h-multigrid agglomeration based solution strategies for discontinuous Galerkin discretizations of incompressible flow problems

L. Botti[a,*], A. Colombo[a], F. Bassi[a]

[a] *Dipartimento di Ingegneria e Scienze Applicate, Università di Bergamo via Marconi 4, 24044 Dalmine (BG), Italy*

Abstract

In this work we exploit agglomeration based h-multigrid preconditioners to speed-up the iterative solution of discontinuous Galerkin discretizations of the Stokes and Navier-Stokes equations. As a distinctive feature h-coarsened mesh sequences are generated by recursive agglomeration of a fine grid, admitting arbitrarily unstructured grids of complex domains, and agglomeration based discontinuous Galerkin discretizations are employed to deal with agglomerated elements of coarse levels. Both the expense of building coarse grid operators and the performance of the resulting multigrid iteration are investigated. For the sake of efficiency coarse grid operators are inherited through element-by-element L^2 projections, avoiding the cost of numerical integration over agglomerated elements. Specific care is devoted to the projection of viscous terms discretized by means of the BR2 dG method. We demonstrate that enforcing the correct amount of stabilization on coarse grids levels is mandatory for achieving uniform convergence with respect to the number of levels. The numerical solution of steady and unsteady, linear and non-linear problems is considered tackling challenging 2D test cases and 3D real life computations on parallel architectures.

*Corresponding author

Email addresses: lorenzo.botti@unibg.it (L. Botti), alessandro.colombo@unibg.it (A. Colombo), francesco.bassi@unibg.it (F. Bassi)

Significant execution time gains are documented.

Keywords: Multigrid, Agglomeration, Discontinuous Galerkin, Incompressible flow problems, Polyhedral elements

2010 MSC: 65N30, 65N55

---

1. Introduction

Discontinuous Galerkin (dG) methods have proved to be effective in the CFD field allowing to simulate complex physics in complex domains while guaranteeing accuracy and robustness. Although very popular for compressible fluid flow simulations, their adoption by incompressible fluid flow practitioners is still limited due to difficulties involved in the numerical solution of the Incompressible Navier-Stokes (INS) equations. On the one hand explicit and decoupled time integration strategies (*e.g.* Pressure Poisson Equation segregated methods) complicate the achievement of high-order pressure accuracy, thereby reducing the appeal of high-order accurate spatial discretizations. On the other hand fully implicit fully coupled velocity-pressure spatial discretisations result in systems of Differential Algebraic Equations (DAEs) that are very expensive to solve due to the indefiniteness of the resulting system matrices, their poor spectral properties, and the saddle point nature of the problem [1].

In this work, in order to speed up the numerical solutions of coupled variables dG discretizations of incompressible flow problems, we consider h-multigrid solution strategies on h-coarsened mesh sequences generated by recursive agglomeration of a fine grid. h-multigrid is very attractive from the efficiency viewpoint in the sense that the number of arithmetic operations needed to solve a discrete problem is proportional to the number of degrees of freedom. Convergence factors, that is the average residual decrease at each multigrid iteration, can be made h-independent and small.

In the context of dG discretizations p-multigrid has been fruitfully applied in practical applications see *e.g.* [2, 3, 4, 5], while the theoretical and practical investigation of h- and hp-multigrid is more recent. In 2003 Gopalakrishnan and Kanschat [6] analyzed a V-cycle preconditioner for diffusion and advection-diffusion problems. Multigrid algorithms for dG discretizations of elliptic problems were considered by Brenner *et al.* [7], who proved uniform convergence with respect to the number of levels for F-,V- and W-cycle on graded meshes, and Antonietti *et al.* [8], who provided similar results for W-cycle h-,p- and hp-multigrid. While the previous works employed h-refined mesh sequences, Prill *et al.* [9] considered smoothed aggregation to build coarse problems for h-multigrid dG solvers. The issue of developing optimal solvers for Composite discontinuous Galerkin Methods, first developed and analyzed Antonietti *et al.* [10] was considered by Antonietti *et al.* [11, 12]. More recently Antonietti *et al.* [13] analysed multigrid strategies for Interior Penalty dG discretizations over agglomerated elements meshes, while Wallraff and Leicht [14] and Wallraff *et al.* [15] applied an agglomeration based h-multigrid solver to dG discretizations of the compressible Reynolds Averaged Navier-Stokes (RANS) equations.

h-coarsening by agglomeration leads to unprecedented flexibility in the definition of the coarse meshes. Starting from a fine grid, a coarse mesh can be generated on the fly clustering together a number of mesh elements. The process can be repeated at will in a recursive manner resulting in a nested mesh sequence. Note that the generation of a sequence of nested grids by recursive refinement of a coarse mesh, *e.g.* by means of element subdivision techniques, might require to improve the rough approximation of the computational domain provided by the coarse mesh. While coarsening by agglomeration is flexible enough to account for complex 3D domains, physical frame dG discretizations allows to handle polyhedral elements of very general shape [16, 17, 18, 19, 20].

Nevertheless, as a consequence of the lack of efficient quadrature rules for agglomerated elements, numerical integration of bilinear and trilinear forms might lead to excessive matrix assembly costs, see Bassi *et al.* [17].

The present investigation focuses on efficiency of building coarse grid operators for dG discretizations of incompressible flow problems and effectiveness of the multigrid V-cycle iteration. In particular, we introduce a strategy for inheriting the BR2 dG formulation of [21], which provides optimal convergence properties and does not require numerical integration during assembly of coarse grid operators. Besides the BR2 formulation, here employed for the discretization of the viscous terms, inherited multigrid can be fruitfully employed for the discrete divergence and the discrete gradient operators, and also for the discretization of the non-linear convective flux terms appearing in the Navier-Stokes equations.

The material is organized as follows. In Section 2 we introduce agglomeration based dG discretization over h-coarsened mesh sequences. Section 3 is dedicated to presenting dG discretizations of incompressible flow problems: i) the incompressible Navier-Stokes equations spatial and temporal discretization in Section 3.1 and 3.2, respectively; ii) the discretization of the steady Stokes problem in Section 3.3; iii) the BR2 dG formulation in Section 3.4. The ingredients of the h-multigrid iteration are described in Section 4: i) the V-cycle in Section 4.1; ii) intergrid transfer operators in Section 4.2; iii) inherited coarse grid operators in Section 4.3. Section 5 briefly comments on the use of the h-multigrid V-cycle iteration as a preconditioner for iterative solvers and introduces block preconditioners for the Stokes problem. Performance gain assessment as compared to state-of-the-art iterative and direct solvers is conducted in Section 6. We consider i) elliptic problems in Section 6.1; ii) linear Stokes problems in Section 6.2; iii) non-linear incompressible flow problems in Section 6.3.

2. Agglomeration based dG discretizations

2.1. Coarsening by agglomeration

Let Ω be a bounded connected open domain. Consider a (possibly non conforming) mesh $\mathcal{T}_0$ of Ω composed of (possibly curved) elements $\kappa \in \mathcal{T}_0$ such that (i) for any $\kappa \in \mathcal{T}_0$, there exists a reference polygon $\widehat{\kappa}$ and a polynomial mapping $\Psi_\kappa : \widehat{\kappa} \to \kappa$ such that $\kappa = \Psi_\kappa(\widehat{\kappa})$. (ii) quadrature rules of arbitrary order are available on the reference polygon $\widehat{\kappa}$. The set of reference polygons includes but is not limited to triangular and quadrilateral reference elements in 2D, tetrahedral, hexahedral, pyramidal and prismatic reference elements in 3D.

Starting from $\mathcal{T}_0$ we can define a sequence of coarsened meshes $\{\mathcal{T}_\ell\}_{\ell=0,...,L}$ by agglomeration, see Figure 1. For the sake of notation we denote by κ_ℓ any element $\kappa \in \mathcal{T}_\ell$ whose diameter is h_κ, and we denote the mesh size of $\mathcal{T}_\ell,\ \ell = 0, ..., L$, by $h_\ell = \max_{\kappa\in\mathcal{T}_\ell}(h_\kappa)$. Agglomeration generates a hierarchic sequence of nested grids, in particular for any $\mathcal{T}_\ell,\ \ell = 0, ..., L-1$, we suppose that

- $\mathcal{T}_{\ell+1}$ is a disjoint partition of Ω obtained clustering together the elements of $\mathcal{T}_\ell$;
- every $\kappa \in \mathcal{T}_{\ell+1}$ is an open bounded connected subset of Ω and there exists $K_\ell^{\ell+1} \subset \mathcal{T}_\ell$ such that

$$\overline{\kappa}_{\ell+1} = \bigcup_{\kappa\in K_\ell^{\ell+1}} \overline{\kappa}_\ell. \tag{1}$$

  The card $(K_\ell^{\ell+1})$ cells clustered into the agglomerated element $\kappa_{\ell+1}$ are referred to as sub-elements.

- for every $\kappa \in \mathcal{T}_\ell$ there exists $K_0^\ell \subset \mathcal{T}_0$ such that

$$\overline{\kappa}_\ell = \bigcup_{\kappa\in K_0^\ell} \overline{\kappa}_0.$$

This can be obtained by applying (1) recursively and formalizes the fact that agglomerated elements on any mesh level ℓ can be expressed as a composition of elements belonging to the finer mesh $\mathcal{T}_0$.

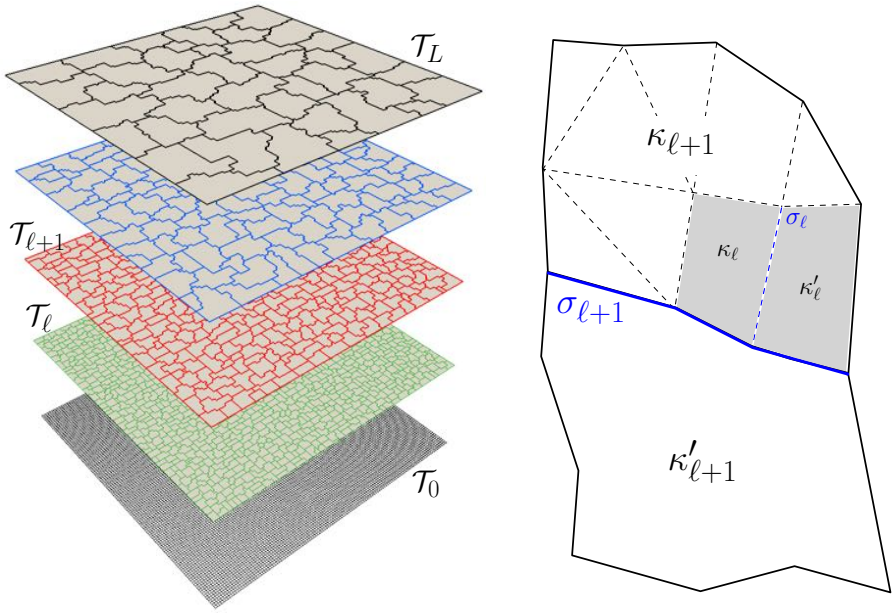

Figure 1: *Left*, example of a five levels ($L = 4$) h-coarsened mesh sequence. *Right*, two mesh elements $\kappa_{\ell+1}, \kappa'_{\ell+1} \in \mathcal{T}_{\ell+1}$ sharing a face σ_{l+1} and two mesh elements $\kappa_\ell, \kappa'_\ell \in K_\ell^{\ell+1} \subset \mathcal{T}_\ell$ sharing a face σ_l.

Clearly the coarsening steepness $h_{\ell+1}/h_\ell$ is influenced by the number of sub-elements composing aggregate elements as well as by the aspect ratio of agglomerated elements, see Figure 1. In this work all the mesh sequences are generated setting $\overline{\text{card}}(K) = 4, 8$ in two and three space dimensions, respectively, where the agglomeration rate $\overline{\text{card}}(K)$ is a strict upper bound for the number of sub-elements, so that $\text{card}\,(K) \leq \overline{\text{card}}(K), \forall \kappa \in \{\mathcal{T}_\ell\}_{\ell=1}^L$. The sequence of coarse meshes are generated by means of the library MGridGen [22], which allows to fix $\overline{\text{card}}(K)$ and relies on optimization algorithms in order to ensure overall good quality of the agglomerated elements. Note that, while the typical coarsening

steepness $h_{\ell+1}/h_\ell = 2$ can be obtained on regular Cartesian grids by regrouping 4 quadrilateral elements (2D case) or 8 hexahedral elements (3D case) sharing a node, on general unstructured grids leaving $\mathrm{card}\,(K)$ unbounded from below gives room for more aggressive aspect ratio optimizations.

To complete the definition of agglomerated grids we introduce inter-element boundaries where to define trace operators and fluxes of the dG discretization.

- Faces of an element $\kappa \in \mathcal{T}_0$ are defined as a portion of $\partial\kappa_0$ such that there exists a (hyperplanar) face $\widehat{\sigma}$ of the corresponding reference element $\widehat{\kappa}$ such that σ is the image of $\widehat{\sigma}$ through the mapping Ψ_κ.

- Faces of an agglomerated element $\kappa \in \mathcal{T}_\ell,\ \ell = 1, ..., L$, are defined as a portion of $\partial\kappa$ such that either $\sigma = \partial\kappa \cap \partial\Omega$ or there exists $\kappa' \in \mathcal{T}_\ell$, $\kappa' \neq \kappa$, such that $\sigma = \partial\kappa \cap \partial\kappa'$.

Mesh faces are collected in the sets $\mathcal{F}_\ell, \ell = 0, ..., L$. As mesh elements are composed by sub-elements, every face $\sigma \in \mathcal{F}_{\ell+1}$ is composed by sub-faces, also called facets, which belong to the set $\mathcal{F}_\ell$. Moreover, for every face σ we introduce the set $\Sigma_\ell^{\ell+1} \subset \mathcal{F}_\ell$ collecting the facets partitioning $\sigma_{\ell+1}$, *i.e.*,

$$\overline{\sigma}_{\ell+1} = \bigcup_{\sigma \in \Sigma_\ell^{\ell+1}} \overline{\sigma}_\ell,$$

and applying the definition recursively we get

$$\overline{\sigma}_\ell = \bigcup_{\sigma \in \Sigma_0^\ell} \overline{\sigma}_0,$$

where $\Sigma_0^\ell \subset \mathcal{F}_0$.

We introduce the set of boundary mesh faces $\sigma \in \mathcal{F}_\ell^{\mathrm{b}}$ such that $\sigma \subset \partial\Omega$ and let $\mathcal{F}_\ell^{\mathrm{i}} \stackrel{\mathrm{def}}{=} \mathcal{F}_\ell \setminus \mathcal{F}_\ell^{\mathrm{b}}$ denote the set of internal faces. Moreover, for any mesh

element $\kappa \in \mathcal{T}_\ell$, the set

$$\mathcal{F}_\kappa \stackrel{\text{def}}{=} \{\sigma \in \mathcal{F}_\ell \,|\, \sigma \subset \partial\kappa\}, \tag{2}$$

collects the mesh faces composing the boundary of κ. The maximum number of mesh faces composing the boundary of mesh elements is denoted by

$$N_{\partial_\ell} \stackrel{\text{def}}{=} \max_{\kappa \in \mathcal{T}_\ell} \left(\text{card}\left(\mathcal{F}_\kappa\right)\right). \tag{3}$$

For any mesh face $\sigma \in \mathcal{F}_\ell$ we define the set

$$\mathcal{T}_\sigma \stackrel{\text{def}}{=} \{\kappa \in \mathcal{T}_\ell \,|\, \sigma \subset \partial\kappa\}. \tag{4}$$

$\mathcal{T}_\sigma$ regroups the two mesh elements κ, κ' sharing σ if $\sigma \in \mathcal{F}_\ell^{\text{i}}$ while it consists of a single mesh element if $\sigma \in \mathcal{F}_\ell^{\text{b}}$.

We remark that agglomerated faces are in general not hyperplanar and have very general shapes. In Figure 1, a single non straight face $\sigma_{\ell+1}$ composed by three straight facets is represented. According to the definition of standard and agglomerated faces, for each mesh element the number faces is equal to the number of neighboring elements.

2.2. Physical frame dG discretizations

For each mesh level $\ell = 0, ..., L$ we consider the following broken polynomial spaces

$$\mathbb{P}_d^k(\mathcal{T}_\ell) \stackrel{\text{def}}{=} \left\{v_\ell \in L^2(\Omega) : v_\ell|_\kappa \in \mathbb{P}_d^k(\kappa),\, \forall \kappa \in \mathcal{T}_\ell\right\}, \tag{5}$$

where $\mathbb{P}_d^k(\kappa)$ is the restriction to a mesh element κ of the polynomials functions of d variables and total degree at most k, such that $N_{\text{dof}}^\kappa = \dim(\mathbb{P}_d^k) = \binom{k+d}{k}$. Since in this work $d = \{2, 3\}$ and no confusion is possible, we drop the subscript

and simply use the notation $\mathbb{P}^k$ in place of $\mathbb{P}^k_d$. Due to the nestedness of mesh elements we have $\mathbb{P}^k(\mathcal{T}_0) \supset \mathbb{P}^k(\mathcal{T}_1) \supset \mathbb{P}^k(\mathcal{T}_2) ... \supset \mathbb{P}^k(\mathcal{T}_L)$.

It is interesting to remark that physical frame discretizations are defined so to inherently span the space $\mathbb{P}^k(\mathcal{T}_\ell)$ and provide optimal approximation properties on regular h-refined mesh sequences $(\mathcal{T}_h)_{h>0}$, see *e.g.* Botti [23]. Accordingly, for all $\kappa \in \mathcal{T}_h$ and for each polynomial degree k, the L^2-orthogonal projection operator $\pi^k_\kappa : L^2(\kappa) \to \mathbb{P}^k(\kappa)$ is such that for all $v \in H^{k+1}(\kappa)$, there holds

$$\|v - \pi^k_\kappa v\|_{L^2(\kappa)} \leq C_{\text{app}} h^{k+1}_\kappa |v|_{H^{k+1}(\kappa)} \tag{6}$$

where C_{app} is independent of h and k. The optimal approximation estimate (6) holds true over mesh sequences composed of agglomerated elements of very general shape, in particular agglomerated elements meshes built on top of a curved elements mesh $\mathcal{T}_0$ are eligible to provide optimal approximation properties, see *e.g.* [24]. While the mesh regularity assumption implies star-shapedness of agglomerated elements, see [25] or [26] for additional details, the numerical convergence rates assessed in [16] and [17] allow to claim that optimal approximation properties are achieved over mesh sequences obtained by means of the MGridGen library (for instance using $\{\mathcal{T}_\ell\}_{\ell=0,...,L}$ in reversed order as a h-refined mesh sequence).

Sharp approximation properties estimates valid in the general framework of *hp*-discontinuous Galerkin discretizations have been obtained introducing the concept of shape regular d-simplexes coverings of polygonal/polyhedral meshes, see [18].

2.3. Basis functions choice

For a given $\kappa \in \mathcal{T}_\ell,\ \ell = 0, ..., L$, let $\Phi^k_{\mathbb{P}(\kappa)} = \{\varphi^\kappa_i, i = 1, ..., \dim(\mathbb{P}^k)\}$ denote a basis for $\mathbb{P}^k(\kappa)$. A basis for the space $\mathbb{P}^k(\mathcal{T}_\ell)$ is given by

$$\Phi^k \stackrel{\text{def}}{=} \{\Phi^k_{\mathbb{P}(\kappa)}\}_{\kappa \in \mathcal{T}_\ell}. \tag{7}$$

where each basis functions φ^κ_i is extended to Ω by simply setting $\varphi^\kappa_i = 0$ on $\Omega \setminus \kappa$.

From a practical viewpoint, in order to find a numerically satisfactory physical frame basis function we rely on the procedure proposed by Bassi, Botti, Colombo, Di Pietro and Tesini [16]. Starting from a monomial basis for each elementary space $\mathbb{P}^k(\kappa)$ defined according to a reference frame whose axes are aligned with the principal axes of inertia of κ, an L^2-orthonormal basis is inferred by means of the Modified Gram-Schmidt (MGS) orthogonalization procedure. The resulting basis functions $\Phi^k = \{\varphi^\kappa_i\}$ are hierarchical, orthogonal with respect to the L^2 inner product and provide well conditioned local matrices at high polynomial degrees. In particular the elementary mass matrices are unit diagonal, for any element shape.

The sole requirement to apply the orthogonalization strategy is the capability to compute the integrals of polynomial functions on each element κ. In the case of agglomerated elements this is achieved by exploiting the partition K_0 into standard-shaped sub-elements. The integral of any $v \in \mathbb{P}^k_d(\mathcal{T}_\ell)$ is computed as follows

$$\int_{\kappa_\ell} v(\mathbf{x})\ \mathrm{d}\mathbf{x} = \sum_{\kappa \in K^\ell_0} \int_{\kappa_0} v(\mathbf{x})\ \mathrm{d}\mathbf{x} = \sum_{\kappa \in K^\ell_0,\, \kappa_0 = \Psi_\kappa(\widehat{\kappa})} \int_{\widehat{\kappa}} (v \circ \Psi_\kappa)(\boldsymbol{\xi}) |J_{\Psi_\kappa}(\boldsymbol{\xi})|\ \mathrm{d}\boldsymbol{\xi}, \tag{8}$$

where $\mathbf{x}$ and $\boldsymbol{\xi}$ are physical and reference space coordinates, respectively, and J_{Ψ_κ} is the Jacobian of the mapping function Ψ_κ. The order of exactness required

for exact integration over each sub-element rapidly increases when considering high order polynomials on curved elements. Moreover, the use of Gaussian quadrature rules defined on the reference frame polygon $\widehat{\kappa}$ might lead to an excessive growth of the number of quadrature points if the agglomerated elements are composed of many sub-elements.

2.4. Average, jump and lifting operators

For all $\sigma \in \mathcal{F}_\ell^{\mathrm{i}}$ and all $v_\ell \in \mathbb{P}^k(\mathcal{T}_\ell)$ we introduce the jump and average operators defined as follows:

$$[\![v_\ell]\!]_\sigma \stackrel{\mathrm{def}}{=} v_\ell|_\kappa - v_\ell|_{\kappa'}, \qquad \{\!\{v_\ell\}\!\}_\sigma \stackrel{\mathrm{def}}{=} \frac{1}{2}(v_\ell|_\kappa + v_\ell|_{\kappa'}).$$

Whenever no confusion can arise we drop the subscript σ. On boundary faces, we conventionally set $[\![v_\ell]\!] = \{\!\{v_\ell\}\!\} = v_\ell$. When v is vector-valued, the weighted average operator acts componentwise on the function v.

For all $\sigma \in \mathcal{F}_\ell^{\mathrm{b}}$, $\mathbf{n}_\sigma$ denotes the unit outward normal to Ω, whereas, for all $\sigma \in \mathcal{F}_\ell^{\mathrm{i}}$ such that $\sigma \subset \partial\kappa \cap \partial\kappa'$, $\mathbf{n}_\sigma$ is defined as the unit normal pointing out of κ (the order of the elements sharing σ is arbitrary but fixed). For all $\sigma \in \mathcal{F}_\ell$ we define the (local) lifting operator $\mathbf{r}_\sigma : L^2(\sigma) \to [\mathbb{P}_d^k(\mathcal{T}_\ell)]^d$, such that, for all $\phi \in L^2(\sigma)$,

$$\int_\Omega \mathbf{r}_\sigma(\phi)\cdot\boldsymbol{\tau}_\ell = \int_\sigma \phi\{\!\{\boldsymbol{\tau}_\ell\}\!\}\cdot\mathbf{n}_\sigma \qquad \forall \boldsymbol{\tau}_\ell \in [\mathbb{P}^k(\mathcal{T}_\ell)]^d. \tag{9}$$

Note that the support of $\mathbf{r}_\sigma$ consists of one and two mesh elements if $\sigma \in \mathcal{F}_\ell^{\mathrm{b}}$ and $\sigma \in \mathcal{F}_\ell^{\mathrm{i}}$, respectively, that is

$$\mathrm{supp}(\mathbf{r}_\sigma) = \bigcup_{\kappa\in\mathcal{T}_\sigma} \overline{\kappa}.$$

For any function $v \in H^1(\mathcal{T}_\ell)$, we also introduce the global lifting

$$\mathbf{R}_l(v) := \sum_{\sigma \in \mathcal{F}_l} \mathbf{r}_\sigma([\![v]\!]), \tag{10}$$

which collects the local lifting contributions, note that $[\![v]\!]_\sigma \in L^2(\sigma)$.

3. Incompressible flow problems

3.1. Incompressible Navier-Stokes equations dG discretization

We consider the unsteady INS equations with Dirichlet boundary conditions,

$$\partial_t \mathbf{u} + \mathbf{u} \cdot \boldsymbol{\nabla} \mathbf{u} - \nabla \cdot (\nu \boldsymbol{\nabla} \mathbf{u}) + \boldsymbol{\nabla} p = 0 \qquad \text{in } \Omega \times (0, t_F), \tag{11a}$$

$$\nabla \cdot \mathbf{u} = 0 \qquad \text{in } \Omega \times (0, t_F), \tag{11b}$$

$$\mathbf{u} = \mathbf{f} \qquad \text{on } \partial\Omega \times (0, t_F), \tag{11c}$$

$$\mathbf{u}(\cdot, t = 0) = \mathbf{u}^0, \qquad \text{in } \Omega, \tag{11d}$$

$$\langle p \rangle_\Omega = 0, \tag{11e}$$

where $\mathbf{u} \in \mathbb{R}^d$ is the velocity vector, p is the pressure, $\nu > 0$ denotes the (constant) viscosity, $\mathbf{f}$ is the boundary datum, $\mathbf{u}^0$ is the initial condition, and $\langle \cdot \rangle_\Omega$ denotes the average value over Ω. The density has been assumed to be uniform and equal to one.

