discontinuous Galerkin methods for elliptic problems on complicated domains. *J. Sci. Comp.*, In press.

[8] P.F. Antonietti and P. Houston. A class of domain decomposition preconditioners for *hp*-discontinuous Galerkin finite element methods. *J. Sci. Comp.*, 46(1):124–149, 2011.

[9] S. Balay, K. Buschelman, W.D. Gropp, D. Kaushik, M.G. Knepley, L.C. McInnes, B.F. Smith, and H. Zhang. PETSc Web page, 2001. http://www.mcs.anl.gov/petsc.

[10] A.T. Barker, S.C. Brenner, E.-H. Park, and Li-Y. Sung. Two-level additive Schwarz preconditioners for a weakly over-penalized symmetric interior penalty method. *J. Sci. Comp.*, 47:27–49, 2011.

[11] F. Bassi, L. Botti, A. Colombo, D.A. Di Pietro, and P. Tesini. On the flexibility of agglomeration based physical space discontinuous Galerkin discretizations. *J. Comput. Phys.*, 231(1):45–65, 2012.

[12] F. Bassi and S. Rebay. A high-order accurate discontinuous finite element method for the numerical solution of the compressible Navier-Stokes equations. *J. Comp. Phys.*, 131:267–279, 1997.

[13] F. Bassi and S. Rebay. High-order accurate discontinuous finite element solution of the 2d Euler equations. *J. Comp. Phys.*, 138:251–285, 1997.

[14] C.E. Baumann and J.T. Oden. A discontinuous *hp* finite element method for the Euler and Navier-Stokes equations. *Internat. J. Numer. Methods Fluids*, 31:79–95, 1999.

[15] C.E. Baumann and J.T. Oden. An adaptive-order discontinuous Galerkin method for the solution of the Euler equations of gas dynamics. *Internat. J. Numer. Methods Engrg.*, 47:61–73, 2000.

[16] S.C. Brenner and K. Wang. Two-level additive Schwarz preconditioners for C^0 interior penalty methods. *Numer. Math.*, 102(2):231–255, 2005.

[17] A. Cangiani, E.H. Georgoulis, and P. Houston. *hp*-Version discontinuous Galerkin methods on polygonal and polyhedral meshes. *Math. Models Methods Appl. Sci.*, In press.

[18] V. Dolejší. On the discontinuous Galerkin method for the numerical solution of the Navier-Stokes equations. *Int. J. Numer. Meth. Fluids*, 45:1083–1106, 2004.

[19] M. Feistauer, J. Felcman, and I. Straškraba. *Mathematical and Computational Methods for Compressible Flow*. Clarendon Press, Oxford, 2003.

[20] X. Feng and O. A. Karakashian. Two-level additive Schwarz methods for a discontinuous Galerkin approximation of second order elliptic problems. *SIAM J. Numer. Anal.*, 39(4):1343–1365 (electronic), 2001.

[21] K.J. Fidkowski and D.L. Darmofal. A triangular cut-cell adaptive method for high-order discretizations of the compressible Navier-Stokes equations. *J. Comput. Phys.*, 225:1653–1672, 2007.

[22] 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. *J. Comput. Phys.*, 207(1):92–113, July 2005.

[23] S. Giani and P. Houston. Anisotropic *hp*–adaptive discontinuous Galerkin finite element methods for compressible fluid flows. *Int. J. Numer. Anal. Model.*, 9(4):928–949, 2012.

[24] W. Hackbusch and S.A. Sauter. Composite finite elements for the approximation of PDEs on domains with complicated micro-structures. *Numer. Math.*, 75:447âĂŞ–472, 1997.

[25] K. Harriman, P. Houston, B. Senior, and E. Süli. *hp*–Version discontinuous Galerkin methods

with interior penalty for partial differential equations with nonnegative characteristic form. In C.-W. Shu, T. Tang, and S.-Y. Cheng, editors, *Recent Advances in Scientific Computing and Partial Differential Equations. Contemporary Mathematics Vol. 330*, pages 89–119. AMS, 2003.

[26] R. Hartmann. The role of the Jacobian in the adaptive discontinuous Galerkin method for the compressible Euler equations. In G. Warnecke, editor, *Analysis and Numerics for Conservation Laws*, pages 301–316. Springer, 2005.

[27] R. Hartmann and P. Houston. Adaptive discontinuous Galerkin finite element methods for the compressible Euler equations. *J. Comput. Phys.*, 183(2):508–532, 2002.

[28] R. Hartmann and P. Houston. Symmetric interior penalty DG methods for the compressible Navier–Stokes equations I: Method formulation. *Int. J. Num. Anal. Model.*, 3(1):1–20, 2006.

[29] R. Hartmann and P. Houston. Symmetric interior penalty DG methods for the compressible Navier–Stokes equations II: Goal–oriented a posteriori error estimation. *Int. J. Num. Anal. Model.*, 3(2):141–162, 2006.

[30] R. Hartmann and P. Houston. An optimal order interior penalty discontinuous Galerkin discretization of the compressible Navier–Stokes equations. *J. Comput. Phys.*, 227(22):9670–9685, 2008.

[31] R. Hartmann and T. Leicht. Error estimation and anisotropic mesh refinement for 3d aerodynamic flow simulations. *J. Comput. Phys.*, 229(19), 2010.

[32] G. Karypis and V. Kumar. A fast and highly quality multilevel scheme for partitioning irregular graphs. *SIAM J. Sci. Comput.*, 20(1):359–392, 1999.

[33] J.T. Oden S. Prudhomme, F. Pascal and A. Romkes. Review of *a priori* error estimation for discontinuous Galerkin methods. Technical report, TICAM Report 00–27, Texas Institute for Computational and Applied Mathematics, 2000.

[34] J.J.W. van der Vegt and H. van der Ven. Space-time discontinuous Galerkin finite element method with dynamic grid motion for inviscid compressible flows, I. General formulation. *J. Comp. Phys.*, 182:546–585, 2002.

[35] K.G. van der Zee. An $H^1(P^h)$-coercive discontinuous Galerkin formulation for the Poisson problem: 1-D Analysis. Master's thesis, TU Delft, 2004.