Letting $\mathbf{F}^\nu = -\nu \nabla \otimes \mathbf{u}$ and $\mathbf{F}^c = \mathbf{u} \otimes \mathbf{u} + p\mathbf{Id}$ be the viscous and convective flux functions, Eqs. (11a)-(11b) can be written in conservation form as

$$\partial_t \mathbf{u} + \nabla \cdot \mathbf{F} = \mathbf{0}, \tag{12}$$

where $\mathbf{F} \overset{\text{def}}{=} [\mathbf{F}^c + \mathbf{F}^\nu, \mathbf{u}] \in \mathbb{R}^d \otimes \mathbb{R}^{d+1}$. For $d = 3$ we get

$$\mathbf{F} = \begin{bmatrix} uu + \nu\frac{\partial u}{\partial x} + p & uv + \nu\frac{\partial v}{\partial x} & uw + \nu\frac{\partial w}{\partial x} & u \\ vu + \nu\frac{\partial u}{\partial y} & vv + \nu\frac{\partial v}{\partial y} + p & vw + \nu\frac{\partial w}{\partial y} & v \\ wu + \nu\frac{\partial u}{\partial z} & wv + \nu\frac{\partial v}{\partial z} & ww + \nu\frac{\partial w}{\partial z} + p & w \end{bmatrix} \tag{13}$$

The dG discretization of the Navier-Stokes equations we rely upon consists in seeking $(\mathbf{u}_0, p_0) \in [\mathbb{P}^k(\mathcal{T}_0)]^{d+1}$ such that

$$\begin{aligned} &\int_\Omega \mathbf{v}_0 \cdot \partial_t \mathbf{u}_0 - \int_\Omega \nabla_0 \mathbf{v}_0 : \left[\mathbf{F}^c(\mathbf{u}_0, p_0) + \widetilde{\mathbf{F}}^\nu\left(\nabla_0 \mathbf{u}_0, \mathbf{R}_0(\mathbf{u}_0)\right)\right] \\ &+ \sum_{\sigma \in \mathcal{F}_0} \int_\sigma \mathbf{n}_\sigma \otimes [\![\mathbf{v}_0]\!] : \left[\widehat{\mathbf{F}}^c\left(\mathbf{u}_0^{\kappa,\kappa'}, p_0^{\kappa,\kappa'}\right) + \widehat{\mathbf{F}}^\nu\left(\nabla_0 \mathbf{u}_0^{\kappa,\kappa'}, \eta_\sigma \mathbf{r}_\sigma^{\kappa,\kappa'}\left([\![\mathbf{u}_0]\!]\right)\right)\right] = 0, \end{aligned} \tag{14a}$$

$$-\int_\Omega \nabla_0 q_0 \cdot \mathbf{u}_0 + \sum_{\sigma \in \mathcal{F}_0} \int_\sigma [\![q_0]\!]\, \mathbf{n}_\sigma \cdot \widehat{\mathbf{u}}(\mathbf{u}_0^{\kappa,\kappa'}, p_0^{\kappa,\kappa'}) = 0, \tag{14b}$$

$$\int_\Omega p_0 = 0$$

for all $(\mathbf{v}_0, q_0) \in [\mathbb{P}^k(\mathcal{T}_0)]^{d+1}$.

According to the BR2 scheme, proposed in [21] and theoretically analyzed in [27] and [28], the viscous numerical fluxes read

$$\widetilde{\mathbf{F}}^\nu\left(\nabla_0 \mathbf{u}_0, \mathbf{R}_0\left(\mathbf{u}_0\right)\right) \overset{\text{def}}{=} -\nu \nabla_0 \mathbf{u}_0 + \mathbf{R}_0\left(\mathbf{u}_0\right), \tag{15}$$

$$\widehat{\mathbf{F}}^\nu\left(\nabla_0 \mathbf{u}_0^{\kappa,\kappa'}, \eta_\sigma \mathbf{r}_\sigma^{\kappa,\kappa'}\left([\![\mathbf{u}_0]\!]\right)\right) \overset{\text{def}}{=} -\nu \{\!\{\nabla_0 \mathbf{u}_0\}\!\} + \eta_\sigma \{\!\{\mathbf{r}_\sigma\left([\![\mathbf{u}_0]\!]\right)\}\!\}. \tag{16}$$

$\widetilde{\mathbf{F}}^\nu$ is the consistent discrete gradient while $\widehat{\mathbf{F}}^\nu$ is the consistent diffusive flux ensuring symmetry and stability of the scheme. In particular coercivity holds provided that η_σ is greater than the maximum number of faces of the elements sharing σ. The inviscid physical and numerical fluxes of the dG discretization

reads

$$\mathbf{F}^{!\nu}(\mathbf{w}_0) \stackrel{\text{def}}{=} [\mathbf{F}^c(\mathbf{w}_0), \mathbf{u}_0] \quad \text{and} \quad \widehat{\mathbf{F}}^{!\nu}(\mathbf{w}_0) \stackrel{\text{def}}{=} \left[\widehat{\mathbf{F}}^c(\mathbf{w}_0), \widehat{\mathbf{u}}_0(\mathbf{w}_0)\right], \tag{17}$$

respectively. The inviscid numerical fluxes $\widehat{\mathbf{F}}^{!\nu}$ result from the exact solution of local Riemann problems based on an artificial compressibility perturbation of the Euler equations, as proposed in [29].

Boundary conditions are enforced weakly by properly defining for each $\sigma \in \mathcal{F}_0^{\mathrm{b}}$ a boundary state $(\mathbf{u}^{\kappa'^b}, p^{\kappa'^b})$ having support on the interface of a ghost neighboring elements κ'^b. The ghost boundary state is defined based on the method of characteristics exploiting the hyperbolic nature of the artificial compressibility perturbation of the Euler equation. Accordingly the ghost state depend on the Dirichlet datum f but also on the internal state $(\mathbf{u}^\kappa, p^\kappa)$. Once ghost states are computed, handling of internal and boundary faces is similar: for each $\sigma \in \mathcal{F}_0$ two neighboring elements κ, κ' concur to the computation of numerical fluxes and lifting operators.

3.2. Navier-Stokes equations temporal discretization

For the sake of notation we collect the vector velocity and the pressure polynomial expansions in the vector $\mathbf{w} \stackrel{\text{def}}{=} (u_{0,1}, ..., u_{0,d}, p_0) \in [\mathbb{P}^k(\mathcal{T}_0)]^{d+1}$ and identify the unknown vector at time t_n with $\mathbf{w}_0^n$, that is $\mathbf{w}_0^n = [\mathbf{u}_0(t_n), p_0(t_n)]$. For all $\delta\mathbf{w}_0, \mathbf{w}_0, \mathbf{k}_0 \in [\mathbb{P}^k(\mathcal{T}_0)]^{d+1}$ we introduce the following bilinear and trilinear

forms

$$m_i(\delta w_i, k_i) = + \sum_{\kappa \in \mathcal{T}_0} \int_\kappa k_i \, \delta w_i$$

$$j^{!\nu}_{i,j}(\mathbf{w}, \delta w_j, k_i) = - \sum_{\kappa \in \mathcal{T}_0} \sum_{l=1}^{d} \int_\kappa \frac{\partial k_i}{\partial x_l} \frac{\partial F^{!\nu}_{l,i}(\mathbf{w})}{\partial w_j} \delta w_j + \sum_{\sigma \in \mathcal{F}_0} \sum_{l=1}^{d} \int_\sigma [\![k_i]\!] n_{\sigma,l} \frac{\partial \widehat{F}^{!\nu}_{l,i}(\mathbf{w})}{\partial w_j} \delta w_j \tag{18}$$

$$j^{\nu}_{i}(\delta w_i, k_i) = - \sum_{\kappa \in \mathcal{T}_0} \sum_{l=1}^{d} \int_\kappa \frac{\partial k_i}{\partial x_l} \frac{\partial \widetilde{F}^{\nu}_{l,i}(w_i)}{\partial w_i} \delta w_i + \sum_{\sigma \in \mathcal{F}_0} \sum_{l=1}^{d} \int_\sigma [\![k_i]\!] n_{\sigma,l} \frac{\partial \widehat{F}^{\nu}_{l,i}(w_i)}{\partial w_i} \delta w_i \tag{19}$$

$$f^{m}_{i}(w_i, k_i) = - \sum_{\kappa \in \mathcal{T}_0} \int_\kappa k_i \, w_i \tag{20}$$

$$f^{!\nu}_{i}(\mathbf{w}, k_i) = + \sum_{\kappa \in \mathcal{T}_0} \sum_{l=1}^{d} \int_\kappa \frac{\partial k_i}{\partial x_l} F^{!\nu}_{l,i}(\mathbf{w}) - \sum_{\sigma \in \mathcal{F}_0} \sum_{l=1}^{d} \int_\sigma [\![k_i]\!] n_{\sigma,l} \widehat{F}^{!\nu}_{l,i}(\mathbf{w}),$$

$$f^{\nu}_{i}(w_i, k_i) = + \sum_{\kappa \in \mathcal{T}_0} \sum_{l=1}^{d} \int_\kappa \frac{\partial k_i}{\partial x_l} \widetilde{F}^{\nu}_{l,i}(w_i) - \sum_{\sigma \in \mathcal{F}_0} \sum_{l=1}^{d} \int_\sigma [\![k_i]\!] n_{\sigma,l} \widehat{F}^{\nu}_{l,i}(w_i).$$

In the above definitions we dropped the mesh sequence subscript for notation convenience. Note that, by abuse of notation, (18) is a bilinear (resp. trilinear) when $F_{l,i}(\mathbf{w})$ is a linear (resp. non-linear) function of w_j.

Given the initial condition $\mathbf{w}_0^0 = \mathbf{w}_0(t = 0) \in [\mathbb{P}^k(\mathcal{T}_0)]^{d+1}$ we define the sequence $\mathbf{w}_0^{n+1}$ iteratively by means of the backward Euler method:

Algorithm 1 Backward Euler

1: set $\mathbf{w}_0^n = \mathbf{w}_0^0$, $n_F = \dfrac{t_F}{\delta t}$
2: **for** $n = 0, 1, ..., n_F$ **do**
3: set $\mathbf{w}_0^{n+1} \leftarrow \mathbf{w}_0^n$
4: **while** $\delta \mathbf{w}_0$ is too large **do**
5: find $\delta \mathbf{w}_0 \in [\mathbb{P}^k(\mathcal{T}_0)]^{d+1}$ such that, for all $\mathbf{k}_0 \in [\mathbb{P}^k(\mathcal{T}_0)]^{d+1}$

$$\frac{1}{\delta t}\sum_{i=1}^{d} m_i(\delta w_{0,i}, k_{0,i}) + \sum_{i=1}^{d+1}\sum_{j=1}^{d+1} j_{i,j}^{\mathrm{l}\nu}(\mathbf{w}_0^{n+1}, \delta w_{0,j}, k_{0,i}) + \sum_{i=1}^{d} j_i^{\nu}(\delta w_{0,i}, k_{0,i}) =$$
$$\frac{1}{\delta t}\sum_{i=1}^{d} f_i^{m}(w_{0,i}^{n+1} - w_{0,i}^{n}, k_i) + \sum_{i=1}^{d+1} f_i^{\mathrm{l}\nu}(\mathbf{w}_0^{n+1}, k_i) + \sum_{i=1}^{d} f_i^{\nu}(w_{0,i}^{n+1}, k_i) \tag{21}$$

$$\langle \delta w_{0,d+1} \rangle_\Omega = 0, \tag{22}$$

6: set $\mathbf{w}_0^{n+1} \mathrel{+}= \delta \mathbf{w}_0$
7: **end while**
8: **end for**

Note that the continuation condition at line 4 can be replaced by checking that a proper norm of the right hand side of Equation (21) is too large. Equation (22) is needed since the average value of the pressure increment is left undefined in Equation (21).

To recast Problem (21) in operator form we let $X^k(\mathcal{T}_0) = \mathbb{P}^k(\mathcal{T}_0) \times [\mathbb{P}^k(\mathcal{T}_0)]^d$

and introduce the linear operators such that, $\forall \mathbf{w}_0 \in [\mathbb{P}^k(\mathcal{T}_0)]^{d+1}$

$$
\begin{aligned}
(J_0^A \delta\mathbf{u}_0, \mathbf{v}_0)_{[L^2(\Omega)]^d} &= \sum_{i=1}^{d} j_i^{\nu}(\delta w_{0,i}, k_{0,i}), && \forall\, \delta\mathbf{u}_0, \mathbf{v}_0 \in [\mathbb{P}^k(\mathcal{T}_0)]^d,\\
(J_0^B(\mathbf{w}_0)\delta\mathbf{u}_0, q_0)_{L^2(\Omega)} &= \sum_{j=1}^{d} j_{d+1,j}^{!\nu}(\mathbf{w}_0, \delta w_{0,j}, k_{0,d+1}), && \forall\, (q_0, \delta\mathbf{u}_0) \in X^k(\mathcal{T}_0),\\
(J_0^{B^t}(\mathbf{w}_0)\delta p_0, \mathbf{v}_0)_{[L^2(\Omega)]^d} &= \sum_{i=1}^{d} j_{i,d+1}^{!\nu}(\mathbf{w}_0, \delta w_{0,d+1}, k_{0,i}), && \forall\, (\delta p_0, \mathbf{v}_0) \in X^k(\mathcal{T}_0),\\
(J_0^C(\mathbf{w}_0)\delta p_0, q_0)_{L^2(\Omega)} &= j_{d+1,d+1}^{!\nu}(\mathbf{w}_0, \delta w_{0,d+1}, k_{0,d+1}), && \forall\, \delta p_0, q_0 \in \mathbb{P}^k(\mathcal{T}_0),\\
(J_0^D(\mathbf{w}_0)\delta\mathbf{u}_0, \mathbf{v}_0)_{[L^2(\Omega)]^d} &= \sum_{i=1}^{d}\sum_{j=1}^{d} j_{i,j}^{!\nu}(\mathbf{w}_0, \delta w_{0,j}, k_{0,i}), && \forall\, \delta\mathbf{u}_0, \mathbf{v}_0 \in [\mathbb{P}^k(\mathcal{T}_0)]^d,\\
(M_0\delta\mathbf{u}_0, \mathbf{v}_0)_{[L^2(\Omega)]^d} &= \sum_{i=1}^{d} m_0(\delta w_{0,i}, k_{0,i}), && \forall\, \delta\mathbf{u}_0, \mathbf{v}_0 \in [\mathbb{P}^k(\mathcal{T}_0)]^d.
\end{aligned}
$$

Moveover we introduce the residuals of momentum and continuity equations

$$
\begin{aligned}
\mathbf{f}_0^M(\mathbf{w}_0^{n,n+1}, \mathbf{v}_0) &= \sum_{i=1}^{d} f_i^m(u_{0,i}^{n+1} - u_{0,i}^n, v_{0,i}) + \sum_{i=1}^{d} f_i^{v,nv}(\mathbf{w}_0^{n+1}, v_{0,i}), \quad \forall \mathbf{v}_0 \in [\mathbb{P}^k(\mathcal{T}_0)]^d,\\
f_0^C(\mathbf{w}_0^{n+1}, q_0) &= f_{d+1}(\mathbf{w}_0^{n+1}, q_0), && \forall q_0 \in \mathbb{P}^k(\mathcal{T}_0).
\end{aligned}
$$

Problem (21) amounts at solving a linear system in the form:

$$
A_0^{\mathrm{INS}} \begin{bmatrix} \delta\mathbf{u}_0 \\ \delta p_0 \end{bmatrix} = \begin{bmatrix} \mathbf{f}_0^M \\ f_0^C \end{bmatrix}, \text{ with } A_0^{\mathrm{INS}} = \begin{bmatrix} M_0 + J_0^A + J_0^D & J_0^{B^t} \\ J_0^B & J_0^C \end{bmatrix}. \tag{23}
$$

3.3. Stokes equations dG discretization

The steady Stokes equation problem can be obtained dropping the time derivative and the convective term, that is the first two terms in equation (11a). In case of a steady Stokes flow the inviscid interface fluxes $\widehat{\mathbf{F}}^c$ and $\widehat{\mathbf{u}}$ can be explicitly computed as the solution of a linear hyperbolic system, see [29] for details. The resulting dG discretization reads: find $(\mathbf{u}_0, p_0) \in [\mathbb{P}^k(\mathcal{T}_0)]^{d+1}$ such

that

$$\nu a_0(\mathbf{u}_0, \mathbf{v}_0) + b_0(\mathbf{v}_0, p_0) = \mathbf{f}_0^M(\mathbf{f}, \mathbf{v}_0), \qquad \forall \mathbf{v}_0 \in [\mathbb{P}^k(\mathcal{T}_0)]^d, \tag{24a}$$

$$-b_0(\mathbf{u}_0, q_0) + c_0(p_0, q_0) = f_0^C(\mathbf{f}, q_0), \qquad \forall q_0 \in \mathbb{P}^k(\mathcal{T}_0), \tag{24b}$$

$$\langle p_0 \rangle_\Omega = 0, \tag{24c}$$

where

$$b_0(\mathbf{v}_0, q_0) \stackrel{\text{def}}{=} -\int_\Omega q_0 \nabla_0 \cdot \mathbf{v}_0 + \sum_{\sigma \in \mathcal{F}_0} \int_\sigma [\![\mathbf{v}_0]\!] \cdot \mathbf{n}_\sigma \{\!\{q_0\}\!\}, \tag{25}$$

$$= \int_\Omega \mathbf{v}_0 \cdot \nabla_0 q_0 - \sum_{\sigma \in \mathcal{F}_0^i} \int_\sigma \{\!\{\mathbf{v}_0\}\!\} \cdot \mathbf{n}_\sigma [\![q_0]\!], \tag{26}$$

$$c_0(q_0, t_0) \stackrel{\text{def}}{=} \sum_{\sigma \in \mathcal{F}_0^i} h_\sigma \int_\sigma [\![q_0]\!][\![t_0]\!], \tag{27}$$

$$a_0(\mathbf{v}_0, \mathbf{w}_0) \stackrel{\text{def}}{=} \sum_{i=1}^d j_0^\nu(v_{0,i}, w_{0,i}) = \sum_{i=1}^d a_0^{\text{BR2}}(v_{0,i}, w_{0,i}), \tag{28}$$

and the terms on right hand side defined below accounts for the week imposition of Dirichlet boundary conditions

$$\mathbf{f}_0^M(\mathbf{f}; \mathbf{v}_0) = -\sum_{i=0}^d \int_\Omega \mathbf{R}_0(f_i) \cdot \nabla_0 v_{0,i} + \eta_\sigma \sum_{\sigma \in \mathcal{F}_0^{\text{b}}} \sum_{i=0}^d \int_\Omega \mathbf{r}_\sigma(f_i) \cdot \mathbf{r}_\sigma(v_{0,i}), \tag{29}$$

$$f_0^C(\mathbf{f}; q_0) = -\sum_{\sigma \in \mathcal{F}_0^{\text{b}}} \int_\sigma \mathbf{f} \cdot \mathbf{n}_\sigma q_0. \tag{30}$$

According to (28) the discretization of the viscous term can be obtained applying the BR2 method to each velocity component, *cf.* definitions (19) and (36). The dG discretization in (24) was analysed by Di Pietro in [30], see also [26, Chapter 6].

Problem (24) has a block structure which we can take advantage for devising

effective preconditioners. To this end we define the operators A_0, C_0 and B_0

$$(A_0\mathbf{v}_0, \mathbf{w}_0)_{[L^2(\Omega)]^d} = \nu a_0(\mathbf{v}_0, \mathbf{w}_0), \qquad \forall \mathbf{v}_0, \mathbf{w}_0 \in [\mathbb{P}^k(\mathcal{T}_0)]^d, \tag{31}$$

$$(C_0 q_0, r_0)_{L^2(\Omega)} = c_0(q_0, r_0), \qquad \forall q_0, r_0 \in \mathbb{P}^k(\mathcal{T}_0), \tag{32}$$

$$(B_0\mathbf{v}_0, q_0)_{L^2(\Omega)} = -b_0(\mathbf{v}_0, q_0), \qquad \forall (\mathbf{v}_0, q_0) \in [\mathbb{P}^k(\mathcal{T}_0)]^d \times \mathbb{P}^k(\mathcal{T}_0). \tag{33}$$

Note that according to (26) we are able to infer $(\mathbf{v}_0, B_0^t q_0)_{[L^2(\Omega)]^d} = b_0(\mathbf{v}_0, q_0)$.

Problem (24) amounts at solving a linear system in the form:

$$A_0^{\text{Stk}} \begin{bmatrix} \mathbf{u}_0 \\ p_0 \end{bmatrix} = \begin{bmatrix} \mathbf{f}_0^M \\ f_0^C \end{bmatrix}, \text{ with } A_0^{\text{Stk}} = \begin{bmatrix} A_0 & B_0^t \\ B_0 & C_0 \end{bmatrix}. \tag{34}$$

3.4. Viscous terms dG discretization

The BR2 dG formulation is employed for the discretization of the viscous terms of Equation (11a) being an important building block of both the Stokes and the Navier-Stokes dG discretizations. In this work we focus on the performance of solving the BR2 dG discretization of the following model Poisson problem

$$\begin{cases} -\triangle u = f & \text{in } \Omega, \\ u = 0 & \text{on } \partial\Omega. \end{cases} \tag{35}$$

Assessing and improving the performance of h-multigrid for elliptic problems has been a critical step for achieving satisfactory performances on incompressible flow problems.

The BR2 method can be written into a consistent and symmetric bilinear form plus stabilization term as follows: For all $v_0, w_0 \in \mathbb{P}^k(\mathcal{T}_0)$,

$$a_0^{\text{BR2}}(v_0, w_0) \overset{\text{def}}{=} a_0^{\text{BR2,CS}}(v_0, w_0) + s_0^{\text{BR2}}(v_0, w_0), \tag{36}$$

where

$$a_0^{\mathrm{BR2,CS}}(v_0,w_0) \stackrel{\mathrm{def}}{=} \int_\Omega (\nabla_0 v_0 - \mathbf{R}_0(v_0)) \cdot (\nabla_0 w_0 - \mathbf{R}_0(w_0)) - \int_\Omega \mathbf{R}_0(v_0) \cdot \mathbf{R}_0(w_0), \tag{37}$$

$$s_0^{\mathrm{BR2}}(v_0,w_0) \stackrel{\mathrm{def}}{=} \sum_{\sigma \in \mathcal{F}_\ell} \eta_\sigma \int_\Omega \mathbf{r}_\sigma([\![v_0]\!]) \cdot \mathbf{r}_\sigma([\![w_0]\!]). \tag{38}$$

Using the definition of the BR2 bilinear form in (36) the discretization of (35) reads:

$$\text{Find } u_0 \in W_0 \ s.t. \ a_0^{\mathrm{BR2}}(u_0,v_0) = \int_\Omega f v_0 \text{ for all } v_0 \in W_0. \tag{39}$$

Well-posedness of problem (39) was proved by Brezzi et al. [27].

The BR2 method in (39) can be reformulated as follows: Given $f \in L^2(\Omega)$,

$$\text{find } u_0 \in \mathbb{P}^k(\mathcal{T}_0) \ s.t. \ A_0^{\mathrm{BR2}} u_0 = \pi_{\mathcal{T}_0}^k f, \tag{40}$$

where $\pi_{\mathcal{T}_0}^k$ is the $L^2(\Omega)$-orthogonal projection operator into $\mathbb{P}^k(\mathcal{T}_0)$ and A_0^{BR2} is the fine grid operator, such that

$$(A_0^{\mathrm{BR2}} u_0, v_0)_{L^2(\Omega)} = a_0^{\mathrm{BR2}}(u_0,v_0), \ \forall u_0, v_0 \in \mathbb{P}^k(\mathcal{T}_0). \tag{41}$$

Solving (40) amounts to solving a linear system in the form

$$A_0^{\mathrm{BR2}} u_0 = f_0. \tag{42}$$

4. *h*-multigrid V-cycle

The ability to define h-coarsened mesh sequences by agglomeration and to perform high-order accurate dG discretizations on general polygonal grids allows

to exploit h-multigrid solvers to improve the efficiency of the solution strategy. In this work we consider h-multigrid preconditioners for the dG discretization of the Stokes problem (24) and the linearized Navier-Stokes problem (21). Besides incompressible flow problems we will also focus on the performance of h-multigrid applied to purely elliptic scalar problems. The interest is twofold: firstly the discretization of the viscous terms relies on a BR2 dG discretization and secondly block preconditioners for the Stokes problem require effective preconditioners for the discrete Laplace operator, see Section 5.

The linear (or linearized) systems arising from dG discretizations of the Navier-Stokes, Stokes and Laplace equation, see Eqs. (23), (34) and (42), are in the form

$$A_0 w_0 = f_0. \tag{43}$$

Solving (43) allows to compute the degrees of freedom $\underline{w}_0$ of $w_0 \in [\mathbb{P}^k(\mathcal{T}_\ell)]^{\mathsf{v}}$, where $\mathsf{v} = 4$ in case of incompressible flow problems. As opposite we deal with a scalar function in case of the Laplace equation, *i.e.* $\mathsf{v} = 1$. In order to accelerate convergence towards w_0 the multigrid iteration relies on several coarse grid problems in the form

$$A_\ell w_\ell = f_\ell, \quad \ell = 1, ..., L. \tag{44}$$

Coarse grids $\mathcal{T}_\ell$ are explicitly built and coarse grid solutions belong to piecewise polynomial spaces defined over them, $w_\ell \in [\mathbb{P}^k(\mathcal{T}_\ell)]^{\mathsf{v}}$. The purpose of this section is to provide some insight on how coarse grid operators A_ℓ are built and how coarse grid solutions w_ℓ can be effectively employed to speed up the achievement of the fine grid solution w_0. A comprehensive review can be found in Refs. [31, 32, 33], while the analysis of multigrid as a preconditioner for Krylov solvers, is analyzed in detail by Smith, Bjørstad and Gropp [32].

4.1. Multigrid V-cycle iteration

In this work we consider the multigrid V-cycle iteration, that is the simplest way of traversing the mesh sequence generated by agglomeration coarsening of the fine grid, see Section 2.1. The recursive multigrid V-cycle for the problem $A_\ell w_\ell = f_\ell$ on level ℓ reads:

Algorithm 2 $\mathrm{MG}_{\mathcal{V}}(l, f_\ell, w_\ell)$

if $(\ell = L)$ **then**
  $\overline{w}_\ell = A_\ell^{-1} f_\ell$
end if
if $(\ell < L)$ **then**
  Pre-smoothing:
  $\overline{w}_\ell = \mathrm{SMOOTH}(w_\ell, f_\ell)$
  Coarse grid correction:
  $r_\ell = f_\ell - A_\ell \overline{w}_\ell$
  $r_{\ell+1} = \mathcal{I}_\ell^{\ell+1} r_\ell$
  $e_{\ell+1} = \mathrm{MG}_{\mathcal{V}}(\ell + 1, r_{\ell+1}, 0)$
  $\widehat{w}_\ell = \overline{w}_\ell + \mathcal{I}_{\ell+1}^{\ell} e_{\ell+1}$
  Post-smoothing:
  $\overline{w}_\ell = \mathrm{SMOOTH}(\widehat{w}_\ell, f_\ell)$
end if
return $\overline{w}_\ell$

As a result of invoking $\mathrm{MG}_{\mathcal{V}}(0, f_0, u_0)$ the grid sequence is traversed moving towards coarser levels, one grid at a time, until the coarsest level L is reached. Note that the coarsest level can be thought to be located at bottom of the V-shaped cycle. The descending phase is followed by ascension towards finer levels, until a new approximation $\overline{u}_0$ of the exact solution over the fine grid is available. This marks the completion of one V-cycle iteration.

On each level ℓ except the coarsest one, three distinct phases take place: pre-smoothing, coarse grid correction and post-smoothing, see Algorithm 2. In the pre-smoothing phase a few iterations (one or two, in this work) of a standard preconditioned iterative solver are performed in order to damp high-frequency modes of the error $e_\ell = \overline{w}_\ell - w_\ell$. Since the convergence of iterative

solvers deteriorates when trying to damp low-frequency modes resulting in an inefficient solution process, the error equations $A_{\ell+1}e_{\ell+1} = r_{\ell+1}$ are solved on a coarser grid (level $\ell+1$), where low-frequency modes appears more oscillatory. Once the error is computed it is transferred back to level ℓ and used to correct the solution: $\widehat{w}_\ell = \widehat{w}_\ell + e_\ell$. Before doing so post-smoothing ensures that only smooth components of the error have survived. Note that correction takes place only after the residual equations have been accurately solved on the coarsest level, usually with a direct solver.

It is interesting to remark that, due to the linearity of the original problem, solving $A_\ell w_\ell = f_\ell$ with an arbitrary initial guess is equivalent to solving the residual equations $A_\ell e_\ell = r_\ell$ with a zero initial guess. As a consequence the coarse grid correction requires to compute the residual $r_{\ell+1}$ but the computation of an initial guess $e_{\ell+1}$ is not needed. The residual is approximated by projecting its fine counterpart $r_\ell = f_\ell - A_\ell \overline{w}_\ell$ to level $\ell+1$ which requires the definition of the so called restriction operator $\mathcal{I}_\ell^{\ell+1}$: $\mathbb{P}^k(\mathcal{T}_\ell) \to \mathbb{P}^k(\mathcal{T}_{\ell+1})$. Similarly the error $e_{\ell+1}$ needs to be prolongated to the coarse mesh by means of the prolongation operator $\mathcal{I}_{\ell+1}^{\ell}$: $\mathbb{P}^k(\mathcal{T}_{\ell+1}) \to \mathbb{P}^k(\mathcal{T}_\ell)$. The implementation of the intergrid transfer operators $\mathcal{I}_{\ell+1}^{\ell}$ and $\mathcal{I}_\ell^{\ell+1}$ is described in detail in the next section.

4.2. Intergrid transfer operators

In any geometric multigrid strategy, transfer operators are required to map functions between two subsequent spaces in the set $\{\mathbb{P}^k(\mathcal{T}_\ell)\}_{\ell=0,\dots,L}$. Since nested grids are generated by recursive coarsening of a fine grid, also the polynomial spaces are nested, that is $\mathbb{P}^k(\mathcal{T}_0) \supset \mathbb{P}^k(\mathcal{T}_1) \dots \supset \mathbb{P}^k(\mathcal{T}_L)$. Accordingly, *prolongation* is the natural injection $\mathcal{I}_{\ell+1}^{\ell} : \mathbb{P}^k(\mathcal{T}_{\ell+1}) \to \mathbb{P}^k(\mathcal{T}_\ell)$ such that

$$\sum_{\kappa\in\mathcal{T}_\ell} \int_\kappa (\mathcal{I}_{\ell+1}^{\ell} u_{\ell+1} - u_{\ell+1}) \; = 0, \;\; \forall u_{\ell+1} \in \mathbb{P}^k(\mathcal{T}_{\ell+1}), \tag{45}$$

while *restriction* is the L^2 projection $\mathcal{I}_\ell^{\ell+1} : \mathbb{P}^k(\mathcal{T}_\ell) \to \mathbb{P}^k(\mathcal{T}_{\ell+1})$ such that

$$\sum_{\kappa \in \mathcal{T}_\ell} \int_\kappa (\mathcal{I}_\ell^{\ell+1} u_\ell - u_\ell)\, v_{\ell+1} = 0, \;\; \forall (u_\ell, v_{\ell+1}) \in \mathbb{P}^k(\mathcal{T}_\ell) \times \mathbb{P}^k(\mathcal{T}_{\ell+1}) \tag{46}$$

The matrix counterpart $\mathbf{I}_\ell^{\ell+1} \in \mathbb{R}^{m,n}, m = \mathrm{card}(\mathcal{T}_{\ell+1})\, N_{\mathrm{dof}}^\kappa, n = \mathrm{card}(\mathcal{T}_\ell)\, N_{\mathrm{dof}}^\kappa$, of the restriction operator $\mathcal{I}_\ell^{\ell+1}$ is a sparse block matrix composed of $\mathrm{card}(\mathcal{T}_\ell)$ blocks $\mathbf{I}_{\kappa_{\ell+1},\kappa_\ell} \in \mathbb{R}^{N_{\mathrm{dof}}^\kappa, N_{\mathrm{dof}}^\kappa}$, each defined as

$$\mathbf{I}_{\kappa_{\ell+1},\kappa_\ell} \stackrel{\mathrm{def}}{=} \mathbf{M}_{\kappa_{\ell+1}}^{-1}\, \mathbf{M}_{\kappa_{\ell+1},\kappa_\ell}, \tag{47}$$

where

$$(\mathbf{M}_{\kappa_\ell})_{i,j} \stackrel{\mathrm{def}}{=} \int_{\kappa_\ell} \varphi_i^{\kappa_\ell}\, \varphi_j^{\kappa_\ell}, \qquad i,j \in 1, ..., N_{\mathrm{dof}}^\kappa, \tag{48}$$

$$\left(\mathbf{M}_{\kappa_{\ell+1},\kappa_\ell}\right)_{i,j} \stackrel{\mathrm{def}}{=} \int_{\kappa_\ell} \varphi_i^{\kappa_{\ell+1}}\, \varphi_j^{\kappa_\ell}, \qquad i,j \in 1, ..., N_{\mathrm{dof}}^\kappa. \tag{49}$$

In particular each row of the matrix $\mathbf{I}_\ell^{\ell+1}$ is associated to an element $\kappa_{\ell+1} \in \mathcal{T}_{\ell+1}$ and consist of $\mathrm{card}(K_\ell^{\ell+1})$ blocks. Similarly the prolongation matrix $\mathbf{I}_{\ell+1}^\ell \in \mathbb{R}^{n,m}$ consists of $\mathrm{card}(\mathcal{T}_{\ell-1})$ blocks $\mathbf{I}_{\kappa_\ell,\kappa_{\ell+1}} \in \mathbb{R}^{N_{\mathrm{dof}}^\kappa, N_{\mathrm{dof}}^\kappa}$, each defined as

$$\mathbf{I}_{\kappa_\ell,\kappa_{\ell+1}} \stackrel{\mathrm{def}}{=} \mathbf{M}_{\kappa_\ell}^{-1}\, \left(\mathbf{M}_{\kappa_{\ell+1},\kappa_\ell}\right)^{\mathrm{T}}. \tag{50}$$

Thanks to the use of orthonormal basis functions, the elemental mass matrices reduce to the identity matrix, that is $\mathbf{M}_\kappa = \mathbf{M}_{\kappa_{\ell+1}} = \mathbf{Id}$, reducing the computational cost of computing transfer operators. Moreover, since $\mathbf{I}_\ell^{\ell+1} = (\mathbf{I}_{\ell+1}^\ell)^T$, storing transfer operators requires to store only $\sum_{\ell=0}^{L-1} \mathrm{card}(\mathcal{T}_\ell)$ blocks of size $(N_{\mathrm{dof}}^\kappa)^2$.

Interestingly the same holds true when considering intergrid transfer operators for a vector function $w_\ell \in [\mathbb{P}^k(\mathcal{T}_\ell)]^{\mathsf{v}}$. Restriction and prolongation can be

efficiently performed componentwise, that is

$$\begin{aligned}
&\text{Restriction:} && w_{\ell+1,i} = \mathcal{I}_{\ell}^{\ell+1} w_{\ell,i}, && i = 1, ..., \mathsf{v}, \ell = 0, ..., L-1 \\
&\text{Prolongation:} && w_{\ell,i} = \mathcal{I}_{\ell+1}^{\ell} w_{\ell+1,i}, && i = 1, ..., \mathsf{v}, \ell = L-1, ..., 0
\end{aligned}$$

without explicitly building the matrix $\mathbf{I}_{\ell}^{\ell+1}$ associated to the restriction operator $\mathcal{I}_{\ell}^{\ell+1} : [\mathbb{P}^k(\mathcal{T}_\ell)]^{\mathsf{v}} \to [\mathbb{P}^k(\mathcal{T}_{\ell+1})]^{\mathsf{v}}$. Matrix-free restriction and prolongation algorithms are implemented as follows.

Algorithm 3 Restriction of a vector function $w_\ell \in [\mathbb{P}^k(\mathcal{T}_\ell)]^{\mathsf{v}}$

for $\kappa_{\ell+1} \in \mathcal{T}_{\ell+1}$ **do**
  for $\kappa_\ell \in K_\ell^{\ell+1}$ **do**
    for $i \in \{1, ..., \mathsf{v}\}$ **do**
      $\underline{w}_{i,\kappa_{\ell+1}}$ += $\mathbf{M}_{\kappa_{\ell+1},\kappa_\ell} \underline{w}_{i,\kappa_\ell}$
    end for
  end for
end for

Algorithm 4 Prolongation of a vector function $w_\ell \in [\mathbb{P}^k(\mathcal{T}_\ell)]^{\mathsf{v}}$

for $\kappa_{\ell+1} \in \mathcal{T}_{\ell+1}$ **do**
  for $\kappa_\ell \in K_\ell^{\ell+1}$ **do**
    for $i \in \{1, ..., \mathsf{v}\}$ **do**
      $\underline{w}_{i,\kappa_\ell} = \mathbf{M}^T_{\kappa_{\ell+1},\kappa_\ell} \underline{w}_{i,\kappa_{\ell+1}}$
    end for
  end for
end for

Note that $\underline{w}_{i,\kappa_{\ell+1}}$ and $\underline{w}_{i,\kappa_\ell}$ are the degrees of freedom associated with the i-th component of the function $w_{\ell+1}|_{\kappa_{\ell+1}} \in [\mathbb{P}^k(\kappa_{\ell+1})]^{\mathsf{v}}$ and of the function $w_\ell|_{\kappa_\ell} \in [\mathbb{P}^k(\kappa_\ell)]^{\mathsf{v}}$, respectively.

4.3. Coarse grid operators

Two possibilities are available for building coarse grid problems in the form of (44), the so called *non-inherited* multigrid, where discrete operators are as-

sembled on each grid of the mesh sequence, and *inherited* multigrid, where coarse operators are recursively built by restricting the fine grid operators.

Recently, evidence emerged that non-inherited multigrid might be preferable from the convergence rates viewpoint, in particular Antonietti *et al.* [8] have analyzed h-multigrid Interion Penalty dG discretization of the Laplace equation demonstrating that only non-inherited multigrid provides uniform convergence with respect to the number of levels. Nevertheless, inherited coarse grid operators are significantly cheaper to compute since evaluation of numerical fluxes and assembly of bilinear forms over agglomerated elements grids is avoided. In particular, from the implementation viewpoint i) numerical integration and basis function orthogonalization over agglomerated elements meshes are required only for the computation of intergrid transfer operators; ii) the parallel implementation is simpler since flux computation on partition boundaries requires to access data from ghost agglomerated elements (note that ghost agglomerated elements are composed by many layers of fine ghost cells). Accordingly choosing between inherited and non-inherited version of h-multigrid might involve a trade-off between efficiency of the solver strategy and computational cost of assembling coarse grid operators. In order to avoid such an uncomfortable situation, in what follows we propose to heal the convergence degradation of inherited multigrid using a rescaled Galerkin projection of the stabilization terms of the BR2 dG discretization. The possibility to suitably rescale the stabilization terms of dG discretizations to improve the performance of coarse grid solvers was first proposed by Antonietti *et al.* [34] in the context of two level Schwarz methods for overpenalized Interior Penalty formulations.

4.3.1. BR2 dG discretization

Consider the BR2 bilinear form $a_0^{\mathrm{BR2}}(v_0, w_0) : \mathbb{P}^k(\mathcal{T}_0) \times \mathbb{P}^k(\mathcal{T}_0) \to \mathbb{R}$ defined in (36) and the corresponding fine grid operator $A_0^{\mathrm{BR2}} : \mathbb{P}^k(\mathcal{T}_0) \to \mathbb{P}^k(\mathcal{T}_0)$, see

definition (41). The coarse grid operators $A_\ell^{\mathrm{BR2}}, A_\ell^{\mathcal{I},\mathrm{BR2}} : \mathbb{P}^k(\mathcal{T}_\ell) \to \mathbb{P}^k(\mathcal{T}_\ell),\ \ell = 1, ..., L$, read

$$\text{Non-Inherited:} \qquad (A_\ell^{\mathrm{BR2}} v_\ell, w_\ell)_{L^2(\Omega)} \stackrel{\mathrm{def}}{=} a_\ell^{\mathrm{BR2}}(v_\ell, w_\ell), \qquad \forall v_\ell, w_\ell \in \mathbb{P}^k(\mathcal{T}_\ell), \tag{51}$$

$$\text{Inherited:} \quad (A_\ell^{\mathcal{I},\mathrm{BR2}} v_\ell, w_\ell)_{L^2(\Omega)} \stackrel{\mathrm{def}}{=} a_0^{\mathrm{BR2}}(\mathcal{I}_\ell^0 v_\ell, \mathcal{I}_\ell^0 w_\ell), \quad \forall v_\ell, w_\ell \in \mathbb{P}^k(\mathcal{T}_\ell), \tag{52}$$

where $\mathcal{I}_\ell^0 = \mathcal{I}_1^0\ \mathcal{I}_2^1\ ...\ \mathcal{I}_\ell^{\ell-1}$ and $\mathcal{I}_{\ell+1}^\ell : \mathbb{P}^k(\mathcal{T}_{\ell+1}) \to \mathbb{P}^k(\mathcal{T}_\ell)$, $\ell = 0, ..., L$-1 are the prolongation operators introduced in Section 4.2. Continuity and coercivity bounds for the $a_\ell^{\mathrm{BR2}}(u_\ell, v_\ell)$ bilinear form over agglomerated elements meshes were proven by Bassi *et al.* [16], in particular on level ℓ stability holds provided that $\eta_\sigma > N_{\partial_\ell}$. Accordingly the coarse grid problems $A_\ell^{\mathrm{BR2}} u_\ell = f_\ell$, $\ell = 1, ..., L$, arising in the non-inherited version of the multigrid V-cycle iteration, see Section 4.1, are well-posed.

In what follows we demonstrate that, given the BR2 bilinear form in (36), for all $v_\ell, w_\ell \in \mathbb{P}^k(\mathcal{T}_\ell)$

$$a_0^{\mathrm{BR2,CS}}(\mathcal{I}_\ell^0 v_\ell, \mathcal{I}_\ell^0 w_\ell) = a_\ell^{\mathrm{BR2,CS}}(v_\ell, w_\ell), \tag{53}$$

$$s_0^{\mathrm{BR2}}(\mathcal{I}_\ell^0 v_\ell, \mathcal{I}_\ell^0 w_\ell) \neq s_\ell^{\mathrm{BR2}}(v_\ell, w_\ell). \tag{54}$$

Accordingly the difference between A_ℓ^{BR2} and $A_\ell^{\mathcal{I},\mathrm{BR2}}$ hinges on the stabilization term.

Using the local and global lifting operator definitions (9) and (10) the consistency and symmetry BR2 bilinear form in (37) can be rewritten as

$$a_0^{\mathrm{BR2,CS}}(v_0, w_0) = a_0^{\mathrm{BR2,CS}_\kappa}(v_0, w_0) + a_0^{\mathrm{BR2,CS}_\sigma}(v_0, w_0)$$

where

$$a_0^{\mathrm{BR2,CS}_\kappa}(v_0, w_0) = \sum_{\kappa\in\mathcal{T}_0}\int_\kappa \nabla_0 v_0 \nabla_0 w_0, \tag{55}$$

$$a_0^{\mathrm{BR2,CS}_\sigma}(v_0, w_0) = -\sum_{\sigma\in\mathcal{F}_0}\int_\sigma \left(\{\!\{\nabla_0 v_0\}\!\}\cdot\mathbf{n}_\sigma\,[\![w_0]\!] + [\![v_0]\!]\,\{\!\{\nabla_0 w_0\}\!\}\cdot\mathbf{n}_\sigma\right). \tag{56}$$

Since $\mathbb{P}^k(\mathcal{T}_0)\supset\mathbb{P}^k(\mathcal{T}_\ell)$, we get

$$\begin{aligned} a_0^{\mathrm{BR2,CS}_\kappa}(\mathcal{I}_\ell^0 v_\ell, \mathcal{I}_\ell^0 w_\ell) &= \sum_{\kappa\in\mathcal{T}_0}\int_\kappa \nabla_0(\mathcal{I}_\ell^0 v_\ell)\cdot\nabla_0(\mathcal{I}_\ell^0 w_\ell) \\ &= \sum_{\kappa\in\mathcal{T}_\ell}\sum_{\kappa\in K_0^\ell}\int_\kappa \nabla_0(\mathcal{I}_\ell^0 v_\ell)\cdot\nabla_0(\mathcal{I}_\ell^0 w_\ell) \\ &= \sum_{\kappa\in\mathcal{T}_\ell}\int_\kappa \nabla_\ell v_\ell\cdot\nabla_\ell w_\ell = a_\ell^{\mathrm{BR2,CS}_\kappa}(v_\ell, w_\ell). \end{aligned} \tag{57}$$

Since $[\![\mathcal{I}_\ell^0 v_\ell]\!]_{\sigma_0} = 0$ if $\sigma_0\notin\mathcal{F}_\ell\cap\mathcal{F}_0$, we get $-a_0^{\mathrm{BR2,CS}_\sigma}(\mathcal{I}_\ell^0 v_\ell, \mathcal{I}_\ell^0 w_\ell) =$

$$\begin{aligned} &= \sum_{\sigma\in\mathcal{F}_0}\int_\sigma \{\!\{\nabla_0(\mathcal{I}_\ell^0 v_\ell)\}\!\}\cdot\mathbf{n}_\sigma\,[\![\mathcal{I}_\ell^0 w_\ell]\!] && + \sum_{\sigma\in\mathcal{F}_0}\int_\sigma [\![\mathcal{I}_\ell^0 v_\ell]\!]\,\{\!\{\nabla_0(\mathcal{I}_\ell^0 w_\ell)\}\!\}\cdot\mathbf{n}_\sigma \\ &= \sum_{\sigma\in\mathcal{F}_\ell}\sum_{\sigma\in\Sigma_0^\ell}\int_\sigma \{\!\{\nabla_0(\mathcal{I}_\ell^0 v_\ell)\}\!\}\cdot\mathbf{n}_\sigma[\![\mathcal{I}_\ell^0 w_\ell]\!] && + \sum_{\sigma\in\mathcal{F}_\ell}\sum_{\sigma\in\Sigma_0^\ell}\int_\sigma [\![\mathcal{I}_\ell^0 v_\ell]\!]\{\!\{\nabla_0(\mathcal{I}_\ell^0 w_\ell)\}\!\}\cdot\mathbf{n}_\sigma \\ &= \sum_{\sigma\in\mathcal{F}_\ell}\int_\sigma \{\!\{\nabla_\ell v_\ell\}\!\}\cdot\mathbf{n}_\sigma\,[\![w_\ell]\!] && + \sum_{\sigma\in\mathcal{F}_\ell}\int_\sigma [\![v_\ell]\!]\,\{\!\{\nabla_\ell w_\ell\}\!\}\cdot\mathbf{n}_\sigma \\ &= \int_\Omega \nabla_\ell v_\ell\cdot\mathbf{R}(w_\ell) && + \int_\Omega \mathbf{R}(v_\ell)\cdot\nabla_\ell w_\ell = -a_\ell^{\mathrm{BR2,CS}_\sigma}(v_\ell, w_\ell) \end{aligned} \tag{58}$$

The above result together with (57) prove (53).

Using the local lifting operator definitions (9) the stabilization term in (38) can be rewritten as

$$s_0^{\mathrm{BR2}}(v_0, w_0) = \sum_{\sigma\in\mathcal{F}_0}\eta_\sigma\int_\sigma \{\!\{\mathbf{r}_\sigma^k([\![v_0]\!])\}\!\}\cdot\mathbf{n}_\sigma\,[\![w_0]\!]. \tag{59}$$

The inherited stabilization term reads

$$
\begin{aligned}
s_0^{\mathrm{BR2}}(\mathcal{I}_\ell^0 v_\ell, \mathcal{I}_\ell^0 w_\ell) &= \sum_{\sigma\in\mathcal{F}_0} \eta_\sigma \int_\sigma \{\!\{ \mathbf{r}_\sigma^k([\![\mathcal{I}_\ell^0 v_\ell]\!]) \}\!\} \cdot \mathbf{n}_\sigma [\![\mathcal{I}_\ell^0 w_\ell]\!] \\
&= \sum_{\sigma\in\mathcal{F}_\ell} \sum_{\sigma\in\Sigma_0^\ell} \eta_\sigma \int_\sigma \{\!\{ \mathbf{r}_\sigma^k([\![\mathcal{I}_\ell^0 v_\ell]\!]) \}\!\} \cdot \mathbf{n}_\sigma [\![\mathcal{I}_\ell^0 w_\ell]\!] \\
&= \sum_{\sigma\in\mathcal{F}_\ell} \sum_{\sigma\in\Sigma_0^\ell} \eta_\sigma \int_\sigma \{\!\{ \mathbf{r}_\sigma^k([\![v_\ell]\!]) \}\!\} \cdot \mathbf{n}_\sigma [\![w_\ell]\!] \\
&= \sum_{\sigma\in\mathcal{F}_\ell} \sum_{\sigma\in\Sigma_0^\ell} \eta_\sigma \int_\Omega \mathbf{r}_\sigma^k([\![v_\ell]\!]) \cdot \mathbf{r}_\sigma^k([\![w_\ell]\!])
\end{aligned} \tag{60}
$$

while its non-inherited counterpart is simply

$$
s_\ell^{\mathrm{BR2}}(v_\ell, w_\ell) = \sum_{\sigma\in\mathcal{F}_\ell} \eta_\sigma \int_\Omega \mathbf{r}_{\sigma_\ell}^k([\![v_\ell]\!]) \cdot \mathbf{r}_{\sigma_\ell}^k([\![w_\ell]\!]). \tag{61}
$$

The inherited stabilization term (60) introduces an excessive amount of stabilization as compared to (61) having a detrimental effect on the spectral properties of inherited coarse grid operators, see Antonietti *et al.* [8].

In order to recover the correct amount of stabilization we propose to rescale it introducing the scaling term

$$
\mathcal{H}_{\sigma_0}^{\sigma_\ell} \stackrel{\mathrm{def}}{=} \frac{\eta_{\sigma_\ell}}{\eta_{\sigma_0}} \frac{h_{\kappa_0,\kappa_0'}}{h_{\kappa_\ell,\kappa_\ell'}}, \tag{62}
$$

and defining the rescaled stabilization term

$$
\widetilde{s}_0^{\mathrm{BR2}}(\mathcal{I}_\ell^0 v_\ell, \mathcal{I}_\ell^0 w_\ell) \stackrel{\mathrm{def}}{=} \sum_{\sigma\in\mathcal{F}_\ell} \sum_{\sigma\in\Sigma_0^\ell} \mathcal{H}_{\sigma_0}^{\sigma_\ell}\, \eta_\sigma \int_\Omega \mathbf{r}_{\sigma_0}^k([\![\mathcal{I}_\ell^0 v_\ell]\!]) \cdot \mathbf{r}_{\sigma_0}^k([\![\mathcal{I}_\ell^0 w_\ell]\!]), \tag{63}
$$

such that

$$\widetilde{s}_\ell^{\mathrm{BR2}}(\mathcal{I}_\ell^0 v_\ell, \mathcal{I}_\ell^0 w_\ell) \lesssim s_\ell^{\mathrm{BR2}}(v_\ell, w_\ell), \qquad \forall v_\ell, w_\ell \in \mathbb{P}^k(\mathcal{T}_\ell), \tag{64}$$

$$\widetilde{s}_\ell^{\mathrm{BR2}}(\mathcal{I}_\ell^0 v_\ell, \mathcal{I}_\ell^0 v_\ell) \gtrsim s_\ell^{\mathrm{BR2}}(v_\ell, v_\ell), \qquad \forall v_\ell \in \mathbb{P}^k(\mathcal{T}_\ell). \tag{65}$$

To prove (64) we recall the following bounds on the local lifting operator: let $\phi \in L^2(\sigma)$, for all $\sigma \in \mathcal{F}_\ell$

$$C_{\mathrm{r}} h_{\kappa,\kappa'}^{-1/2} \|[\![\phi]\!]\|_{L^2(\sigma)} \le \|\mathbf{r}_\sigma^k(\phi)\|_{[L^2(\Omega)]^d} \le C_{\mathrm{tr}} h_{\kappa,\kappa'}^{-1/2} \|\phi\|_{L^2(F)}, \tag{66}$$

where $h_{\kappa,\kappa'} = \min(h_\kappa, h_{\kappa'})$, see *e.g.* [27, Lemma 2], [35, Lemma 7.2] or [26, Lemma 4.33 and Lemma 5.18] for a proof. The constant C_{tr} depends on d, k and the shape regularity of the elements sharing σ and is inherited from the discrete trace inequality: for all $\kappa \in \mathcal{T}_\ell$, $\sigma \in \mathcal{F}_\kappa$

$$\|v_\ell\|_{L^2(\sigma)} \le C_{\mathrm{tr}} h_{\kappa,\kappa'}^{-1/2} \|v_\ell\|_{L^2(\kappa)} \tag{67}$$

While trace inequalities in the form of (67) are commonly available in the context of simplicial and quadrilateral/hexahedral meshes we refer to [26, Lemma 1.46] for a version valid in the context of matching simplicial submeshes and to [18, 19] for an optimal version derived in the context of polygonal/polyhedral element meshes.

Using (66) we get the following bounds

$$\begin{aligned}
s_\ell^{\mathrm{BR2}}(v_\ell, w_\ell) &= \sum_{\sigma \in \mathcal{F}_\ell} \eta_{\sigma_\ell} \int_\Omega \mathbf{r}_{\sigma_\ell}^k([\![v_\ell]\!]) \cdot \mathbf{r}_{\sigma_\ell}^k([\![w_\ell]\!]) \\
&\le \sum_{\sigma \in \mathcal{F}_\ell} \eta_{\sigma_\ell} \|\mathbf{r}_{\sigma_\ell}^k([\![v_\ell]\!])\|_{[L^2(\Omega)]^d} \, \|\mathbf{r}_{\sigma_\ell}^k([\![w_\ell]\!])\|_{[L^2(\Omega)]^d} \\
&\lesssim \sum_{\sigma \in \mathcal{F}_\ell} \eta_{\sigma_\ell} h_{\kappa_\ell,\kappa_\ell'}^{-1} \|[\![v_\ell]\!]\|_{L^2(\sigma)} \, \|[\![w_\ell]\!]\|_{L^2(\sigma)}
\end{aligned}$$

$$
\begin{aligned}
\widetilde{s}_0^{\mathrm{BR2}}(\mathcal{I}_\ell^0 v_\ell, \mathcal{I}_\ell^0 w_\ell) &= \sum_{\sigma\in\mathcal{F}_\ell}\sum_{\sigma\in\Sigma_0^\ell} \mathcal{H}_{\sigma_0}^{\sigma_\ell}\, \eta_{\sigma_0} \int_\Omega \mathbf{r}_{\sigma_0}^k([\![(\mathcal{I}_\ell^0 v_\ell)]\!])\cdot\mathbf{r}_{\sigma_0}^k([\![(\mathcal{I}_\ell^0 w_\ell)]\!]) \\
&\le \sum_{\sigma\in\mathcal{F}_\ell}\sum_{\sigma\in\Sigma_0^\ell} \frac{\eta_{\sigma_\ell}}{\eta_{\sigma_0}}\frac{h_{\kappa_0,\kappa_0'}}{h_{\kappa_\ell,\kappa_\ell'}}\eta_{\sigma_0}\|\mathbf{r}_{\sigma_0}^k([\![v_\ell]\!])\|_{[L^2(\Omega)]^d}\ \|\mathbf{r}_{\sigma_0}^k([\![w_\ell]\!])\|_{[L^2(\Omega)]^d} \\
&\lesssim \sum_{\sigma\in\mathcal{F}_\ell}\sum_{\sigma\in\Sigma_0^\ell} \eta_{\sigma_\ell}\frac{h_{\kappa_0,\kappa_0'}}{h_{\kappa_\ell,\kappa_\ell'}}h_{\kappa_0,\kappa_0'}^{-1}\|[\![v_\ell]\!]\|_{L^2(\sigma)}\ \|[\![w_\ell]\!]\|_{L^2(\sigma)} \\
&\le \sum_{\sigma\in\mathcal{F}_\ell} \eta_{\sigma_\ell} h_{\kappa_\ell,\kappa_\ell'}^{-1}\left(\sum_{\sigma\in\Sigma_0^\ell}\|[\![v_\ell]\!]\|_{L^2(\sigma)}^2\right)^{\frac12}\left(\sum_{\sigma\in\Sigma_0^\ell}\|[\![w_\ell]\!]\|_{L^2(\sigma)}^2\right)^{\frac12} \\
&= \sum_{\sigma\in\mathcal{F}_\ell} \eta_{\sigma_\ell} h_{\kappa_\ell,\kappa_\ell'}^{-1}\|[\![v_\ell]\!]\|_{L^2(\sigma)}\ \|[\![w_\ell]\!]\|_{L^2(\sigma)}
\end{aligned}
$$

which prove (64).

In view of (65), using (66), we now infer

$$
\begin{aligned}
\widetilde{s}_0^{\mathrm{BR2}}(\mathcal{I}_\ell^0 v_\ell, \mathcal{I}_\ell^0 v_\ell)|_{\sigma_\ell} &= \sum_{\sigma\in\Sigma_0^\ell} \eta_{\sigma_\ell}\frac{h_{\kappa_0,\kappa_0'}}{h_{\kappa_\ell,\kappa_\ell'}}\|\mathbf{r}_{\sigma_0}^k([\![v_\ell]\!])\|_{[L^2(\Omega)]^d}^2 \\
&\ge \sum_{\sigma\in\Sigma_0^\ell} \eta_{\sigma_\ell}\frac{h_{\kappa_0,\kappa_0'}}{h_{\kappa_\ell,\kappa_\ell'}}\frac{C_{\mathrm{r}_0}}{h_{\kappa_0,\kappa_0'}}\|[\![v_\ell]\!]\|_{L^2(\sigma_0)}^2 \\
&= \eta_{\sigma_\ell}\frac{C_{\mathrm{r}_0}}{h_{\kappa_\ell,\kappa_\ell'}}\|[\![v_\ell]\!]\|_{L^2(\sigma_\ell)}^2 \\
&\ge \eta_{\sigma_\ell}\frac{C_{\mathrm{r}_0}}{C_{\mathrm{tr}_\ell}}\|\mathbf{r}_{\sigma_\ell}^k([\![v_\ell]\!])\|_{[L^2(\Omega)]^d}^2. \\
&= \frac{C_{\mathrm{r}_0}}{C_{\mathrm{tr}_\ell}} s_\ell^{\mathrm{BR2}}(v_\ell, v_\ell)|_{\sigma_\ell} \qquad (68)
\end{aligned}
$$

and summing over mesh faces on level ℓ we get the desired result. As remarked by Antonietti *et al.* [13], C_{tr_ℓ} is influenced by the aspect ratio of the agglomerated element as well as by the ratio between the agglomerated element measure and the agglomerated face measure. Interestingly enough MGridGen algorithms are designed to optimize the aspect ratio of agglomerates and minimize the number of graph neighbors, which should also limit the occurrence of small degenerate faces (note that according to the definitions given in Section

2.1 the number of faces is equivalent to the number of element neighbors). Note that the theoretical analysis proposed in [13] also encompasses the limiting case of small degenerate faces.

Consider now the inherited coarse grid operators

$$A_\ell^{\widetilde{\mathcal{I}},\mathrm{BR2}} \overset{\mathrm{def}}{=} A_\ell^{\mathcal{I},\mathrm{BR2,CS}} + A_\ell^{\widetilde{\mathcal{I}},\mathrm{BR2,STB}} \tag{69}$$

such that $\forall v_\ell, w_\ell \in \mathbb{P}^k(\mathcal{T}_\ell)$, $\ell = 1, ..., L$

$$\text{Mod-Inherited:} \quad \begin{array}{ll} (A_\ell^{\mathcal{I},\mathrm{BR2,CS}} v_\ell, w_\ell)_{L^2(\Omega)} & \overset{\mathrm{def}}{=} a_0^{\mathrm{BR2,CS}}(\mathcal{I}_\ell^0 v_\ell, \mathcal{I}_\ell^0 w_\ell), \\ (A_\ell^{\widetilde{\mathcal{I}},\mathrm{BR2,STB}} v_\ell, w_\ell)_{L^2(\Omega)} & \overset{\mathrm{def}}{=} \widetilde{s}_0^{\mathrm{BR2}}(\mathcal{I}_\ell^0 v_\ell, \mathcal{I}_\ell^0 w_\ell). \end{array} \tag{70}$$

Since $a_0^{\mathrm{BR2,CS}}(\mathcal{I}_\ell^0 v_\ell, \mathcal{I}_\ell^0 w_\ell) + \widetilde{s}_0^{\mathrm{BR2}}(\mathcal{I}_\ell^0 v_\ell, \mathcal{I}_\ell^0 v_\ell) =$

$$= \int_\Omega \left|\nabla_\ell v_\ell - \mathbf{R}_\ell^k(v_\ell)\right|^2 + \sum_{\sigma\in\mathcal{F}_\ell}\sum_{\sigma\in\Sigma_0^\ell} \mathcal{H}_{\sigma_0}^{\sigma_\ell}\eta_{\sigma_0}\int_\Omega \left|\mathbf{r}_{\sigma_0}^k([\![v_\ell]\!])\right|^2 - \int_\Omega \left|\mathbf{R}_\ell^k(v_\ell)\right|^2$$

$$\geq \|\nabla_\ell v_\ell - \mathbf{R}_\ell^k(v_\ell)\|_{[L^2(\Omega)]^d}^2$$

$$+ \sum_{\sigma\in\mathcal{F}_\ell}\left(\eta_{\sigma_\ell}\frac{C_{\mathrm{r}_0}}{C_{\mathrm{tr}_\ell}}\|\mathbf{r}_{\sigma_\ell}^k([\![v_\ell]\!])\|_{[L^2(\Omega)]^d}^2 - \max_{\kappa\in\mathcal{T}_\sigma}(\mathrm{card}(\mathcal{F}_\kappa))\|\mathbf{r}_{\sigma_\ell}^k([\![v_\ell]\!])\|_{[L^2(\Omega)]^d}^2\right),$$

stability holds provided that $\eta_{\sigma_\ell}\frac{C_{\mathrm{r}_0}}{C_{\mathrm{tr}_l}} > \max_{\kappa\in\mathcal{T}_{\sigma_\ell}}(\mathrm{card}(\mathcal{F}_\kappa))$. In practice, motivated by the observation that the stabilization parameter choice suggested by theory is abundant, see *e.g.* [17], we deliberately neglected the dependence on C_{tr} and C_{r} in definition (62). Note that a strategy for estimating C_{tr} over agglomerated element meshes has been proposed by [18].

As we already pointed out the main advantage of inherited multigrid is the possibility to build coarse grid operator by means of intergrid transfer operators, avoiding numerical integration over agglomerated elements. The matrix restriction algorithm is described in Appendix A and exploit the possibility to

recursively inherit operators according to the following identities

$$(A_{\ell+1}^{\mathcal{I},\mathrm{BR2,CS}} v_{\ell+1}, w_{\ell+1})_{L^2(\Omega)} = (\mathcal{I}_\ell^{\ell+1} A_\ell^{\mathrm{BR2,CS}} \mathcal{I}_{\ell+1}^{\ell} v_{\ell+1}, w_{\ell+1})_{L^2(\Omega)} \tag{71}$$

$$\begin{aligned}&\sum_{\sigma\in\mathcal{F}_{\ell+1}} (A_{\ell+1}^{\widetilde{\mathcal{I}},\mathrm{BR2,STB}} v_{\ell+1}, w_{\ell+1})_{L^2(\kappa_{\ell+1}\cup\kappa'_{\ell+1})} \\ &\qquad = \sum_{\sigma\in\mathcal{F}_{\ell+1}} \sum_{\sigma\in\Sigma_\ell^{\ell+1}} (\mathcal{H}_{\sigma_\ell}^{\sigma_{\ell+1}} \mathcal{I}_\ell^{\ell+1} A_\ell^{\mathrm{BR2,STB}} \mathcal{I}_{\ell+1}^{\ell} v_{\ell+1}, w_{\ell+1})_{L^2(\kappa_\ell\cup\kappa'_\ell)}\end{aligned} \tag{72}$$

where $\mathcal{I}_\ell^{\ell+1}$ and $\mathcal{I}_{\ell+1}^{\ell}$ are the restriction and prolongation operators described in Section 4.2.

4.3.2. Stokes dG discretization

Consider the Stokes operator A_0^{Stk} defined in (34), the inherited coarse grid operators employed in this work read

$$A_\ell^{\widetilde{\mathcal{I}},\mathrm{Stk}} = \begin{bmatrix} A_\ell^{\widetilde{\mathcal{I}}}, & B_\ell^{\mathcal{I},t} \\ B_\ell^{\mathcal{I}} & C_\ell^{\mathcal{I}} \end{bmatrix}. \tag{73}$$

Consider the bilinear form $b_0(\mathbf{v}_0, q_0) : [\mathbb{P}^k(\mathcal{T}_0)]^d \times \mathbb{P}^k(\mathcal{T}_0) \to \mathbb{R}$ defined in (25), and the corresponding fine grid operator $B_0 : [\mathbb{P}^k(\mathcal{T}_0)]^d \to \mathbb{P}^k(\mathcal{T}_0)$, see definition (33). The coarse grid operators $B_\ell, B_\ell^{\mathcal{I}} : [\mathbb{P}^k(\mathcal{T}_\ell)]^d \to \mathbb{P}^k(\mathcal{T}_\ell),\ \ell = 1, ..., L$, read

$$\begin{aligned}&\text{Non-Inherited: } (B_\ell \mathbf{v}_\ell, q_\ell)_{L^2(\Omega)} \overset{\text{def}}{=} b_\ell(\mathbf{v}_\ell, q_\ell), && \forall (\mathbf{v}_\ell, q_\ell) \in [\mathbb{P}^k(\mathcal{T}_\ell)]^d \times \mathbb{P}^k(\mathcal{T}_\ell), \\ &\text{Inherited: } (B_\ell^{\mathcal{I}} \mathbf{v}_\ell, q_\ell)_{L^2(\Omega)} \overset{\text{def}}{=} b_0(\mathcal{I}_\ell^0 \mathbf{v}_\ell, \mathcal{I}_\ell^0 q_\ell), && \forall (\mathbf{v}_\ell, q_\ell) \in [\mathbb{P}^k(\mathcal{T}_\ell)]^d \times \mathbb{P}^k(\mathcal{T}_\ell),\end{aligned}$$

where the restriction of a vector function is performed componentwise

$$\mathcal{I}_\ell^0 \mathbf{v}_\ell = \sum_{i=0}^{d} \mathcal{I}_\ell^0 v_{\ell,i}.$$

Proceeding as in Section 4.3.1, see in particular (58), it is straightforward to show that $B_\ell = B_\ell^{\mathcal{I}}$.

According to definition (28) the operator $A_\ell^{\widetilde{\mathcal{I}}} : [\mathbb{P}^k(\mathcal{T}_\ell)]^d \to [\mathbb{P}^k(\mathcal{T}_\ell)]^d$, read

$$\text{Inherited:} \quad (A_\ell^{\widetilde{\mathcal{I}}}\mathbf{v}_\ell, \mathbf{w}_\ell)_{L^2(\Omega)} \stackrel{\text{def}}{=} \sum_{i=1}^{d} (A_\ell^{\widetilde{\mathcal{I}},\text{BR2}} v_{\ell,i}, w_{\ell,i})_{L^2(\Omega)}, \quad \forall \mathbf{v}_\ell, \mathbf{w}_\ell \in [\mathbb{P}^k(\mathcal{T}_\ell)]^d, \tag{74}$$

see (69) for the definition of $A_\ell^{\widetilde{\mathcal{I}},\text{BR2}}$.

To conclude, the coarse operators $C_\ell^{\mathcal{I}} : \mathbb{P}^k(\mathcal{T}_\ell) \to \mathbb{P}^k(\mathcal{T}_\ell)$, read

$$\text{Inherited:} \ (C_\ell^{\mathcal{I}} q_\ell, r_\ell)_{L^2(\Omega)} \stackrel{\text{def}}{=} c_0(\mathcal{I}_\ell^0 q_\ell, \mathcal{I}_\ell^0 r_\ell), \quad \forall (q_\ell, r_\ell) \in \mathbb{P}^k(\mathcal{T}_\ell) \times \mathbb{P}^k(\mathcal{T}_\ell),$$

see (27) for the definition of the bilinear form $c_0(q_0, r_0) : \mathbb{P}^k(\mathcal{T}_0) \times \mathbb{P}^k(\mathcal{T}_0) \to \mathbb{R}$. Even if the inherited bilinear form introduces a different (read smaller) amount of stabilization as compared to its non-inherited counterpart the numerical test case corroborate the choice not to modify the scaling of $C_\ell^{\mathcal{I}}$.

Coarse grid operators are built by means of intergrid transfer operators, exploiting the possibility to recursively inherit operators. For example the operators $B_\ell^{\mathcal{I}}$ are such that, for all $(\mathbf{v}_\ell, q_\ell) \in [\mathbb{P}^k(\mathcal{T}_\ell)]^d \times \mathbb{P}^k(\mathcal{T}_\ell)$

$$(B_{\ell+1}^{\mathcal{I}} \mathbf{v}_{\ell+1}, q_{\ell+1})_{L^2(\Omega)} = (\mathcal{I}_\ell^{\ell+1} B_\ell \mathcal{I}_{\ell+1}^\ell \mathbf{v}_{\ell+1}, q_{\ell+1})_{L^2(\Omega)}$$

where $\mathcal{I}_\ell^{\ell+1} : [\mathbb{P}^k(\mathcal{T}_\ell)]^d \to [\mathbb{P}^k(\mathcal{T}_{\ell+1})]^d$ and $\mathcal{I}_{\ell+1}^\ell : [\mathbb{P}^k(\mathcal{T}_{\ell+1})]^d \to [\mathbb{P}^k(\mathcal{T}_\ell)]^d$ and $\ell = 0, ..., L-1$. Similarly to vector restriction and prolongation, matrix restriction can be performed matrix-free without requiring to assemble the matrices $\mathbf{I}_\ell^{\ell+1}$ and $\mathbf{I}_{\ell+1}^\ell$. This practice yields large memory savings when the operators $\mathcal{I}_\ell^{\ell+1}, \mathcal{I}_{\ell+1}^\ell$ act on vector functions.

4.3.3. Navier-Stokes dG discretization

Consider the Navier-Stokes operator A_0^{INS} defined in (23), the inherited coarse grid operators employed in this work read

$$A_\ell^{\widetilde{\mathcal{I}},\text{INS}} = \begin{bmatrix} M_\ell^{\mathcal{I}} + J_\ell^{A,\widetilde{\mathcal{I}}} + J_\ell^{D,\mathcal{I}} & J_\ell^{B^t,\mathcal{I}} \\ J_\ell^{B,\mathcal{I}} & J_\ell^{C,\mathcal{I}} \end{bmatrix}. \tag{75}$$

To inherit the viscous operators we follow the same path of the Stokes case. Accordingly we get $J_\ell^{A,\widetilde{\mathcal{I}}} = A_\ell^{\widetilde{\mathcal{I}}}$, $\ell = 1, ...L$, see Definition (74) and note that, according to Definition (28), $J_0^A = A_0$.

Regarding inviscid operators we consider the trilinear form

$$j_0^{!\nu}(\mathbf{w}_0, \mathbf{u}_0, \mathbf{v}_0) = \sum_{i=1}^{d}\sum_{j=1}^{d} j_{i,j}^{!\nu}(\mathbf{w}_0, \delta w_{0,j}, k_{0,i}),$$

see Definition (18), such that

$$(J_0^D(\mathbf{w}_0)\delta\mathbf{u}_0, \mathbf{v}_0)_{[L^2(\Omega)]^d} = j_0^{!\nu}(\mathbf{w}_0, \mathbf{u}_0, \mathbf{v}_0), \quad \forall \delta\mathbf{u}_0, \mathbf{v}_0 \in [\mathbb{P}^k(\mathcal{T}_0)]^d.$$

We remark that operators J_0^B, $J_0^{B^t}$ and J_0^C can be restricted in a similar fashion. The coarse grid operators $J_\ell^D, J_\ell^{D,\mathcal{I}} : [\mathbb{P}^k(\mathcal{T}_\ell)]^d \to [\mathbb{P}^k(\mathcal{T}_\ell)]^d$, $\ell = 1, ..., L$, read

$$\text{Non-Inherited:} \qquad (J_\ell^D(\mathcal{I}_0^\ell \mathbf{w}_0)\mathbf{u}_\ell, \mathbf{v}_\ell)_{[L^2(\Omega)]^d} \stackrel{\text{def}}{=} j_\ell(\mathcal{I}_0^\ell \mathbf{w}_0, \mathbf{u}_\ell, \mathbf{v}_\ell), \tag{76}$$

$$\text{Inherited:} \qquad (J_\ell^{D,\mathcal{I}}(\mathbf{w}_0)\mathbf{u}_\ell, \mathbf{v}_\ell)_{[L^2(\Omega)]^d} \stackrel{\text{def}}{=} j_0(\mathbf{w}_0, \mathcal{I}_\ell^0 \mathbf{u}_\ell, \mathcal{I}_\ell^0 \mathbf{v}_\ell), \tag{77}$$

$\forall\, \mathbf{u}_\ell, \mathbf{v}_\ell \in [\mathbb{P}^k(\mathcal{T}_\ell)]^d, \mathbf{w}_0 \in [\mathbb{P}^k(\mathcal{T}_0)]^{d+1}$. The non-inherited version of coarse grid operators is not employed in this work but is included for the sake of comparison.

In practice, given the fine grid operator $J_0^D(\mathbf{w}_0)$, the inherited coarse grid operators $J_\ell^{D,\mathcal{I}}(\mathbf{w}_0)$ are defined recursively by means of the Galerkin projection. The operators $J_\ell^{D,\mathcal{I}}(\mathbf{w}_0)$ are such that, for all $\mathbf{u}_\ell, \mathbf{v}_\ell \in [\mathbb{P}^k(\mathcal{T}_\ell)]^d, \mathbf{w}_0 \in$

$[\mathbb{P}^k(\mathcal{T}_0)]^{d+1}$

$$(J^{D,\mathcal{I}}_{\ell+1}(\mathbf{w}_0)\mathbf{u}_{\ell+1},\mathbf{v}_{\ell+1})_{[L^2(\Omega)]^d} = (\mathcal{I}^{\ell+1}_{\ell} J^{D,\mathcal{I}}_{\ell}(\mathbf{w}_0)\mathcal{I}^{\ell}_{\ell+1}\mathbf{u}_{\ell+1},\mathbf{v}_{\ell+1})_{[L^2(\Omega)]^d}.$$

Accordingly

$$\text{Galerkin projection: } J^{\mathcal{I}}_{\ell+1}(\mathbf{w}_0) \stackrel{\text{def}}{=} \mathcal{I}^{\ell+1}_{\ell}\ J^{\mathcal{I}}_{\ell}(\mathbf{w}_0)\ \mathcal{I}^{\ell}_{\ell+1}, \qquad \ell = 0, ..., L-1, \quad (78)$$

where $\mathcal{I}^{\ell+1}_{\ell}$ and $\mathcal{I}^{\ell}_{\ell+1}$ are the restriction and prolongation operators described in Section 4.2 and (78) is performed matrix-free.

5. Multigrid and Block preconditioners

In this work we consider multigrid preconditioners for the Navier-Stokes equations and block preconditioners for the dG discretization of the Stokes problem (34). Both the Stokes and the Navier-Stokes problem have a block structure that can be exploited to devise preconditioners based on Schur complement decompositions, nevertheless pressure Schur complement preconditioners are less trivial in the Navier-Stokes case than in the Stokes case [1]. Comparison between block and h-multigrid preconditioners will be performed on a Stokes model problem, while in the Navier-Stokes case we will focus on h-multigrid as a preconditioner of a FGMRES backward Euler iteration.

Incompressible flow problem dG discretizations in the form (43) can be solved by preconditioned Krylov iterative methods, say $ksp(A_0, \widehat{A}_0)$, where the preconditioner $\widehat{A}_0$ is a suitable approximation of A_0 (such that the application of $\widehat{A}_0^{-1}$ to a vector is cheap to compute). For example an Incomplete Lower Upper (ILU) decomposition of the system matrix is a common preconditioner choice, *i.e.* $ksp\,(A_0, \mathrm{ILU}(A_0))$. Interestingly a Krylov iterative method, say $\widehat{ksp}(A_0)$, can serve as a preconditioner by triggering convergence of the iteration on loose

tolerances, *i.e.* $ksp\left(A_0, \widehat{ksp}(A_0)\right)$. Note that in this case Flexible Generalized Minimal RESidual (FGMRES) is usually employed as a solver as the preconditioner varies at each outer Krylov iteration.

Similarly, the multigrid V-cycle iteration of Section 4.1 can be employed as preconditioner, thus the solver strategy reads: $\mathrm{FGMRES}\left(A_0, \mathrm{MG}_{\mathcal{V}}(A_0)\right)$. Building the coarse grid operators $A_\ell,\ \ell = 1, ..., L$ as described in Section 4.3, the multigrid V-cycle $\mathrm{MG}_{\mathcal{V}}(A_0)$ can be applied as a preconditioner of the Stokes and Navier-Stokes operators A_0^{Stk}, A_0^{INS}.

Besides multigrid preconditioners, block preconditioners for the Stokes problem (34) are derived by noticing that A^{Stk} admits the following LDU factorization

$$A_0^{\mathrm{Stk}} = \begin{bmatrix} I & 0 \\ B_0 A_0^{-1} & I \end{bmatrix} \begin{bmatrix} A_0 & 0 \\ 0 & S_0 \end{bmatrix} \begin{bmatrix} I & A_0^{-1} B_0^t \\ 0 & I \end{bmatrix}, \tag{79}$$

where

$$S_0 = C_0 - B_0 A_0^{-1} B_0^t, \tag{80}$$

is the *pressure Schur complement* matrix. Since

$$\left(A_0^{\mathrm{Stk}}\right)^{-1} = \begin{bmatrix} I & -A_0^{-1} B_0^t \\ 0 & I \end{bmatrix} \begin{bmatrix} A_0^{-1} & 0 \\ 0 & S_0^{-1} \end{bmatrix} \begin{bmatrix} I & 0 \\ -B_0 A_0^{-1} & I \end{bmatrix}, \tag{81}$$

$\left(\widehat{A}_0^{\mathrm{Stk}}\right)^{-1}$ can be obtained by replacing A_0^{-1} and S_0^{-1} with preconditioned Krylov solvers, say $ksp(A_0, \widehat{A}_0)$ and $ksp(S_0, \widehat{S}_0)$. Whereas computing S_0 explicitly is not viable, an approximate solver $ksp(\widetilde{S}_0, \widehat{S}_0)$ can be employed with

$$\widetilde{S}_0 = C_0 - B_0\ ksp(A_0, \widehat{A}_0)\ B_0^t.$$

Note that applying $\widetilde{S}_0$ to a vector involves a nested Krylov iteration. Clearly the performance of the outer solver $ksp(A_0^{\mathrm{Stk}}, \widehat{A}_0^{\mathrm{Stk}})$ is strongly influenced by

the availability of good preconditioners for the Laplace and the pressure Schur complement operators, read $\widehat{A}_0$ and $\widehat{S}_0$.

As suggested by Shahbazi *et al.* [36] , $\widehat{S}_0$ can be constructed by a dG discretization of the Laplace operator with homogeneous Neumann boundary conditions on Dirichlet boundaries. Accordingly the operator $\widehat{S}_0$ is such that

$$(\widehat{S}_0 v_0, w_0)_{L^2(\Omega)} = -a_0^{\mathrm{BR2,hN}}(v_0, w_0),\ \forall v_0, w_0 \in \mathbb{P}^k(\mathcal{T}_0) \tag{82}$$

where

$$\begin{aligned} a_0^{\mathrm{BR2,hN}}(v_0, w_0) = &\sum_{\kappa\in\mathcal{T}_0}\int_\kappa \nabla_0 v_0 \nabla_0 w_0 + \sum_{\sigma\in\mathcal{F}_0^{\mathrm{i}}}\eta_\sigma \int_\sigma \{\!\{\mathbf{r}_\sigma^k([\![v_0]\!])\}\!\}\cdot\mathbf{n}_\sigma\, [\![w_0]\!] \\ &- \sum_{\sigma\in\mathcal{F}_0^{\mathrm{i}}}\int_\sigma \left(\{\!\{\nabla_0 v_0\}\!\}\cdot\mathbf{n}_\sigma\, [\![w_0]\!] + [\![v_0]\!]\, \{\!\{\nabla_0 w_0\}\!\}\cdot\mathbf{n}_\sigma\right). \end{aligned} \tag{83}$$

Note that (83) can be obtained from the BR2 bilinear form in (36) using the local and global lifting operator definitions (9) and (10) and dropping the boundary face terms. As a preconditioner for A_0 we employ the h-multigrid V-Cycle iteration described in Section 4 using the rescaled-inherited version of coarse grid operators defined in (74).

The solver and preconditioners options are summarized in what follows. Richardson iteration serves as the outer loop, *i.e.* RCHRD($A_0^{\mathrm{Stk}}, \widehat{A}_0^{\mathrm{Stk}}$). The application of the block preconditioner reads: $\left(\widehat{A}_0^{Stk}\right)^{-1} =$

$$\begin{bmatrix} ksp(A_0, \widehat{A}_0) & 0 \\ 0 & I \end{bmatrix} \begin{bmatrix} I & -B_0^t \\ 0 & I \end{bmatrix} \begin{bmatrix} I & 0 \\ 0 & ksp(\widetilde{S}_0, \widehat{S}_0) \end{bmatrix} \begin{bmatrix} I & 0 \\ -B_0 ksp(A_0, \widehat{A}_0) & I \end{bmatrix}, \tag{84}$$

see PETSc User manual [37], where

$$ksp(A_0, \widehat{A}_0) = \text{FGMRES}\left(A_0, \text{MG}_{\mathcal{V}}(A_0)\right),$$
$$ksp(\widetilde{S}_0, \widehat{S}_0) = \text{GMRES}\left(\widetilde{S}_0, \text{ILU}(\widehat{S}_0)\right),$$

and $\widetilde{S}_0 = C_0 - B_0\ ksp(A_0, \widehat{A}_0)\ B_0^t$.

6. Numerical results

6.1. BR2 dG discretization

In this section we apply the h-multigrid V-cycle iteration of Algorithm 2 for solving a Poisson problem discretized by means of the BR2 dG formulation. For the sake of comparison we consider the three strategies for defining coarse grid operators introduced in Section 4.3.1, that is: i) non-inherited operators defined assembling bilinear forms on each mesh level ii) inherited operators defined by means of a Galerkin projection iii) the newly introduced inherited operators with stability rescaling. We compare these approaches on the basis of convergence rate and computation time and we assess the benefits of using h-multigrid as compared to state-of-the-art single grid solvers like the preconditioned Conjugate-Gradient (CG) method and the preconditioned Generalized Minimal RESidual (GMRES) method.

We consider the Poisson problem in (35) on the bi-unit square and cube, $\Omega = [-1, 1]^d$ with $d = 2$ and $d = 3$, respectively, where the forcing term is imposed according to the following smooth analytical solution

$$u = \prod_{i=1}^{d} \sin(\pi x_i), \tag{85}$$

and homogeneous boundary conditions are imposed on $\partial\Omega$.

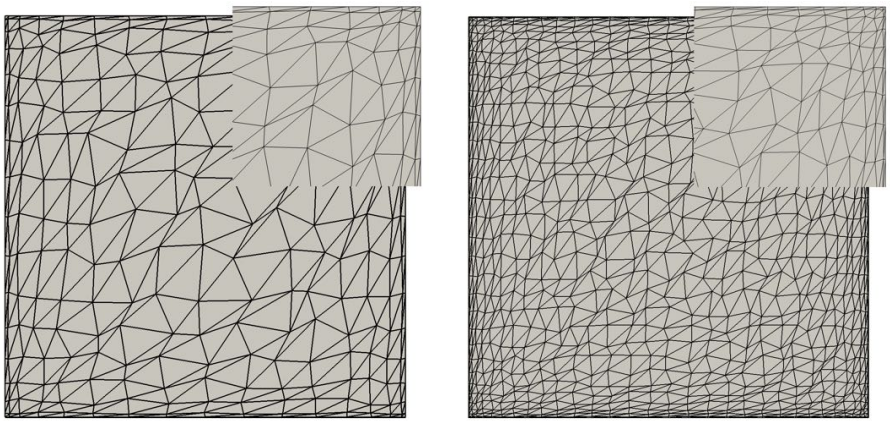

Figure 2: Two grids of the distorted and graded triangular mesh sequence. *Left*: $2\,(32^2)$ grid and *right*: $2\,(64^2)$ grid. The square corner detail allows to appreciate that the aspect ratio of triangular elements increases moving towards the domain boundaries.

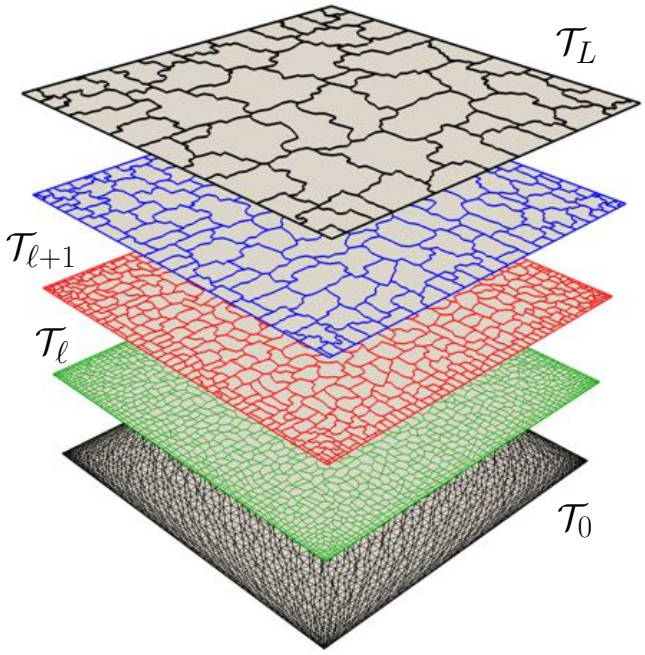

Figure 3: Example of a five levels ($L = 4$) h-coarsened mesh sequence generated on top the $2(64^2)$ triangular elements grid of Figure 2.

In order to investigate the growth of computational costs while increasing

the mesh size, 2D solutions are computed on three uniform quadrilateral elements meshes of size $(128 \cdot 2^n)^2, n = \{0, 1, 2\}$ and three distorted and graded triangular meshes of size $2(64 \cdot 2^n)^2, n = \{0, 1, 2\}$, see Figure 2. As for 3D solutions we consider a 128^3 grid, counting of more than two million hexahedral elements, and we investigate parallel performance of the multigrid algorithm running on up to 128 processes. In both two and three space dimensions we check the influence of raising the polynomial degree on the convergence rate and the computational expense considering $k = \{1, 2, 3\}$, that is first, second and third polynomial degree dG discretizations. The L^2 error norm is on the order of 10^{-11} for the fourth-order accurate dG discretization on the 512^2 quadrilateral grid. We do not consider a further raise of the polynomial degree since for higher-order discretizations p-multigrid or hp-multigrid solution strategies might be best suited. Indeed h-multigrid is to be applied in the context of large scale computations where the mesh size is constrained by the need to accurately discretize a complex computational domain, note that the design of coarse high-order meshes suited for higher-order discretizations is an open field of research, see *e.g.* [38]. We remark that the agglomeration strategy does not take advantage of the triviality of the geometry here considered. In Section 6.3.3 the multigrid strategy will be applied without any modification to unstructured, possibly hybrid, meshes of complex computational domains.

To investigate the influence of the number of coarse levels on the convergence rate we consider $L = \{2, 3, 4, 5\}$ and $L = \{2, 3, 4\}$ for $d = 2$ and $d = 3$, respectively, that is we consider a stack of 3 to 6 grids in two space dimensions and a stack of 3 to 5 grids in three space dimensions, see Figure 1 and Figure 3. The number of mesh elements at each level ℓ and the maximum and the minimum number of elements among the distributed grid partitions at level ℓ is reported in Table (1) and Table (2), respectively. It is interesting to remark

h-coarsened quadrilateral mesh sequences

$\text{card}(\mathcal{T}_0)$	$\text{card}(\mathcal{T}_\ell)$				
	$\ell = 1$	$\ell = 2$	$\ell = 3$	$\ell = 4$	$\ell = 5$
64^2	1208	365	109	34	11
128^2	4824	1447	437	136	41
256^2	19218	5791	1754	535	161
512^2	76880	23087	6976	2116	643

h-coarsened triangular mesh sequences

$\text{card}(\mathcal{T}_0)$	$\text{card}(\mathcal{T}_\ell)$				
	$\ell = 1$	$\ell = 2$	$\ell = 3$	$\ell = 4$	$\ell = 5$
$2(32^2)$	541	157	46	13	4
$2(64^2)$	2290	660	194	57	17
$2(128^2)$	9287	2683	780	231	66
$2(256^2)$	37551	10956	3214	935	274

Table 1: Six levels h-coarsened agglomerated elements mesh sequences of the bi-unit square. Number of agglomerated elements at each mesh level $\ell = 0, ..., 5$.

3D h-coarsened mesh sequences, grid partition size

$\text{card}(\mathcal{T}_0)$	processes (np)	$\max_{i=1,...,np}$ / $\min_{i=1,...,np}$ $\text{card}(\mathcal{T}_\ell^i)$				
		$\ell = 1$	$\ell = 2$	$\ell = 3$	$\ell = 4$	$\ell = 5$
128^3	16	131078	19378	2895	434	64
		131066	19258	2846	416	60
128^3	32	65541	9763	1468	219	34
		65531	9666	1426	207	28
128^3	64	32770	4878	734	111	17
		32766	4809	708	104	14
128^3	128	16392	2454	372	57	9
		16380	2407	353	50	7

Table 2: Five levels ($\ell = 1, ..., 5$) h-coarsened mesh sequences agglomerated on top of a 128^3 hexahedral elements grids of the bi-unit cube. Maximum and minimum number of elements among the distributed grid partitions $\mathcal{T}_\ell^i,\ i = 1, ..., np$, obtained running in parallel with np processes.

that in three (resp. two) space dimensions a 6.7 (resp. 3.3) fold decrease of the number of elements is obtained at each agglomeration step, whereas an 8 (resp. 4) fold decrease would be required in order to halve the mesh step size in uniform hexahedral (resp. quadrilateral) elements mesh sequences. Nevertheless, since agglomerated elements have very general shapes, the element size h_κ is not uniform and, as demonstrated in [16], the maximum and average mesh step size are usually bigger as compared to quadrilateral elements meshes of the same cardinality. Accodingly, even if the agglomerated mesh cardinality is suboptimal, the optimal coarsening steepness is closely approached, that is $\frac{h_{\kappa_{\ell+1}}}{h_{\kappa_\ell}} \lessapprox 2$ for each $\kappa_\ell \in \mathcal{T}_\ell$.

We complete the definition of the V-cycle preconditioned FGMRES iteration, *i.e.* FGMRES $\left(A_0^{\mathrm{BR2}}, \mathrm{MG}_{\mathcal{V}}(A_0^{\mathrm{BR2}})\right)$ specifying the relevant solver and preconditioner options. We employ a single iteration of multigrid V-cycle as a preconditioner for a Flexible GMRES solver (restarted FGMRES with 60 Krylov spaces) [39]. High-order modes of the error are smoothed with a single iteration of a right preconditioned GMRES solver. In 2D serial runs we employ an Incomplete Lower-Upper (ILU) preconditioner while in 3D parallel runs we opt for an Additive Schwarz domain decomposition Method (ASM) with one level of overlap between sub-domains and an ILU decomposition for each sub-domain matrix. On the coarsest level L linear systems are solved with a direct solver in 2D. For parallel 3D runs we rely on the solver employed in smoothing steps but, instead of a single iteration, we impose a four order of magnitude decrease of the relative residual norm, that is $\frac{\|f_L - A_L^{\mathrm{BR2}} \bar{u}_L\|}{\|f_L\|} \leq 10^{-4}$. The FGMRES solver is forced to reach tight relative residual tolerance, in particular the linear system solution converges in N_{it} iterations if at the i-th iterate $\|\hat{r}_0^i\| = \frac{\|f_0 - A_0^{\mathrm{BR2}} \bar{u}_0^i\|}{\|f_0\|} \leq 10^{-10}$.

The numerical results reported in the next section have been computed by exploiting the PCMG multigrid preconditioner framework available in the `PETSc`

library [40, 37, 41]. The `MOAB` [42] library is employed for storing distributed mesh data at all mesh levels and `METIS` [43] library is employed to partition the mesh in case of parallel computations.

6.1.1. 2D Poisson problem

Linear solver iterations, 2D Poisson problem, quadrilateral mesh sequences

operators	inherited			non-inherited			rescaled-inherited		
grid (card $(\mathcal{T}_0) = (\cdot)^2$)	128	256	512	128	256	512	128	256	512
k = 1									
FGMRES MG$_\mathcal{V}$ $L = 2$	12	12	13	10	10	10	10	10	10
FGMRES MG$_\mathcal{V}$ $L = 3$	15	15	15	10	10	10	10	11	11
FGMRES MG$_\mathcal{V}$ $L = 4$	19	19	20	10	10	10	10	11	11
FGMRES MG$_\mathcal{V}$ $L = 5$	24	24	25	10	10	10	11	11	11
CG ILU(0)							206	398	743
GMRES(120) ILU(0)							229	566	1255
k = 2									
FGMRES MG$_\mathcal{V}$ $L = 2$	9	9	9	8	8	8	8	8	8
FGMRES MG$_\mathcal{V}$ $L = 3$	16	15	15	8	8	8	8	8	8
FGMRES MG$_\mathcal{V}$ $L = 4$	27	27	26	9	8	8	8	8	8
FGMRES MG$_\mathcal{V}$ $L = 5$	60	45	49	9	9	8	9	9	8
CG ILU(0)							230	446	827
GMRES(120) ILU(0)							312	811	1925
k = 3									
FGMRES MG$_\mathcal{V}$ $L = 2$	8	7	7	7	7	6	7	6	6
FGMRES MG$_\mathcal{V}$ $L = 3$	12	12	13	7	7	6	7	7	6
FGMRES MG$_\mathcal{V}$ $L = 4$	20	21	23	8	7	7	8	7	7
FGMRES MG$_\mathcal{V}$ $L = 5$	37	39	42	8	8	7	8	8	7
CG ILU(0)							250	480	899
GMRES(120) ILU(0)							310	1015	2000*

Table 3: Number of iterations required to solve the 2D model Poisson problem (35)-(85) with an FGMRES solver preconditioned with h-multigrid. Quadrilateral mesh sequence. Linear system relative residual tolerance is 10^{-10}, see text for details.

Tables 3 and 4 report the number of iterations required to solve the Poisson problem (35)-(85) discretized with the BR2 method. Single grid solver options mimic multigrid ones: we impose a relative residual decrease of 10^{-10} and employ ILU (right) preconditioned Conjugate Gradient (CG) and GMRES solvers setting the number of Krylov spaces to 120 for GMRES.

As expected only non-inherited and inherited multigrid with stabilization term rescaling (*rescaled-inherited*) are able to provide uniform convergence with

Linear solver iterations, 2D Poisson problem, triangular mesh sequences

operators	inherited			non-inherited			rescaled-inherited		
grid (card $(\mathcal{T}_0) = 2(\cdot)^2$)	64	128	256	64	128	256	64	128	256
k = 1									
FGMRES MG_ν $L = 2$	19	22	28	24	29	37	19	23	30
FGMRES MG_ν $L = 3$	20	23	28	24	29	38	19	24	30
FGMRES MG_ν $L = 4$	22	24	29	24	29	38	19	23	30
FGMRES MG_ν $L = 5$	26	29	32	24	29	38	19	23	30
CG ILU(0)							250	509	1011
GMRES(120) ILU(0)							298	677	1520
k = 2									
FGMRES MG_ν $L = 2$	17	19	24	21	26	37	17	20	27
FGMRES MG_ν $L = 3$	20	22	26	21	26	37	17	21	27
FGMRES MG_ν $L = 4$	26	32	37	21	26	38	17	21	27
FGMRES MG_ν $L = 5$	44	61	77	21	27	38	17	21	27
CG ILU(0)							297	599	1197
GMRES(120) ILU(0)							330	866	2000*
k = 3									
FGMRES MG_ν $L = 2$	16	17	20	17	20	28	15	18	21
FGMRES MG_ν $L = 3$	18	19	23	17	21	29	16	18	21
FGMRES MG_ν $L = 4$	25	25	31	18	21	29	16	18	22
FGMRES MG_ν $L = 5$	71	43	67	18	22	29	17	19	22
CG ILU(0)							334	664	1324
GMRES(120) ILU(0)							396	911	2000*

Table 4: Number of iterations required to solve the 2D model Poisson problem (35)-(85) with an FGMRES solver preconditioned with h-multigrid. Distorted triangular mesh sequence. Linear system relative residual tolerance is 10^{-10}, see text for details.

respect to the number of levels, note that this is the case even on bad quality triangular meshes. Comparison of the number of iterations on quadrilateral elements meshes highlights that the performance of single grid solvers worsen on finer meshes while the rescaled-inherited multigrid iteration is almost grid independent. Moving towards finer distorted triangular meshes also the number of multigrid iterations increases, but far less dramatically than with single grid solvers. Interestingly the number of iterations of single grid solvers is also affected by raising the polynomial degree while the convergence rates of h-multigrid improve increasing the polynomial degree when non-inherited and rescaled-inherited multigrid are employed. Note that for $k = 2$ and $k = 3$ the average residual decrease exceeds one order of magnitude at each V-cycle iteration on quadrilateral mesh sequences.

Total CPU time (s), 2D Poisson problem, quadrilateral mesh sequences

operators	inherited			non-inherited			rescaled-inherited		
grid ($\text{card}(\mathcal{T}_0) = (\cdot)^2$)	128	256	512	128	256	512	128	256	512
k = 1									
FGMRES MG$_\mathcal{V}$ $L = 2$	1.58	**5.95**	**27.4**	2.03	**7.23**	**31.7**	1.49	**5.70**	**24.5**
FGMRES MG$_\mathcal{V}$ $L = 3$	1.72	6.70	28.1	2.10	7.89	32.3	1.69	6.11	25.9
FGMRES MG$_\mathcal{V}$ $L = 4$	1.92	7.78	31.2	2.27	8.61	34.6	1.60	6.29	25.1
FGMRES MG$_\mathcal{V}$ $L = 5$	2.14	8.32	36.0	2.67	8.95	36.9	1.57	6.69	26.3
CG ILU(0)							**1.45**	8.01	48.7
GMRES(120) ILU(0)							3.01	28.1	230
k = 2									
FGMRES MG$_\mathcal{V}$ $L = 2$	**2.75**	**11.3**	54.4	**3.30**	**13.6**	64.4	**2.43**	10.3	51.1
FGMRES MG$_\mathcal{V}$ $L = 3$	3.49	12.0	**54.2**	3.34	14.1	58.3	**2.43**	9.4	42.1
FGMRES MG$_\mathcal{V}$ $L = 4$	4.12	16.7	73.9	3.36	13.8	**57.9**	**2.43**	**9.3**	**41.4**
FGMRES MG$_\mathcal{V}$ $L = 5$	7.18	24.3	115	3.74	15.6	60.3	**2.43**	9.7	41.7
CG ILU(0)							3.64	22.1	143
GMRES(120) ILU(0)							7.90	86.7	811
k = 3									
FGMRES MG$_\mathcal{V}$ $L = 2$	6.62	30.3	195	8.40	37.5	226	6.65	29.4	191
FGMRES MG$_\mathcal{V}$ $L = 3$	**6.03**	**26.1**	**126**	**7.46**	**30.0**	133	**5.07**	22.0	96.2
FGMRES MG$_\mathcal{V}$ $L = 4$	7.91	34.6	160	7.59	32.1	**129**	5.30	**20.9**	88.9
FGMRES MG$_\mathcal{V}$ $L = 5$	11.9	52.5	242	8.17	33.4	134	5.38	21.8	**87.7**
CG ILU(0)							11.0	68.4	475
GMRES(120) ILU(0)							19.3	240	1851*

Table 5: BR2 dG discretization of the 2D model Poisson problem (35)-(85). Total CPU times (assembly plus solution times) required to reach a relative residual tolerance of 10^{-10}, see text for details. Bold text highlights the best result column-wise.

$MG_{\mathcal{V}}(k)$ vs $CG(k)$	Total CPU time speedup								
degree	$k=1$			$k=2$			$k=3$		
quad grid	128	256	512	128	256	512	128	256	512
CG/MG step time	0.92	1.2	1.8	1.5	2.3	3.4	2.0	3.1	5.4
tri grid	64	128	256	64	128	256	64	128	256
CG/MG step time	0.95	1.6	2.4	1.4	2.3	3.5	1.7	3.1	5.0

Table 6: BR2 dG discretization of the 2D model Poisson problem (35)-(85). Comparison of total CPU times (assembly plus solution times) required to reach a relative residual tolerance of 10^{-10} with Conjugate Gradient and h-multigrid ($L = 5$), see text for details.

Total CPU times (total means the sum of solution and assembly CPU times) reported in Table 5 demonstrate that the newly introduced rescaled-inherited multigrid strategy is the best performing. From the solution time viewpoint inherited and rescaled-inherited multigrid are almost indistinguishable while inherited multigrid is largely affected by the performance degradation increasing the number of grid levels. From the assembly time viewpoint inherited and rescaled-inherited multigrid avoids the burden of numerically integrating bilinear forms over agglomerated elements meshes. Non-inherited multigrid assembly times strongly increase with the number of levels (note that assembly can be twice as expensive than solution) negatively impacting total CPU times. We remark that, if quadrature formulas are defined over sub-elements as described in (8), the number of quadrature points is the same at each mesh level, irrespectively of the mesh density.

Rescaled-inherited total CPU times are almost independent of the number of levels provided that the coarsest grid on level L is coarse enough. This is a very important result in view of applying multigrid in real-world computations since it basically removes the burden of choosing of the number of grid levels. Most importantly both non-inherited and rescaled-inherited multigrid are close to the optimal multigrid efficiency. They lead to a four-fold increase of the total computation time with a four-fold increase of the mesh size at all the polynomials degrees, provided that L is chosen large enough.

$CG(k_{CG})$ vs $MG_{\mathcal{V}}(k_{MG})$	Total CPU time ratio					
solver(degree)	$CG(1)/MG_{\mathcal{V}}(2)$		$CG(2)/MG_{\mathcal{V}}(3)$		$CG(1)/MG_{\mathcal{V}}(3)$	
finest grid	quad	tri	quads	tri	quads	tri
CG/MG step time	0.95	1.0	1.6	1.3	0.55	0.37

Table 7: BR2 dG discretization of the 2D model Poisson problem (35)-(85). Comparison of total CPU times (assembly plus solution times) required to reach a relative residual tolerance of 10^{-10} with Conjugate Gradient and h-multigrid ($L = 5$). The multigrid solution strategy is applied to an higher polynomial degree dG discretization ($k_{MG} > k_{CG}$).

Since CPU times for the 2D Poisson problem are measured running on a 2010 laptop, we recommend not to consider absolute values but rather relative gains. In this regard, the gains with respect to the best performing single grid solver (an ILU preconditioned Conjugate Gradient iteration) are significant, especially at the highest polynomial degrees on fine meshes, see Table 6. As a result, if we consider the finest quadrilateral and triangular meshes, solving a second degree polynomial degree BR2 dG discretization with h-multigrid is comparable to solving a first degree dG discretization with CG, see Table 7. Interestingly the time required for solving a third polynomial degree BR2 dG discretization with h-multigrid is twice the time required for solving a first degree dG discretization with Conjugate Gradient, which is quite impressive considering the accuracy gap.

Preprocessing CPU time (s), quadrilateral elements mesh sequence

task	grid topology			orthogonalization and intergrid operators								
				k=1			k=2			k=3		
grid	128	256	512	128	256	512	128	256	512	128	256	512
$L = 2$	0.72	3.28	22.1	0.17	0.64	2.79	0.34	1.37	5.36	0.72	2.63	10.5
$L = 3$	0.75	3.84	25.1	0.21	0.80	4.01	0.45	1.68	6.82	0.97	3.72	15.1
$L = 4$	0.82	4.22	26.8	0.25	1.00	4.68	0.58	2.11	8.24	1.26	4.76	19.3
$L = 5$	0.88	4.65	28.1	0.29	1.19	5.46	0.67	2.53	10.4	1.57	5.95	24.2
$L = 0$	0.09	0.29	1.13	0.03	0.10	0.38	0.07	0.22	0.83	0.13	0.42	1.77

Table 8: Preprocessing phases required for h-multigrid computations as compared to single grid computations ($L = 0$). CPU times for generation of h-coarsened mesh sequences (grid topology computation task), orthogonalization of shape functions and computation of intergrid operators, see text for details.

To conclude we also report preprocessing CPU times including generation of h-coarsened mesh sequence and orthogonalization of shape functions together

with computation and storage of intergrid transfer operators. In Table 8 it is possible to appreciate that both these operations are time consuming as compared to setup times of single grid computations. Nevertheless for $k > 1$ dG discretizations, even considering preprocessing times, multigrid outperforms single grid solvers.

6.1.2. 3D Poisson problem

The Poisson problem in three space dimensions is here considered to assess the performance of the h-multigrid solution strategy in parallel computations. The 128^3 hexahedral mesh is first partitioned and distributed across the processes, thus each process build an h-coarsened mesh sequence of its own partition. This practice is optimal from the distribution of computational load viewpoint but requires ad hoc strategies to deal with agglomerated elements whose faces are shared between partitions.

While in single grid dG solvers the replication of a single layer of cells (the so called *ghost cells*) across partition boundaries ensures that the *stencil* of the discretization is fully accessible, the definition of ghost agglomerated cells involves many layers of cells of the fine grid. Note that the exact amount of layers depends on the grid level and the shape of agglomerated elements. Interestingly, only non-inherited multigrid requires to evaluate basis functions of ghost agglomerated elements at partition boundaries, while inherithed and rescaled-inherited multigrid rely on integrid transfer operators associated to ghost cells. In this regard the design decision of storing intergrid transfer operators in preprocessing is very handy, see Section 4.2. Indeed, intergrid transfer operators associated to ghost cells can be communicated across partitions without needing to actually build ghost cells, only adjacencies informations are required. As a consequence the implementation of inherited and rescaled-inherited multigrid is simpler in parallel.

Linear solver iterations, 3D Poisson problem in parallel

k	1			2			3		
processes	8	16	32	16	32	64	32	64	128
FGMRES MG$_\mathcal{V}$ $L=2$	11	11	11	10	10	10	9	9	10
FGMRES MG$_\mathcal{V}$ $L=3$	12	12	12	11	11	11	10	10	11
FGMRES MG$_\mathcal{V}$ $L=4$	12	12	12	11	11	11	10	11	11
CG BJACOBI(ILU)	488	509	530	680	708	687	567	856	898
GMRES ASM(1,ILU)	432	467	455	675	699	614	868	794	745

Table 9: BR2 dG discretization of the 3D model Poisson problem (35)-(85). Number of iterations required by a h-multigrid preconditioned FMGMRES solver and block-ILU preconditioned single grid solvers, see text for details. Linear system relative residual tolerance is 10^{-10}.

We consider the rescaled-inherited strategy for defining coarse grid operators which has demonstrated to provide uniform convergence with respect to the number of levels and affordable assembly times, see Section 6.1.1. Beside the baseline computations performed with 8, 16 and 32 processes at first, second and third polynomial degree, respectively, we double the number of processes two times for a total of three runs at each polynomial degree. Thanks to the use of an ASM preconditioner for the GMRES smoother the number of FGMRES iterations is independent from the number of processes, see Table 9. The single grid GMRES solver uses the same kind of ASM preconditioner employed by the GMRES smoothers, that is an ASM preconditioner with one level of overlap between the sub-domains and an ILU decomposition in each sub-domain matrix. The single grid CG solver employs a block-Jacobi preconditioner with an ILU decomposition in each sub-domain matrix (but no overlap between sub-domains).

The CPU times reported in Table 10 demonstrate that the h-multigrid efficiency does not deteriorate increasing the number of processes. The gains with respect to single grid solvers are comparable to those observed in serial computations in two space dimensions, see Section 6.1.1. Similarly, a sufficient number of grid levels must be employed to ensure that the grid on level L is coarse enough, note the poor performance for $L = 2$. Even if a scalability

CPU time (s)	solution			assembly			total		
k = 1									
processes	8	16	32	8	16	32	8	16	32
FGMRES MG$_\mathcal{V}$ $L = 2$	12.2	5.94	5.11	28.3	13.2	6.26	40.5	19.2	11.4
FGMRES MG$_\mathcal{V}$ $L = 3$	11.6	5.85	3.12	28.8	12.8	6.72	40.4	18.7	9.84
FGMRES MG$_\mathcal{V}$ $L = 4$	11.6	5.82	3.26	28.8	12.9	6.27	37.4	18.7	9.53
CG BJACOBI(ILU)	55.7	29.3	15.1	25.6	12.8	6.71	88.4	42.2	21.8
GMRES(120) ASM(1,ILU)	144	80.8	41.3	25.9	12.9	6.74	170	93.8	48.0
k = 2									
processes	16	32	64	16	32	64	16	32	64
FGMRES MG$_\mathcal{V}$ $L = 2$	32.7	26.1	17.9	45.6	23.3	11.8	78.3	49.3	29.8
FGMRES MG$_\mathcal{V}$ $L = 3$	26.4	14.3	7.95	47.2	24.1	11.8	73.6	38.4	19.7
FGMRES MG$_\mathcal{V}$ $L = 4$	25.8	14.2	7.92	48.2	24.4	12.5	74.0	38.6	20.4
CG BJACOBI(ILU)	224	109	54.6	45.4	22.9	11.4	269	132	66.1
GMRES(120) ASM(1,ILU)	411	237	100	45.4	23.2	11.4	457	261	111
k = 3									
processes	32	64	128	32	64	128	32	64	128
FGMRES MG$_\mathcal{V}$ $L = 2$	111	54.2	36.9	98.4	49.6	25.1	209	104	62.0
FGMRES MG$_\mathcal{V}$ $L = 3$	60.9	24.1	15.1	101	50.2	26.1	162	74.2	41.2
FGMRES MG$_\mathcal{V}$ $L = 4$	53.8	30.2	14.7	103	51.9	26.4	157	81.4	41.1
CG BJACOBI(ILU)	472	305	172	94.5	46.5	23.5	567	352	195
GMRES(120) ASM(1,ILU)	729	386	236	94.6	47.3	24.1	824	433	262

Table 10: BR2 dG discretization of the 3D model Poisson problem (35)-(85). Solution, assembly and total CPU times (total means assembly plus solution times) required to reach a relative residual tolerance of 10^{-10}, see text for details.

analysis would require to further increase the number of processes we observe a strong linear scaling, that is the computation times halves doubling the number of processes.

CPU time (s)	grid topology computation				
processes	8	16	32	64	128
$L = 2$	31.3	17.1	7.15	2.88	1.77
$L = 3$	35.2	19.4	8.94	3.65	2.13
$L = 4$	41.7	22.5	10.2	4.90	2.52
$L = 0$	6.05	4.91	1.71	0.75	0.45

Table 11: Preprocessing phases required for h-multigrid computations as compared to single grid computations ($L = 0$). CPU times for generation of h-coarsened mesh sequences (grid topology computation task), see text for details.

CPU time (s)	orthonormalization and intergrid operators								
	k=1			k=2			k=3		
processes	8	16	32	16	32	64	32	64	128
$L = 2$	5.58	2.78	1.51	9.65	4.95	2.88	24.4	12.3	6.69
$L = 3$	7.75	3.88	2.02	13.9	7.73	3.65	37.6	19.7	10.7
$L = 4$	11.5	5.81	2.95	22.8	9.85	5.58	61.9	31.1	18.4
$L = 0$	1.03	0.52	0.26	1.88	0.96	0.48	4.06	2.04	1.11

Table 12: Preprocessing phases required for h-multigrid computations as compared to single grid computations ($L = 0$). CPU times for orthogonalization of shape functions and computation of intergrid operators, see text for details.

To demonstrate that also the preprocessing phase is scalable, in Table 11 and Table 12 we report CPU times for the generation of h-coarsened mesh sequence and orthogonalization of shape functions together with computation and storage of intergrid transfer operators, respectively. Remarkably, since the problem size has increased as compared to 2D computations, multigrid outperforms single grid solvers in terms of overall computation time (that is considering assembly, solution and preprocessing CPU times) at all polynomials degrees.

The gains with respect to the best performing single grid solver (the preconditioned Conjugate Gradient iteration) are on pair with those observed in 2D computations and do not deteriorate increasing the number of processes, see

$\text{MG}_{\mathcal{V}}(k)$ vs $\text{CG}(k)$	Total CPU time speedup, 2M hex elems grid								
degree	$k=1$			$k=2$			$k=3$		
processes	8	16	32	16	32	64	32	64	128
CG/MG step time	2.4	2.3	2.3	3.6	3.4	3.2	3.6	4.3	4.7

Table 13: BR2 dG discretization of the 3D model Poisson problem (35)-(85). Comparison of total CPU times (assembly plus solution times) required to reach a relative residual tolerance of 10^{-10} with Conjugate Gradient and h-multigrid ($L=4$), see text for details.

$\text{CG}(k_{\text{CG}})$ vs $\text{MG}_{\mathcal{V}}(k_{\text{MG}})$	Total CPU time ratio		
solver(degree)	$\text{CG}(1)/\text{MG}_{\mathcal{V}}(2)$	$\text{CG}(2)/\text{MG}_{\mathcal{V}}(3)$	$\text{CG}(1)/\text{MG}_{\mathcal{V}}(3)$
CG/MG step time	1.1	1.6	0.53

Table 14: BR2 dG discretization of the 3D model Poisson problem (35)-(85). Comparison of total CPU times (assembly plus solution times) required to reach a relative residual tolerance of 10^{-10} with Conjugate Gradient and h-multigrid. The multigrid solution strategy is applied to an higher polynomial degree dG discretization ($k_{\text{MG}} > k_{\text{CG}}$).

Table 13. Similarly to the 2D case solving a second degree polynomial degree BR2 dG discretization with h-multigrid is comparable to solving a first degree dG discretization with CG and the time required for solving a third polynomial degree BR2 dG discretization with h-multigrid is twice the time required for solving a first degree dG discretization with Conjugate Gradient, see Table 7.

6.2. Stokes dG discretization

In this section we tackle the solution of a model Stokes problem discretized by means of the dG formulation in (24). The computational domain is the bi-unit square, $\Omega = [-1,1]^2$, and we impose Dirichlet boundary conditions on $\partial\Omega$ according to the following smooth analytical solution

$$\mathbf{u} = \left[-e^x\left(y\,\cos(y)+\sin(y)\right)\mathbf{i}, e^x\left(y\,\sin(y)\right)\mathbf{j}\right],$$
$$p = 2\,e^x\,\sin(y),$$

In order to investigate the growth of computational costs while increasing the mesh size, solutions are computed on four uniform quadrilateral elements meshes of size $(64\cdot 2^n)^2, n=\{0,1,2,3\}$ and four distorted and graded triangular meshes

of size $2(32\cdot 2^n)^2, n=\{0,1,2,3\}$, see Figure 2. We check the influence of raising the polynomial degree and the number of coarse levels considering $k=\{1,2,3\}$ and $L=\{2,3,4,5\}$. The number of mesh elements at each level ℓ of the stack of grids is reported in Table (1).

For the sake of comparison we consider the h-multigrid V-cycle preconditioned FGMRES iteration and the Pressure Schur Complement preconditioned Richardson iteration of Section 5. The relevant solvers options are summarized in what follows.

$\mathrm{MG}_{\mathcal{V}}$ *preconditioned FGMRES solver.* One iteration of $\mathrm{MG}_{\mathcal{V}}$ cycle is used as a preconditioner for the FMGRES(60) iteration. On quadrilateral meshes high-order modes of the error are smoothed with a single iteration of a right ILU preconditioned GMRES solver while on distorted and graded triangular meshes we consider one and two smoothing iterations. On the coarsest level L we employ the same solver but, instead of fixing the iteration number, we impose a four order of magnitude decrease of the relative residual norm, that is $\frac{\|f_L - A_L^{\mathrm{Stk}}\bar{w}_L\|}{\|f_L\|} \leq 10^{-4}$. The FGMRES iteration is forced to reach tight relative residual tolerance, in particular the linear system solution converges in N_{it} iterations if at the i-th iterate $\|\hat{r}_0^i\| = \frac{\|f_0 - A_0^{\mathrm{Stk}}\bar{w}_0^i\|}{\|f_0\|} \leq 10^{-12}$.

Pressure Schur Complement $\mathrm{MG}_{\mathcal{V}}$ *preconditioned Richardson solver.* To approximatively invert the discrete vector Laplace operator A_0 and the pressure Schur complement $\widetilde{S}_0$ appearing in the block factorization of A_0^{Stk}, see Section 5, we employ a FGMRES(60) and a GMRES(60) solver, respectively. We set a two order of magnitude decrease of the relative residual norm and limit the maximum number of iterations to 2 and 40, respectively. The pressure Schur Complement GMRES solver is preconditioned with an ILU decomposition of the operator $\widehat{S}_0$ in (82). The FMGRES solver acting on the discrete Laplace operator is preconditioned with one and two iteration of multigrid V-cycle on

quadrilateral and triangular grids, respectively. Smoothing options are the same of Section 6.1. The Richardson iteration is forced to reach a relative residual tolerance of 10^{-12}.

We compare the solvers on the basis of convergence rate and computation time and we compare execution times with a Lower Upper (LU) decomposition direct solver and the ILU preconditioned GMRES(200) solver. All the numerical results have been computed by exploiting the PCMG multigrid preconditioner and the PCFIELDSPLIT block preconditioner frameworks available in the `PETSc` library [40, 37]. Provided that the iterative solver's convergence criterion is satisfied and the LU factorisation is computed without running out of memory, the same error with respect to the exact solution is measured. For the $k = 3$ dG discretization on the 128^2 quadrilateral grid the L^2 error norm is on the order of 10^{-10} and 10^{-8} for velocity and pressure, respectively.

The number of iteration reported in Table 15 and Table 16 confirm uniform converge with respect to the number of levels for multigrid preconditioned solvers, both on quadrilateral and triangular mesh sequences. Nevertheless, while on uniform quadrilateral elements grids the convergence is grid independent, on distorted and graded triangular meshes the number of iterations increases on finer grids. Moreover, only on quadrilateral elements meshes increasing the polynomial degree entails less iterations. This can be better appreciated by inspecting the average residual decrease or *convergence factor*

$$\rho = exp\left(\frac{1}{N_{\text{it}}} ln \frac{\|r_{N_{\text{it}}}\|}{\|r_0\|}\right)$$

reported in Table 17. Convergence failure after 2000 ILU preconditioned GMRES iterations reads 2000^* in Tables 15-16.

Looking at the wall clock times (solution times plus assembly times) reported in Table 18 it is clear that both multigrid preconditioned iterative solvers yield

significant execution times gains with respect to direct solver on the quadrilateral mesh sequence. Since we get a four-to-five-fold increase of the total computation time with a four-fold increase of the mesh size at all the polynomials degrees, optimal multigrid efficiency is approached. Note that in case of the h-multigrid preconditioned FGMRES solver the number of levels L must be chosen large enough because of the poor performance of the coarse grid GMRES solver, even on relatively coarse meshes. In this regard uniform convergence with respect to the number of levels is highly beneficial.

Linear solver iterations, 2D Stokes problem, quadrilateral mesh sequence

solver	FGMRES $MG_{\mathcal{V}}$				SchurCompl $MG_{\mathcal{V}}$				GMRES ILU
L	2	3	4	5	2	3	4	5	0
grid	**k = 1**								
64	25	26	27	27	11	11	11	11	1013
128	25	27	27	27	11	12	11	11	2000*
256	26	27	28	28	11	12	12	12	2000*
512	27	27	28	28	11	12	12	12	2000*
	k = 2								
64	16	16	17	17	10	10	10	10	531
128	16	16	17	17	10	10	10	10	2000*
256	16	16	17	17	10	10	10	10	2000*
512	16	16	17	17	10	10	10	10	2000*
	k = 3								
64	13	14	14	14	10	10	10	10	597
128	13	14	14	14	10	10	10	10	2000*
256	13	14	14	14	10	10	10	10	2000*
512	18	14	14	14	10	10	10	10	2000*

Table 15: Comparison of the number of iterations required to solve a 2D model Stokes problem, see text for details. Linear system relative residual tolerance is 10^{-12}.

The wall clock times measured on the triangular mesh sequence and reported in Table 19 demonstrate that only the preconditioned FGMRES solver allows to bit direct solvers. The performance of the Schur Complement block preconditioner is hit by the poor performance of the Schur complement subsolver which fails to lower the residual by two orders of magnitude in 40 iterations (we verified that increasing the maximum number of iteration beyond 40 is not beneficial in terms of execution times). As opposite the multigrid preconditioned FGM-

Linear solver iterations, 2D Stokes problem, triangular mesh sequence

solver	FGMRES $MG_{\mathcal{V}}$								SchurCompl $MG_{\mathcal{V}}$				GMRES
SM it	1				2				2				
L	2	3	4	5	2	3	4	5	2	3	4	5	0
grid	**k = 1**												
32	35	35	35	35	20	20	20	20	28	29	29	30	464
64	39	39	39	39	23	23	23	23	44	45	45	46	1433
128	46	47	47	47	26	26	26	26	70	70	70	71	2000*
256	51	51	51	51	29	29	29	29	86	86	86	86	2000*
	k = 2												
32	29	29	29	29	16	16	16	16	29	29	29	29	374
64	36	36	36	36	22	22	22	22	45	45	45	45	1526
128	55	55	55	55	28	28	28	28	81	80	81	81	2000*
256	73	70	75	76	41	41	41	41	64	63	66	65	2000*
	k = 3												
32	27	27	27	27	14	14	14	14	39	39	39	39	437
64	36	36	36	36	19	19	19	19	57	57	57	57	1863
128	44	44	44	44	26	26	26	26	72	72	72	72	2000*
256	63	63	63	63	40	40	40	40	119	119	119	120	2000*

Table 16: Comparison of the number of iterations required to solve a 2D model Stokes problem, see text for details. One and two smoothing iterations (SM it) are considered for triangular meshes to improve the performance of multigrid preconditioners. Linear system relative residual tolerance is 10^{-12}.

Convergence factor ρ, 2D Stokes problem

grid	quadrilateral meshes				(triangular meshes)			
solver	FGMRES $MG_{\mathcal{V}}$		SchurCompl $MG_{\mathcal{V}}$		FGMRES $MG_{\mathcal{V}}$		SchurCompl $MG_{\mathcal{V}}$	
SM it	1		1		1		2	
L	2	5	2	5	2	5	2	5
grid size	**k = 1**							
64 (32)	.324	.349	.070	.075	.456	.456	.372	.387
128 (64)	.328	.354	.078	.079	.497	.498	.536	.544
256 (128)	.337	.364	.078	.089	.554	.556	.676	.679
512 (256)	.348	.370	.079	.087	.583	.584	.728	.727
	k = 2							
64 (32)	.162	.186	.053	.055	.382	.383	.383	.383
128 (64)	.165	.193	.052	.053	.470	.471	.544	.543
256 (128)	.165	.196	.055	.056	.608	.606	.711	.712
512 (256)	.167	.196	.051	.054	.687	.695	.651	.658
	k = 3							
64 (32)	.116	.124	.060	.061	.349	.349	.495	.494
128 (64)	.117	.127	.051	.051	.464	.465	.616	.617
256 (128)	.118	.129	.053	.051	.539	.540	.682	.682
512 (256)	.205	.130	.051	.051	.648	.648	.792	.794

Table 17: 2D model Stokes problem. Convergence factors of a FGMRES solver preconditioned with h-multigrid and a Richardson solver with a Pressure Schur Complement Block preconditioner, see text for details. Quadrilateral and distorted triangular mesh sequences. One and two smoothing iterations (SM it) are considered for triangular meshes to improve the performance of multigrid preconditioners. Linear system relative residual tolerance is 10^{-12}.

RES solver employed for the Laplace operator performs fairly well on distorted triangular meshes, see Table 4. Also the performance of h-multigrid FGMRES degrade as compare to quadrilateral elements meshes: we observe a six-fold increase of the total computation time with a four-fold increase of the mesh size. Worsening of convergence factors is awaited given that the number of stretched triangles increases and their aspect ratios worsen when the mesh is refined, see Figure 2.

Total CPU time (s), 2D Stokes problem, quadrilateral mesh sequence

solver	FGMRES $MG_{\mathcal{V}}$				SchurCompl $MG_{\mathcal{V}}$				GMRES	LU
L	2	3	4	5	2	3	4	5	0	0
grid	**k = 1**									
64	1.03	0.91	0.91	0.89	1.61	2.0	1.86	1.92	11.1	1.33
128	5.07	3.87	3.83	3.89	7.57	8.42	7.62	7.58	90.4*	10.7
256	42.2	17.6	16.8	16.8	31.0	36.3	35.4	37.0	349*	84.0
512	480	99.4	75.9	75.7	137	151	153	149	1278*	691
	k = 2									
64	2.81	1.84	1.83	1.89	5.86	5.97	6.1	6.22	11.2	6.06
128	13.76	7.86	7.78	7.83	25.7	27.1	28.6	28.0	168*	49.9
256	113	39.9	33.1	32.7	114	110	109	112	713*	409
512	977	288	172	159	497	494	485	547	2690*	
	k = 3									
64	6.19	4.47	4.44	4.51	18.0	18.1	18.5	18.6	30.2	17.2
128	37.2	20.8	18.6	18.2	79.8	82.4	85.6	81.8	365*	146
256	301	103	77.8	74.8	326	325	325	324	1533*	
512	3012	734	381	347	1419	1360	1355	1350	5973*	

Table 18: Comparison of wall clock time (solution plus assembly times) required to solve a 2D model Stokes problem, see text for details. Linear system relative residual tolerance is 10^{-12}.

The comparison between direct solver and h-multigrid FGMRES CPU times proposed in Table 20 confirms that strong gains can be attained by means of multilevel preconditioners, even on unstructured meshes composed of stretched and skewed elements. On fine enough uniform quadrilateral mesh solving a first degree dG discretization with an LU solver is comparable to solving a third degree dG discretization with a multigrid preconditioned FGMRES solver. Similarly, on a fine enough distorted triangular grids, $k+1$ and k degree dG discretizations are comparable in terms of exacution time if solved with multigrid

Total CPU time (s), 2D Stokes problem, triangular mesh sequence

solver	FGMRES MG$_\mathcal{V}$				SchurCompl MG$_\mathcal{V}$				GMRES	LU
L	2	3	4	5	2	3	4	5	0	0
grid	**k = 1**									
32	0.42	0.42	0.42	0.42	7.63	8.33	8.75	9.33	1.70	0.26
64	2.08	1.93	1.90	1.94	51.0	55.1	56.1	57.7	21.8	2.22
128	11.1	9.04	9.10	8.74	329	346	356	357	121*	18.9
256	80.2	44.5	40.3	40.6	1798	1870	1895	1879	522*	158
	k = 2									
32	1.04	0.99	0.99	1.10	19.5	20.6	21.6	22.1	3.30	1.65
64	6.49	5.53	5.57	5.57	136	144	138	137	58.6	14.9
128	39.8	29.5	28.0	26.5	1000	1082	1123	1098	315*	129
256	290	181	164	156	3433	3315	3458	3397	1305*	1087
	k = 3									
32	2.73	2.74	2.57	2.64	70.6	72.7	73.8	74.4	9.45	4.16
64	17.6	14.4	14.1	13.9	457	472	480	510	160	39.5
128	116.8	77.5	72.1	71.0	2473	2444	2459	2472	689*	343
256	973	484	419	434	12112	12360	12355	12350	2782*	

Table 19: Comparison of wall clock time (solution plus assembly times) required to solve a 2D model Stokes problem, see text for details. Linear system relative residual tolerance is 10^{-12}.

MG$_\mathcal{V}(k)$ vs LU(k)	Total CPU time speedup										
degree	$k=1$				$k=2$				$k=3$		
quad grid	64	128	256	512	64	128	256	512	64	128	256
LU/MG step time	1.5	2.7	5	9.1	3.2	6.4	12.5		3.8	8	
tri grid	32	64	128	256	32	64	128	256	32	64	128
LU/MG step time	0.6	1.1	2.2	3.9	1.5	2.7	4.9	7	1.6	2.8	4.8

Table 20: 2D model Stokes problem. Comparison of total CPU times (assembly plus solution times) required to solve with a direct solver and with h-multigrid preconditioned FGMRES ($L = 5$), see text for details. Linear system relative residual tolerance is 10^{-12} in case of FGMRES.

preconditioned FGMRES and direct LU solver, respectively.

Since the multigrid V-cycle preconditioner is the best performing and is reliable on low quality grids, in the next section the strategy will be applied for solving non-linear incompressible flow problems.

6.3. Navier-Stokes dG discretization

In this section we assess the performance of the multigrid preconditioned FGMRES solver applied to repeatedly solve the linearized system of equation in (21), as required for advancing in time the dG discretization of the incom-

pressible Navier-Sokes equation in (14) by means of the Backward Euler method.

We consider the 2D Kovasznay and 2D-3D Lid-driven cavity problems, admitting a steady state solution at the Reynolds numbers considered in this work. Thus we tackle a real-life transient hemodynamic application: we consider the possibility to simulate the blood flow behavior all along the cardiac cycle in a 3D cerebral aneurysm geometry reconstructed from medical images.

We remark that the Backward Euler method can be modified to implement a pseudo-transient continuation strategy, Ψ_{tc} see *e.g.* [44], that can be employed to seek steady state solutions of the incompressible Navier-Stokes equations. Roughly speaking it is sufficient to omit the while loop in Algorithm 1 (which is done for efficiency purposes since accuracy of the time integration is unnecessary) and introduce a time step adaptation strategy, *e.g.* the Successive Evolution Relaxation Strategy [45], which allows to progressively enlarge the pseudo time step (starting from a sufficiently small initial guess) when the steady state solution is approached. Ψ_{tc} is a globalization of Newton method that guarantees convergence even when the tentative solution is far from the sought steady state solution. Moreover the favourable convergence rates of the Newton method can be exploited when the pseudo time step is large enough.

6.3.1. Kovasznay test case

To assess convergence with respect to the number of levels we consider the 2D Kovasznay problem [46] at Reynolds 40. Dirichlet boundary conditions are imposed according to the exact solution and we seek for the steady state solution starting from fluid at rest. Since the flow regime is diffusion dominated we set the initial pseudo-time step of the continuation strategy to a very large value (10^{13}) and fall back to pure Newton for the steady Navier-Stokes equations. We impose a four order of magnitude decrease of the relative residual norm at each Newton iteration: *i.e.* at the n-th Newton iterate the

i-th iterate of the multigrid preconditioned FGMRES solver has congerged if $\|\hat{r}_0^{n,i}\| = \frac{\|f_0(w_0^n) - A_0^{\mathrm{INS}}(w_0^n)\overline{\delta w}_0^{n,i}\|}{\|f_0(w_0^n)\|} \leq 10^{-4}$. Convergence is achieved is six Newton iterations, the steady state solution w_0^6 is such that $|f_0(w_0^6)| \leq 10^{-12}$.

The smoothing and solver option for the multigrid preconditioner are the same that in the Stokes case. One iteration of $\mathrm{MG}_{\mathcal{V}}$ cycle is used as a preconditioner for the FMGRES(60) iteration. Smoothing is performed with a single iteration of a right ILU preconditioned GMRES solver while for the ILU preconditioned GMRES solver on level L we impose a four order of magnitude residual decrease. Besides the multilevel V-cycle iteration, for the Kovasznay test case we include the results obtained with the W-cycle iteration, see *e.g.* [33]. The V- and W-cycle iterations differ in terms of the coarse grid correction of Algorithm 2, as outlined below

Coarse grid correction (V-cycle)	*Coarse grid correction (W-cycle)*
$r_\ell = f_\ell - A_\ell \overline{w}_\ell$	$r_\ell = f_\ell - A_\ell \overline{w}_\ell$
$r_{\ell+1} = \mathcal{I}_\ell^{\ell+1} r_\ell$	$r_{\ell+1} = \mathcal{I}_\ell^{\ell+1} r_\ell$
$e_{\ell+1} = \mathrm{MG}_{\mathcal{V}}(\ell+1, r_{\ell+1}, 0)$	$\widehat{e}_{\ell+1} = \mathrm{MG}_{\mathcal{V}}(\ell+1, r_{\ell+1}, 0)$
	$e_{\ell+1} = \mathrm{MG}_{\mathcal{V}}(\ell+1, r_{\ell+1}, \widehat{e}_{\ell+1})$
$\widehat{w}_\ell = \overline{w}_\ell + \mathcal{I}_{\ell+1}^{\ell} e_{\ell+1}$	$\widehat{w}_\ell = \overline{w}_\ell + \mathcal{I}_{\ell+1}^{\ell} e_{\ell+1}$

In order to investigate the growth of computational costs while increasing the mesh size, 2D solutions are computed on three uniform quadrilateral elements meshes of size $(128 \cdot 2^n)^2, n = \{1, 2, 3\}$ of the bi-unit square domain $[-0.5, 1.5] \times [0, 2]$. We check the influence of raising the polynomial degree on the convergence rate and the computational expense considering $k = \{1, 2, 3\}$. To investigate the influence of the number of coarse levels on the convergence rate we consider $L = \{2, 3, 4, 5\}$.

Since we are considering the performance of a linear multigrid iteration ap-

plied to each of six Newton method steps required to reach the steady state solution all the numerical results presented in what follows are averaged over the six steps. The number of linear iterations reported in Table 21 and the convergence factors reported in Table 22 show that only the W-cycle iteration yields uniform convergence with respect to the number of levels. The influence of the number of levels on the V-cycle iteration is not dramatic but clearly noticeable. The number of iterations is not grid independent but moving from a 128^2 to a 512^2 quadrilateral elements mesh (a sixteen-fold increase of the number of elements) the iterations increase is less than two-fold. Also the polynomial degree dependence is mild: similarly to the Stokes case a slight worsening of the convergence rates is observed for $k = 2$.

Average linear solver iterations									
k	1			2			3		
grid	128	256	512	128	256	512	128	256	512
FGMRES $MG_{\mathcal{V}}$ $L = 2$	7	8	9	8	11	16	8	9	12
FGMRES $MG_{\mathcal{V}}$ $L = 3$	8	9	10	10	12	17	9	11	13
FGMRES $MG_{\mathcal{V}}$ $L = 4$	9	10	12	11	14	18	11	13	16
FGMRES $MG_{\mathcal{V}}$ $L = 5$	10	11	13	12	15	20	12	14	18
FGMRES $MG_{\mathcal{W}}$ $L = 2$	5	6	8	7	9	13	6	8	11
FGMRES $MG_{\mathcal{W}}$ $L = 3$	5	6	8	7	9	13	6	8	11
FGMRES $MG_{\mathcal{W}}$ $L = 4$	5	6	8	7	9	13	6	8	11
FGMRES $MG_{\mathcal{W}}$ $L = 5$	5	6	8	7	9	13	6	8	11
GMRES(200) ILU(0)	464	1589^*		449	1518		579	1669^*	

Table 21: 2D Kovasznay problem. Number of iterations of a FGMRES solver preconditioned with a V-cycle and a W-cycle h-multigrid iteration (one iteration), see text for details. Linear system relative residual tolerance is 10^{-4}. Average linear iterations over the six Newton steps required to find the steady state solution.

The wall clock times comparison of Table 23 confirms that strong gains can be obtained as compared to the ILU preconditioned GMRES(200) iteration. Interestingly, even if the W-cycle iteration is the best performing in terms of convergence rates, the increased computational cost as compared to the V-cycle penalizes execution times. Similarly to the Stokes case we get a four-to-five fold increase of the computational cost with a four fold increase of the number of levels.

Average convergence factor, ρ									
k	1			2			3		
grid	128	256	512	128	256	512	128	256	512
FGMRES MG$_{\mathcal{V}}$ $L = 2$	.23	.26	.34	.31	.40	.53	.28	.34	.43
FGMRES MG$_{\mathcal{V}}$ $L = 3$	.29	.33	.39	.36	.44	.55	.35	.41	.47
FGMRES MG$_{\mathcal{V}}$ $L = 4$	.33	.37	.43	.40	.49	.58	.40	.46	.52
FGMRES MG$_{\mathcal{V}}$ $L = 5$	.36	.41	.47	.43	.52	.61	.42	.49	.57
FGMRES MG$_{\mathcal{W}}$ $L = 2$	.14	.20	.28	.22	.33	.45	.20	.28	.40
FGMRES MG$_{\mathcal{W}}$ $L = 3$	.14	.20	.28	.23	.33	.45	.20	.28	.40
FGMRES MG$_{\mathcal{W}}$ $L = 4$	.14	.20	.28	.23	.33	.45	.21	.28	.40
FGMRES MG$_{\mathcal{W}}$ $L = 5$	.14	.20	.28	.23	.33	.45	.21	.28	.40

Table 22: 2D Kovasznay problem. Convergence factors of a FGMRES solver preconditioned with a V-cycle and a W-cycle h-multigrid iteration (one iteration), see text for details. Linear system relative residual tolerance is 10^{-4}. Average convergence factors over the six Newton steps required to find the steady state solution.

Total CPU time (s), 2D Kovasznay problem, quadrilateral mesh sequence

solver	FGMRES MG$_{\mathcal{V}}$				FGMRES MG$_{\mathcal{W}}$				GMRES
L	2	3	4	5	2	3	4	5	0
grid	**k = 1**								
128	3.18	2.90	3.08	3.24	2.86	2.47	2.48	2.56	16.5
256	25.6	13.4	13.3	14.0	26.2	12.4	11.0	11.1	229
512	229	79.1	59.8	62.3	294	108	58.9	51.8	
	k = 2								
128	11.1	8.14	8.34	8.58	13.4	8.46	8.15	8.27	46.0
256	109	42.9	37.9	39.2	138	58.1	42.5	40.8	586
512	1090	313	176	179	1589	627	269	215	
	k = 3								
128	29.4	19.6	19.8	20.8	35.1	22.6	19.8	19.8	107
256	283	108	94.1	97.6	361	152	106	94.8	1640
512	2218	829	481	471	3399	1563	754	550	

Table 23: 2D Kovasznay problem. Comparison of wall clock time (solution plus assembly times) required to solve linearized systems of Newton method, see text for details. Linear system relative residual tolerance is 10^{-4}. . Average CPU times over the six Newton steps required to find the steady state solution

.

As observed for the Stokes problem, the assembly and solution wall clock times of Table 24 confirms that it is important to choose a sufficiently high number of levels not to get penalized by the poor performance of the ILU preconditioned GMRES coarse grid solver.

CPU time (s)	solution			assembly			total		
grid	128	256	512	128	256	512	128	256	512
k = 1									
FGMRES MG $L = 2$	1.64	19.3	204	1.54	6.25	24.8	3.19	25.6	229
FGMRES MG $L = 3$	1.14	6.34	50.8	1.75	7.01	28.2	2.90	13.4	79.1
FGMRES MG $L = 4$	1.18	5.68	29.3	1.89	7.59	30.5	3.08	13.3	59.8
FGMRES MG $L = 5$	1.25	5.97	30.1	1.99	8.01	32.1	3.24	14.0	62.3
GMRES(200) ILU(0)	15.8	226		0.70	2.78		16.5	229	
k = 2									
FGMRES MG $L = 2$	8.18	97.4	1048	2.90	11.7	42.5	11.1	109	1090
FGMRES MG $L = 3$	4.89	30.0	271	3.24	12.9	41.4	8.14	42.9	313
FGMRES MG $L = 4$	4.88	24.5	131	3.07	13.5	44.3	8.34	37.9	176
FGMRES MG $L = 5$	4.96	24.9	131	3.62	14.2	47.0	8.58	39.2	179
GMRES(200) ILU(0)	44.4	580		1.58	5.90		46.0	586	
k = 3									
FGMRES MG $L = 2$	23.6	260	2121	5.81	23.5	96.7	29.5	283	2218
FGMRES MG $L = 3$	13.2	81.8	720	6.40	25.9	109	19.6	108	829
FGMRES MG $L = 4$	12.9	66.3	364	6.89	27.7	115	19.8	94.1	480
FGMRES MG $L = 5$	13.5	68.1	349	7.27	29.6	121	20.8	97.6	471
GMRES(200) ILU(0)	104	1626		3.01	13.4		107	1640	

Table 24: 2D Kovasznay problem. Solution, assembly and total (solution plus assembly) times for solving linearized systems of Newton method with with a FGMRES solver preconditioned with a V-cycle h-multigrid iteration, see text for details. Linear system relative residual tolerance is 10^{-4}. Average CPU times over the six Newton steps required to find the steady state solution.

6.3.2. Lid-Driven cavity test case

To investigate the influence of the Reynolds number on the convergence rates and the performance in three space dimensions we consider the lid-driven cavity problem. We rely on a uniform 100^2 quadrilateral and a uniform 80^3 hexahedral grid of the unit square and the unit cube, respectively. We check the influence of raising the polynomial degree on the convergence rate considering $k = \{1, 2, 3, 4\}$ in 2D but omitting $k = 4$ in 3D. We consider two Reynolds numbers, $Re = 1000$ and $Re = 5000$, in 2D and $Re = 1000$ in 3D.

	Ψ_{tc} iterations		convergence factor ρ max(avg)	
Re	1000	5000	1000	5000
k = 1				
FGMRES MG $L = 3$	24	45	.556 (.395)	.815 (.537)
FGMRES MG $L = 4$			.646 (.451)	.883 (.567)
k = 2				
FGMRES MG $L = 3$	23	53	.321 (.228)	.638 (.349)
FGMRES MG $L = 4$			.463 (.306)	.753 (.394)
k = 3				
FGMRES MG $L = 3$	23	79	.171 (.126)	.483 (.228)
FGMRES MG $L = 4$			.313 (.199)	.682 (.284)
k = 4				
FGMRES MG $L = 3$	23	67	.126 (.090)	.469 (.225)
FGMRES MG $L = 4$			.201 (.132)	.654 (.289)

Table 25: 2D lid-driven cavity problem. Number of pseudo-transient continuation iterations and maximum/average convergence factors (ρ) measured over the Ψ_{tc} iterations. Linearized systems of Ψ_{tc} method are solved with a FGMRES solver preconditioned with a V-cycle h-multigrid iteration, see text for details.

Since the Reynolds number is higher than in the Kovasznay case and we approach convection dominated flow regimes, we seek for a steady state solution starting from fluid at rest by means of of the pseudo-transient continuation strategy with SER time stepping. Besides adapting the time step, it is convenient to adapt the *forcing terms*, that is the relative relative tolerance triggering convergence of the linear system at each continuation step, we adopt the strategy proposed in [47]. The goal is to avoid *oversolving* of the linear system when the linearization of the residual $f(w^{n+1}) = f(w^n) + J(w^n)\overline{\delta w}^n$ is not sufficiently accurate to pay off in terms of convergence towards the steady state.

The smoothing and solver option for the multigrid preconditioner are the same that in the Stokes and Kovasznay case. One iteration of $\mathrm{MG}_{\mathcal{V}}$ cycle is used as a preconditioner for the FMGRES(60) iteration. High-order modes of the error are smoothed with a single iteration of a right ILU preconditioned GMRES solver while we require a four order of magnitude residual decrease for the ILU preconditioned GMRES solver on level L. In parallel computations ILU preconditioners are replaced with ASM preconditioners with one level of

Re 1000	np	avg card($\mathcal{T}_\ell^i$)	Ψ_{tc} it	ρ max(avg)
k = 1				
FGMRES MG $L = 2$	32	16000/2350/350	33	.806 (.509)
FGMRES MG $L = 3$		16000/2350/350/53		.942 (.589)
k = 2				
FGMRES MG $L = 2$	64	8000/1175/170	33	.535 (.356)
FGMRES MG $L = 3$		8000/1175/170/25		.822 (.487)
k = 3				
FGMRES MG $L = 2$	128	4000/585/90	34	.392 (.259)
FGMRES MG $L = 3$		4000/585/90/13		.630 (.390)

Table 26: 3D lid-driven cavity problem. Number of processes (np), average grid partition cardinality on each level of the h-coarsened mesh sequence, number of pseudo-transient continuation iterations and maximum/average convergence factors (ρ) measured over the Ψ_{tc} iterations. Linearized systems of Ψ_{tc} method are solved with a FGMRES solver preconditioned with a V-cycle h-multigrid iteration, see text for details.

overlap, as we did for solving elliptic problems in parallel in Section 6.1.

The average and maximum convergence factors measured over the Ψ_{tc} iterations are reported in Table 25. While the maximum convergence factors, usually observed in the terminal phase of the convergence (that is when the time step is large), are significantly affected by raising the Reynolds number, the average convergence factors are satisfactorily small at Reynold 5000. Interestingly increasing the polynomial degree is beneficial from the convergence rates viewpoint, very good performances are observed for $k = 4$.

Parallel 3D computations demonstrate that the convergence rates do not degrade, even if the number of mesh elements in each grid partition is remarkably small on the coarsest level, see Table 26. The trend observed in 2D is confirmed, raising the polynomial degree is advantageous from the convergence rate viewpoint. We remark that the third polynomial degree dG discretization on the 80^3 hexahedral elements grid tops at approximatively 10M unknowns.

6.3.3. Cerebral aneurysm hemodynamics

In this section we apply the Backward Euler time integration strategy of Algorithm 1 to approximate the blood flow field in a pathological Internal Carotid Artery (ICA) reconstructed from medical images, see Figure 4.

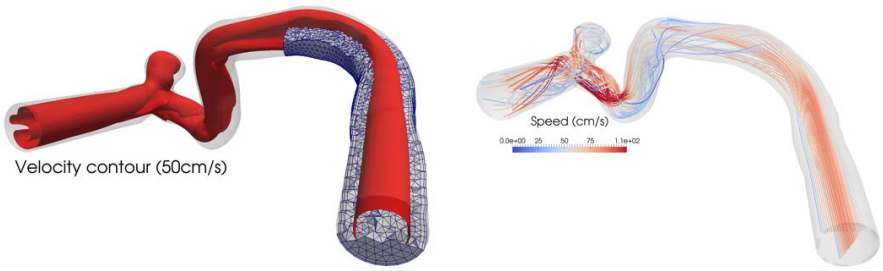

Figure 4: Hemodynamics of a cerebral aneurysm reconstructed from medical images. *Left*, hybrid (tetrahedral and prismatic) 270k elements grid and velocity contour. *Right*, streamlines computed at the systolic peak.

In order to take into account the pulsatile flow behaviour Dirichlet boundary conditions are imposed at the circular inflow section relying on the Womersley analytical solution [48] and considering a physiological flow rate all along the cardiac cycle [49]. The average Reynolds number $Re_{avg} = 500$. Stress-free boundary conditions are imposed at the outflow section and no-slip boundary conditions are imposed at the vessel walls. We apply $k = \{1, 2, 3\}$ polynomial degree dG discretizations over the 270K hybrid grid generated with the open-source Vascular Modeling Toolkit (VMTK) [50]. Simulations are performed running in parallel on 16, 32 and 64 processes for first, second and third degree dG discretizations, respectively. The fixed time steps is chosen such that 150 numerical solution are computed in each cardiac cycle. The time integration strategy is initialized with fluid at rest and conducted for three cardiac cycles.

$Re_{\text{avg}} = 500$	np	avg card$(\mathcal{T}_\ell^i)$	time steps	ρ max(avg)
k = 1				
FGMRES MG $L = 2$	16	10600/1560/230	150	.644 (.547)
k = 2				
FGMRES MG $L = 2$	32	5300/770/114	150	.678 (.540)
k = 3				
FGMRES MG $L = 2$	64	2650/390/57	150	.631 (.477)

Table 27: Cerebral aneurysm hemodynamics. Number of processes (np), average grid partition cardinality on each level of the h-coarsened mesh sequence, number of times steps per cardiac cycle and maximum/average convergence factors (ρ) measured over the Backward Euler time integration strategy. Linearized systems of Ψ_{tc} method are solved with a FGMRES solver preconditioned with a V-cycle h-multigrid iteration, see text for details.

For solving the linearized systems of the BE method (21) one iteration of $\text{MG}_{\mathcal{V}}$ cycle is used as a preconditioner for the FMGRES(60) solver. High-order modes of the error are smoothed with a single iteration of an ASM preconditioned GMRES solver while we require a four order of magnitude residual decrease for the ASM preconditioned GMRES solver on level L. ASM preconditioners employ one level of overlap between sub-domains and an ILU decompositions for each sub-domain matrix.

Table 27 reports the maximum and average convergence rates measured over the third cardiac cycle. Since the time step is fixed, the gap between maximum and average converge factors is narrower than in the pseudo-transient continuation strategy, *cf.* Table 26, and reflects the influence of varying the Reynolds number. In particular the maximum convergence factor is recorded during systole where convection is more pronounced as compared to diastole.

Even if the average convergence rates reported in Table 27 are less satisfactory that in the lid-driven cavity case, the fact that the linear system residual halves at each FGMRES iteration is a significant achievement. Hemodynamic computations are considered very challenging from the numerical solution viewpoint, to the point that even segregated Pressure Corrections strategies might require ad-hoc preconditioners [51].

7. Conclusions

This work demonstrates the feasibility and effectiveness of h-multigrid preconditioners applied to high-order accurate dG discretizations of incompressible flow problems. In view of efficiency agglomeration based h-multigrid strategies with inherited coarse grid operators are attractive because the expensive process of numerically integrating over agglomerated elements can be avoided in all but the preprocessing phase. Indeed, intergrid transfer operators can be

computed once prior to the non-linear iteration, and stored for later use. In this work we introduced an effective strategy for improving performance of inherited coarse grid operators which exploits a rescaled Galerkin projection of the BR2 dG discretization stabilization term. Using a single iteration of preconditioned GMRES as smoothing strategy, the multigrid convergence is uniform with respect to the number of levels and the *typical multigrid efficiency* is closely approached on model problems. The ability to beat direct solvers on arbitrarily unstructured low quality grids and the appealing performance obtained on parallel real-life computations might revert the common belief that discontinuous Galerkin discretizations are more expensive to solve as compared to standard finite element and finite volume formulations.

Appendix A. Implementation details: restriction of BR2 operators

We provide implementations details about the matrix-free implementation of the restriction of coarse grid operators. For the sake of brevity we consider inheritance of the BR2 bilinear forms, the Stokes and Navier-Stokes coarse grid operators can be obtained in a similar fashion. Note that BR2 coarse grid operators involve Galerkin projections for consistency terms, see Equation (71) and the rescaled Galerkin projection for the stabilization term, see Equation (72).

The matrices counterparts $\mathbf{A}_\ell^{\mathrm{BR2}}$ of the operators A_ℓ^{BR2} are sparse block matrices of size $(\mathrm{card}(\mathcal{T}_\ell)\ N_{\mathrm{dof}}^\kappa)^2$ (the block size is $(N_{\mathrm{dof}}^\kappa)^2$) composed of diagonal blocks $\mathbf{A}_{\kappa_\ell,\kappa_\ell}$ and off-diagonal blocks $\mathbf{A}_{\kappa_\ell,\kappa'_\ell}$. Off-diagonal blocks are responsible of the coupling between neighboring elements $\kappa_\ell, \kappa'_\ell$ sharing a face σ_ℓ. The coarse operators $A_\ell^{\widetilde{\mathcal{I}},\mathrm{BR2}}$ are obtained matrix-free as described in Algorithms 5 and 6.

Matrix restriction is performed contextually to fine matrix assembly so that stability term contributions $\mathbf{A}^{\mathrm{STB}}$, and consistency-symmetry terms contribu-

Algorithm 5 Inherited BR2 (mesh elements and boundary faces)

for $\kappa_0 \in \mathcal{T}_0$ **do**
  assemble $(\mathbf{A}^{\mathrm{CS}_\kappa}_{\kappa_0,\kappa_0})_{i,j} = a_0^{\mathrm{CS}_\kappa}(\varphi_i^{\kappa_0}, \varphi_j^{\kappa_0})$
  for $\ell = 0, ..., L-1$ **do**
    find $\kappa_{\ell+1} \in \mathcal{T}_{\ell+1}$ such that $\kappa_\ell \in K_\ell^{\ell+1}$
    $\mathbf{A}_{\kappa_{\ell+1},\kappa_{\ell+1}}$ += $\mathbf{M}_{\kappa_{\ell+1},\kappa_\ell}\ (\mathbf{A}^{\mathrm{CS}_\kappa}_{\kappa_\ell,\kappa_\ell})\ \mathbf{M}^T_{\kappa_{\ell+1},\kappa_\ell}$
  end for
end for
for $\sigma_0 \in \mathcal{F}_0^{\mathrm{b}}$ **do**
  find κ_0 such that $\sigma_0 = \partial\kappa_0 \cap \partial\Omega$
  assemble $(\mathbf{A}^{\mathrm{CS}_\sigma}_{\kappa_0,\kappa_0})_{i,j} = a_\ell^{\mathrm{CS}_\sigma}(\varphi_i^{\kappa_0}, \varphi_j^{\kappa_0})$
  assemble $(\mathbf{A}^{\mathrm{STB}}_{\kappa_0,\kappa_0})_{i,j} = s_\ell(\varphi_i^{\kappa_0}, \varphi_j^{\kappa_0})$
  for $\ell = 0, ..., L-1$ **do**
    find $\kappa_{\ell+1}$ such that $\kappa_\ell \in K_\ell^{\ell+1}$
    find $\sigma_{\ell+1}$ such that $\sigma_\ell \in \Sigma_\ell^{\ell+1}$
    $\mathbf{A}^{\mathrm{CS}_\sigma}_{\kappa_{\ell+1},\kappa_{\ell+1}} = \mathbf{M}_{\kappa_{\ell+1},\kappa_\ell}\ (\mathbf{A}^{\mathrm{CS}_\sigma}_{\kappa_\ell,\kappa_\ell})\ \mathbf{M}^T_{\kappa_{\ell+1},\kappa_\ell}$
    $\mathbf{A}^{\mathrm{STB}}_{\kappa_{\ell+1},\kappa_{\ell+1}} = \mathcal{H}^{\sigma_{\ell+1}}_{\sigma_\ell}\left(\mathbf{M}_{\kappa_{\ell+1},\kappa_\ell}\ (\mathbf{A}^{\mathrm{STB}}_{\kappa_\ell,\kappa_\ell})\ \mathbf{M}^T_{\kappa_{\ell+1},\kappa_\ell}\right)$
    $\mathbf{A}_{\kappa_{\ell+1},\kappa_{\ell+1}}$ += $\left(\mathbf{A}^{\mathrm{CS}_\sigma}_{\kappa_{\ell+1},\kappa_{\ell+1}} + \mathbf{A}^{\mathrm{STB}}_{\kappa_{\ell+1},\kappa_{\ell+1}}\right)$
  end for
end for

tions $\mathbf{A}^{\mathrm{CS}_\kappa}$ and $\mathbf{A}^{\mathrm{CS}_\sigma}$, see Section 4.3.1, are restricted separately, before being collected into diagonal and off-diagonal blocks of the fine matrix.

It is interesting to remark that only a subset of the internal faces contributions on level ℓ is restricted on level $\ell+1$. In particular we remark that all diagonal and off-diagonal contributions $\mathbf{A}^{\mathrm{STB}}, \mathbf{A}^{\mathrm{CS}_\sigma}$ associated to facets $\sigma \in \mathcal{F}_\ell^{\mathrm{i}}$ that do not belong to the boundary of agglomerated elements on level $\ell+1$ are ignored in Algorithm 6. This optimization is permitted thanks to the local conservation properties of dG formulations.

Acknowledgements

We acknowledge the CINECA HPC facility for the availability of high performance computing resources and support within the agreement "Convenzione di Ateneo Università degli Studi di Bergamo".

Algorithm 6 Inherited BR2 (internal faces)

for $\sigma_0 \in \mathcal{F}_0^{\mathrm{i}}$ **do**
find κ_0, κ_0' such that $\kappa_0 \neq \kappa_0'$ and $\sigma_0 = \partial\kappa_0 \cap \partial\kappa_0'$
assemble $\mathbf{A}^{\mathrm{CS}_\sigma}_{\kappa_0,\kappa_0}, \mathbf{A}^{\mathrm{CS}_\sigma}_{\kappa_0',\kappa_0'}, \mathbf{A}^{\mathrm{CS}_\sigma}_{\kappa_0,\kappa_0'}$ and $(\mathbf{A}^{\mathrm{CS}_\sigma}_{\kappa_0',\kappa_0})_{i,j} = a_0^{\mathrm{CS}_\sigma}(\varphi_i^{\kappa_0'}, \varphi_j^{\kappa_0})$
assemble $\mathbf{A}^{\mathrm{STB}}_{\kappa_0,\kappa_0}, \mathbf{A}^{\mathrm{STB}}_{\kappa_0',\kappa_0'}, \mathbf{A}^{\mathrm{STB}}_{\kappa_0,\kappa_0'}$ and $(\mathbf{A}^{\mathrm{STB}}_{\kappa_0',\kappa_0})_{i,j} = s_0(\varphi_i^{\kappa_0'}, \varphi_j^{\kappa_0})$
for $\ell = 0, ..., L-1$ **do**
find $\kappa_{\ell+1}, \kappa'_{\ell+1}$ such that $\kappa_\ell \in K_\ell^{\ell+1},\ \kappa'_\ell \in {K'}_\ell^{\ell+1}$
find $\sigma_{\ell+1}$ such that $\sigma_\ell \in \Sigma_\ell^{\ell+1}$
if $\kappa_{\ell+1} = \kappa'_{\ell+1}$ **then**
break {ignore contributions of internal facets $\sigma_l \notin \partial K_l^{l+1} \cap \partial {K'}_\ell^{\ell+1}$}
else
$\mathbf{A}^{\mathrm{CS}_\sigma}_{\kappa_{\ell+1},\kappa'_{\ell+1}} = \mathbf{M}_{\kappa_{\ell+1},\kappa_\ell}\ (\mathbf{A}^{\mathrm{CS}_\sigma}_{\kappa_\ell,\kappa'_\ell})\ \mathbf{M}^T_{\kappa'_{\ell+1},\kappa'_\ell}$
$\mathbf{A}^{\mathrm{CS}_\sigma}_{\kappa'_{\ell+1},\kappa_{\ell+1}} = \mathbf{M}_{\kappa'_{\ell+1},\kappa'_\ell}\ (\mathbf{A}^{\mathrm{CS}_\sigma}_{\kappa'_\ell,\kappa_\ell})\ \mathbf{M}^T_{\kappa_{\ell+1},\kappa_\ell}$
$\mathbf{A}^{\mathrm{CS}_\sigma}_{\kappa_{\ell+1},\kappa_{\ell+1}} = \mathbf{M}_{\kappa_{\ell+1},\kappa_\ell}\ (\mathbf{A}^{\mathrm{CS}_\sigma}_{\kappa_\ell,\kappa_\ell})\ \mathbf{M}^T_{\kappa_{\ell+1},\kappa_\ell}$
$\mathbf{A}^{\mathrm{CS}_\sigma}_{\kappa'_{\ell+1},\kappa'_{\ell+1}} = \mathbf{M}_{\kappa'_{\ell+1},\kappa'_\ell}\ (\mathbf{A}^{\mathrm{CS}_\sigma}_{\kappa'_\ell,\kappa'_\ell})\ \mathbf{M}^T_{\kappa'_{\ell+1},\kappa'_\ell}$
$\mathbf{A}^{\mathrm{STB}}_{\kappa_{\ell+1},\kappa'_{\ell+1}} = \mathcal{H}^{\sigma_{\ell+1}}_{\sigma_\ell}\left(\mathbf{M}_{\kappa_{\ell+1},\kappa_\ell}\ (\mathbf{A}^{\mathrm{STB}}_{\kappa_\ell,\kappa'_\ell})\ \mathbf{M}^T_{\kappa'_{\ell+1},\kappa'_\ell}\right)$
$\mathbf{A}^{\mathrm{STB}}_{\kappa'_{\ell+1},\kappa_{\ell+1}} = \mathcal{H}^{\sigma_{\ell+1}}_{\sigma_\ell}\left(\mathbf{M}_{\kappa'_{\ell+1},\kappa'_\ell}\ (\mathbf{A}^{\mathrm{STB}}_{\kappa'_\ell,\kappa_\ell})\ \mathbf{M}^T_{\kappa_{\ell+1},\kappa_\ell}\right)$
$\mathbf{A}^{\mathrm{STB}}_{\kappa_{\ell+1},\kappa_{\ell+1}} = \mathcal{H}^{\sigma_{\ell+1}}_{\sigma_\ell}\left(\mathbf{M}_{\kappa_{\ell+1},\kappa_\ell}\ (\mathbf{A}^{\mathrm{STB}}_{\kappa_\ell,\kappa_\ell})\ \mathbf{M}^T_{\kappa_{\ell+1},\kappa_\ell}\right)$
$\mathbf{A}^{\mathrm{STB}}_{\kappa'_{\ell+1},\kappa'_{\ell+1}} = \mathcal{H}^{\sigma_{\ell+1}}_{\sigma_\ell}\left(\mathbf{M}_{\kappa'_{\ell+1},\kappa'_\ell}\ (\mathbf{A}^{\mathrm{STB}}_{\kappa'_\ell,\kappa'_\ell})\ \mathbf{M}^T_{\kappa'_{\ell+1},\kappa'_\ell}\right)$
$\mathbf{A}_{\kappa_{\ell+1},\kappa'_{\ell+1}}$ += $\left(\mathbf{A}^{\mathrm{CS}_\sigma}_{\kappa_{\ell+1},\kappa'_{\ell+1}} + \mathbf{A}^{\mathrm{STB}}_{\kappa_{\ell+1},\kappa'_{\ell+1}}\right)$
$\mathbf{A}_{\kappa'_{\ell+1},\kappa_{\ell+1}}$ += $\left(\mathbf{A}^{\mathrm{CS}_\sigma}_{\kappa'_{\ell+1},\kappa_{\ell+1}} + \mathbf{A}^{\mathrm{STB}}_{\kappa'_{\ell+1},\kappa_{\ell+1}}\right)$
$\mathbf{A}_{\kappa_{\ell+1},\kappa_{\ell+1}}$ += $\left(\mathbf{A}^{\mathrm{CS}_\sigma}_{\kappa_{\ell+1},\kappa_{\ell+1}} + \mathbf{A}^{\mathrm{STB}}_{\kappa_{\ell+1},\kappa_{\ell+1}}\right)$
$\mathbf{A}_{\kappa'_{\ell+1},\kappa'_{\ell+1}}$ += $\left(\mathbf{A}^{\mathrm{CS}_\sigma}_{\kappa'_{\ell+1},\kappa'_{\ell+1}} + \mathbf{A}^{\mathrm{STB}}_{\kappa'_{\ell+1},\kappa'_{\ell+1}}\right)$
end if
end for
end for

References

[1] M. Benzi, G. H. Golub, J. Liesen, Numerical solution of saddle point problems, ACTA NUMERICA 14 (2005) 1–137.

[2] K. J. Fidkowski, T. A. Oliver, J. Lu, D. L. Darmofal, p-multigrid solution of high-order discontinuous Galerkin discretizations of the compressible Navier-Stokes equations, J. Comput. Phys. 207 (1) (2005) 92–113. doi:10.1016/j.jcp.2005.01.005.

[3] C. R. Nastase, D. J. Mavriplis, High-order discontinuous Galerkin methods using an hp-multigrid approach, Journal of Computational Physics 213 (1) (2006) 330 – 357. doi:10.1016/j.jcp.2005.08.022.

[4] F. Bassi, A. Ghidoni, S. Rebay, P. Tesini, High-order accurate p-multigrid discontinuous Galerkin solution of the Euler equations, International Journal for Numerical Methods in Fluids 60 (8) (2009) 847–865. doi:10.1002/fld.1917.

[5] K. Shahbazi, D. J. Mavriplis, N. K. Burgess, Multigrid algorithms for high-order discontinuous Galerkin discretizations of the compressible Navier-Stokes equations, Journal of Computational Physics 228 (21) (2009) 7917–7940. doi:10.1016/j.jcp.2009.07.013.

[6] J. Gopalakrishnan, G. Kanschat, A multilevel discontinuous Galerkin method, Numerische Mathematik 95 (3) (2003) 527–550. doi:10.1007/s002110200392.

[7] S. Brenner, J. Cui, T. Gudi, L.-Y. Sung, Multigrid algorithms for symmetric discontinuous Galerkin methods on graded meshes, Numerische Mathematik 119 (1) (2011) 21–47. doi:10.1007/s00211-011-0379-y.

[8] P. F. Antonietti, M. Sarti, M. Verani, Multigrid algorithms for hp-discontinuous Galerkin discretizations of elliptic problems, SIAM Journal on Numerical Analysis 53 (1) (2015) 598–618. doi:10.1137/130947015.

[9] F. Prill, M. Lukáčová-Medvidová, R. Hartmann, Smoothed aggregation multigrid for the Discontinuous Galerkin method, SIAM Journal on Scientific Computing 31 (5) (2009) 3503–3528. doi:10.1137/080728457.

[10] P. F. Antonietti, S. Giani, P. Houston, hp-version composite discontinuous Galerkin methods for elliptic problems on complicated domains, SIAM Journal on Scientific Computing 35 (3) (2013) A1417–A1439. doi:10.1137/120877246.

[11] P. F. Antonietti, S. Giani, P. Houston, Domain decomposition preconditioners for discontinuous Galerkin methods for elliptic problems on complicated domains, Journal of Scientific Computing 60 (1) (2014) 203–227. doi:10.1007/s10915-013-9792-y.

[12] P. F. Antonietti, P. Houston, I. Smears, A note on optimal spectral bounds for nonoverlapping domain decomposition preconditioners for hp-version discontinuous Galerkin methods, International Journal of Numerical Analysis and Modeling 13 (4) (2016) 513–524.

[13] P. F. Antonietti, P. Houston, X. Hu, M. Sarti, M. Verani, Multigrid algorithms for hp-version Interior Penalty Discontinuous Galerkin methods on polygonal and polyhedral meshes, eprint arXiv:1412.0913, submitted for publication.

[14] M. Wallraff, T. Leicht, Higher order multigrid algorithms for a discontinuous Galerkin RANS solver, in: 52nd Aerospace Sciences Meeting, no. 936 in AIAA SciTech, American Institute of Aeronautics and Astronautics, 2014, pp. 1055–1072. doi:10.2514/6.2014-0936.

[15] M. Wallraff, R. Hartmann, T. Leicht, Multigrid solver algorithms for DG methods and applications to aerodynamic flows, in: N. Kroll, C. Hirsch, F. Bassi, C. Johnston, K. Hillewaert (Eds.), IDIHOM: Industrialization of High-Order Methods - A Top-Down Approach, Vol. 128 of Notes on Numerical Fluid Mechanics and Multidisciplinary Design, Springer International Publishing, 2015, pp. 153–178. doi:10.1007/978-3-319-12886-3_9.

[16] F. Bassi, L. Botti, A. Colombo, D. A. Di Pietro, P. Tesini, On the flexibility of agglomeration based physical space discontinuous Galerkin discretizations, Journal of Computational Physics 231 (1) (2012) 45 – 65. doi:10.1016/j.jcp.2011.08.018.

[17] F. Bassi, L. Botti, A. Colombo, Agglomeration based physical frame dG discretizations: An attempt to be mesh free, Mathematical Models and Methods in Applied Sciences 24 (08) (2014) 1495–1539. doi:10.1142/S0218202514400028.

[18] A. Cangiani, E. H. Georgoulis, P. Houston, hp-version discontinuous Galerkin methods on polygonal and polyhedral meshes, Mathematical Models and Methods in Applied Sciences 24 (10) (2014) 2009–2041. doi:10.1142/S0218202514500146.

[19] S. Giani, P. Houston, hp-adaptive composite discontinuous Galerkin methods for elliptic problems on complicated domains, Numerical Methods for Partial Differential Equations 30 (4) (2014) 1342–1367. doi:10.1002/num.21872.

[20] A. Cangiani, Z. Dong, E. H. Georgoulis, P. Houston, hp-version discontinuous Galerkin methods for advection-diffusion-reaction problems on polytopic meshes, Mathematical Modelling and Numerical Analysis 50 (3) (2016) 699–725. doi:10.1051/m2an/2015059.

[21] F. Bassi, S. Rebay, G. Mariotti, S. Pedinotti, M. Savini, A high-order accurate discontinuous finite element method for inviscid and viscous turbomachinery flows, in: R. Decuypere, G. Dibelius (Eds.), Proceedings of the 2nd European Conference on Turbomachinery Fluid Dynamics and Thermodynamics, Technologisch Instituut, Antwerpen, Belgium, 1997, pp. 99–108.

[22] I. Moulitsas, G. Karypis, MGridGen/ParmGridGen, Serial/Parallel library for generating coase meshes for multigrid methods, Technical Report Version 1.0, University of Minnesota, Department of Computer Science/Army HPC Research Center, http://www-users.cs.umn.edu/∼moulitsa/software.html (2001).

[23] L. Botti, Influence of reference-to-physical frame mappings on approximation properties of discontinuous piecewise polynomial spaces, Journal of Scientific Computing 52 (3) (2012) 675–703.

[24] F. Bassi, L. Botti, A. Colombo, S. Rebay, Agglomeration based discontinuous Galerkin discretization of the Euler and Navier-Stokes equations, Computers & Fluids 61 (2012) 77–85. doi:10.1016/j.compfluid.2011.11.002.

[25] S. C. Brenner, L. R. Scott, The Mathematical Theory of Finite Element Methods, 3rd Edition, Springer-Verlag, New York–Berlin–Heidelberg, 2008.

[26] D. A. Di Pietro, A. Ern, Mathematical Aspects of Discontinuous Galerkin Methods, Vol. 69 of Maths & Applications, Springer-Verlag, 2011.

[27] F. Brezzi, G. Manzini, D. Marini, P. Pietra, A. Russo, Discontinuous Galerkin approximations for elliptic problems, Numer. Methods Partial Differential Equations 16 (2000) 365–378.

[28] D. N. Arnold, F. Brezzi, B. Cockburn, D. Marini, Unified analysis of discontinuous Galerkin methods for elliptic problems, SIAM J. Numer. Anal. 39 (5) (2002) 1749–1779.

[29] F. Bassi, A. Crivellini, D. A. Di Pietro, S. Rebay, An artificial compressibility flux for the discontinuous Galerkin solution of the incompressible Navier-Stokes equations, J. Comput. Phys. 218 (2006) 794–815.

[30] D. A. Di Pietro, Analysis of a discontinuous Galerkin approximation of the Stokes problem based on an artificial compressibility flux, International Journal for Numerical Methods in Fluids 55 (8) (2007) 793–813. doi:10.1002/fld.1495.

[31] W. L. Briggs, V. E. Henson, S. F. McCormick, A multigrid tutorial (2nd ed.), Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 2000.

[32] B. F. Smith, P. E. Bjørstad, W. Gropp, Domain Decomposition: Parallel Multilevel Methods for Elliptic Partial Differential Equations, Cambridge University Press, 2004.

[33] U. Trottenberg, C. W. Oosterlee, A. Schüller, Multigrid, Academic Press, Inc., 2001.

[34] P. F. Antonietti, B. A. de Dios, S. C. Brenner, L. yeng Sung, Schwarz methods for a preconditioned WOPSIP method for elliptic problems, Computational Methods in Applied Mathematics Comput. Methods Appl. Math. 12 (3) (2012) 241–272. doi:10.2478/cmam-2012-0021.

[35] D. Schötzau, C. Schwab, A. Toselli, Mixed hp-DGFEM for incompressible flows, SIAM Journal on Numerical Analysis 40 (6) (2002) 2171–2194. doi:10.1137/S0036142901399124.

[36] K. Shahbazi, P. F. Fischer, C. R. Ethier, A high-order Discontinuous Galerkin method for the unsteady incompressible Navier-Stokes equations, J. Comput. Phys. 222 (1) (2007) 391–407. doi:10.1016/j.jcp.2006.07.029.

[37] S. Balay, S. Abhyankar, M. F. Adams, J. Brown, P. Brune, K. Buschelman, L. Dalcin, V. Eijkhout, W. D. Gropp, D. Kaushik, M. G. Knepley, L. C. McInnes, K. Rupp, B. F. Smith, S. Zampini, H. Zhang, H. Zhang, PETSc users manual, Tech. Rep. ANL-95/11 - Revision 3.7, Argonne National Laboratory (2016).
URL http://www.mcs.anl.gov/petsc

[38] T. Toulorge, C. Geuzaine, J.-F. Remacle, J. Lambrechts, Robust untangling of curvilinear meshes, Journal of Computational Physics 254 (2013) 8 – 26. doi:10.1016/j.jcp.2013.07.022.

[39] Y. Saad, A flexible inner-outer preconditioned GMRES algorithm, SIAM Journal on Scientific Computing 14 (2) (1993) 461–469. doi:10.1137/0914028.

[40] S. Balay, S. Abhyankar, M. F. Adams, J. Brown, P. Brune, K. Buschelman, L. Dalcin, V. Eijkhout, W. D. Gropp, D. Kaushik, M. G. Knepley, L. C. McInnes, K. Rupp, B. F. Smith, S. Zampini, H. Zhang, H. Zhang, PETSc Web page, http://www.mcs.anl.gov/petsc (2016).
URL http://www.mcs.anl.gov/petsc

[41] S. Balay, W. D. Gropp, L. C. McInnes, B. F. Smith, Efficient management of parallelism in object oriented numerical software libraries, in: E. Arge, A. M. Bruaset, H. P. Langtangen (Eds.), Modern Software Tools in Scientific Computing, Birkhäuser Press, 1997, pp. 163–202.

[42] T. J. Tautges, R. Meyers, K. Merkley, C. Stimpson, C. Ernst, MOAB: a mesh-oriented database, SAND2004-1592, Sandia National Laboratories, report (Apr. 2004).

[43] G. Karypis, V. Kumar, METIS, a software package for partitioning unstructured graphs, partitioning meshes, and computing fill-reducing orderings of sparse matrices, Technical Report Version 4.0, University of Minnesota, Department of Computer Science/Army HPC Research Center (1998).

[44] C. Kelley, D. Keyes, Convergence analysis of pseudo-transient continuation, SIAM Journal on Numerical Analysis 35 (2) (1998) 508–523. doi:10.1137/S0036142996304796.

[45] W. A. Mulder, B. Van Leer, Experiments with implicit upwind methods for the Euler equations, Journal of Computational Physics 59 (2) (1985) 232–246.

[46] L. I. G. Kovasznay, Laminar flow behind a two-dimensional grid, Mathematical Proceedings of the Cambridge Philosophical Society 44 (1948) 58–62. doi:10.1017/S0305004100023999.

[47] L. Botti, A choice of forcing terms in inexact Newton iterations with application to pseudo-transient continuation for incompressible fluid flow computations, Applied Mathematics and Computation 266 (2015) 713 – 737. doi:10.1016/j.amc.2015.05.136.

[48] J. R. Womersley, Method for the calculation of velocity, rate of flow and viscous drag in arteries when the pressure gradient is known, The Journal of Physiology 127 (3) (1955) 553–563.

[49] J. L. Cezeaux, A. van Grondelle, Accuracy of the inverse Womersley

method for the calculation of hemodynamic variable, Annals of Biomedical Engineering 25 (3) (1997) 536–546.

[50] L. Antiga, M. Piccinelli, L. Botti, B. Ene-Iordache, A. Remuzzi, D. A. Steinman, An image-based modeling framework for patient-specific computational hemodynamics, Medical & Biological Engineering & Computing 46 (11) (2008) 1097–1112. doi:10.1007/s11517-008-0420-1.

[51] F. Mut, R. Aubry, R. Lhner, J. R. Cebral, Fast numerical solutions of patient-specific blood flows in 3d arterial systems, International Journal for Numerical Methods in Biomedical Engineering 26 (1) (2010) 73–85. doi:10.1002/cnm.1235.