p-Multigrid matrix-free discontinuous Galerkin solution strategies for the under-resolved simulation of incompressible turbulent flows

p-Multigrid matrix-free discontinuous Galerkin solution strategies for the under-resolved simulation of incompressible turbulent flows

M. Franciolini[a,c,*], L. Botti[b], A. Colombo[b], A. Crivellini[c]

[a] *USRA/NPP Fellow, NASA Ames Research Center, Moffett Field, CA, 94035, United States*
[b] *Dipartimento di Ingegneria e Scienze Applicate, Università degli Studi di Bergamo, 24044 Dalmine (BG), Italy*
[c] *Dipartimento di Ingegneria Industriale e Scienze Matematiche, Università Politecnica delle Marche, Ancona 60131, Italy*

Abstract

In recent years several research efforts focused on the development of high-order discontinuous Galerkin (dG) methods for scale resolving simulations of turbulent flows. Nevertheless, in the context of incompressible flow computations, the computational expense of solving large scale equation systems characterized by indefinite Jacobian matrices has often prevented the simulation of industrially-relevant computations. In this work we seek to improve the efficiency of Rosenbrock-type linearly-implicit Runge-Kutta methods by devising robust, scalable and memory-lean solution strategies. In particular, we introduce memory saving p-multigrid preconditioners coupling matrix-free and matrix-based Krylov iterative smoothers. The p-multigrid preconditioner relies on cheap element-wise block-diagonal smoothers on the fine space to reduce assembly costs and memory allocation, and ensures an adequate resolution of the coarsest space of the multigrid iteration using Additive Schwarz smoothers to obtain satisfactory convergence rates and optimal parallel efficiency of the method. In addition, the use of specifically crafted rescaled-inherited coarse operators to overcome the excess of stabilization provided by the standard inheritance of the fine space operators is explored. Extensive numerical validation is performed. The Rosenbrock formulation is applied to test cases of growing complexity: the laminar unsteady flow around a two-dimensional cylinder at $Re = 200$ and around a sphere at $Re = 300$, the transitional flow problem of the ERCOFTAC T3L test case suite with different

levels of free-stream turbulence. As proof of concept, the numerical solution of the Boeing rudimentary landing gear test case at $Re = 10^6$ is reported. A good agreement of the solutions with experimental data is documented, whereas a reduction in memory footprint of about 92% and an execution time gain of up to 3.5 is reported with respect to state-of-the-art solution strategies.

Keywords: incompressible flows, implicit LES, discontinuous Galerkin, p-multigrid, matrix-free, parallel efficiency

1. Introduction

In recent years the increasing availability of High Performance Computing (HPC) resources strongly promoted the wide spread of Large Eddy Simulation (LES) turbulence modelling approaches. In particular, Implicit LES (ILES) based on discontinuous Galerkin (dG) spatial discretizations showed very promising results due to the favourable dispersion and dissipation properties of the method [1]. The high potential of dG approximations for the under-resolved simulation of turbulent flows has been demonstrated in the literature for those moderate Reynolds numbers conditions where Reynolds-averaged Navier–Stokes (RANS) approaches are known to fall short, *e.g.*, massively separated flows [2, 3].

Research on this topic is growing fast and several efforts focused on devising efficient time integration strategies suited for massively parallel architectures. Indeed, the inherently unsteady nature of LES/ILES and the need to reduce time-to-results pose serious challenges to the achievement of cost effective scale-resolving computations and the ability to fruitfully exploit large computational facilities. In this context high-order implicit time integration schemes are attractive to overcome the strict stability limits of explicit methods when dealing with high-degree polynomial approximations, [4, 5, 6]. Nevertheless, implicit schemes require to solve large non-linear/linear systems of equations. The sparsity pattern of the global system matrix imposes the use of iterative methods, indeed the number of non-zeros scales as k^{2d}, where k is the degree of dG polynomial spaces and d is the number of dimensions of the problem. As a result high-order accurate computations for industrially relevant application turn out to be highly memory intensive and expensive from the CPU time point of view, even when employing state-of-the-art iterative solvers and modern HPC facilities.

*Corresponding author: matteo.franciolini@nasa.gov

Previous studies considered the possibility of using memory-saving implementations of the iterative solver in the context of discontinuous Galerkin discretizations. In [7], a matrix-free GMRES solver was used to solve steady compressible flows. The algorithm showed to be competitive on the overall computational efficiency for sufficiently high-order polynomial approximations when using ILU(0) and Additive Schwarz preconditioners. However, the use of full-matrix operators was still required for preconditioning purposes, and thus only a limited reduction of the overall memory footprint has been achieved. In [8] a matrix-free approach is employed in the context of several time integration strategies with applications to unsteady, laminar two-dimensional problems. Recently the use of a matrix-free implementation was explored and compared to a matrix-based approach in the context of incompressible unsteady turbulent flows, see [9, 10]. In particular the solution of the Rayleigh–Benard convection problem and turbulent channel flows at moderately high Reynolds numbers was considered coupling matrix-free with cheap element-wise Block-Jacobi (EWBJ) preconditioners. The algorithm showed considerable memory savings: being the use of off-diagonal Jacobian blocks not required for time stepping purposes and preconditioning, the overall memory footprint could be significantly reduced. Unfortunately, when dealing with stiff problems, *e.g.*, stretched elements, low Mach flows or large time step, a severe convergence degradation is observed when using EWBJ preconditioners: the solution is achieved only after a considerable number of iterations. Bearing that in mind, it is trivial to highlight that one of the main challenges to obtain a memory efficient implicit solution strategy for complex unsteady flow problems is the implementation of an efficient and memory saving preconditioning method to be coupled to matrix-free iterative solvers. For example, in [11] the use of a matrix-free implementation is coupled with preconditioner operators expressed in separable tensor product form, whose arithmetic complexity scales more favourably with the order of accuracy of the scheme than a standard block operator. Other implementations in the same line exist in literature, see for example [12], where the best Kronecker product approximation of the block diagonal portion of the Jacobian is solved through the use of a matrix-free Singular Value Decomposition. In [13] the same objective is obtained through the solution of an optimization problem. Despite being memory saving and capable of reducing the computational complexity of the algorithm due to the use of tensor product matrices, the main drawback of those strategies is the fact that they are based on approximations of the EWBJ preconditioner, and they fail to solve efficiently complex flow problems involving stiff

computational meshes.

On the other hand, multilevel methods have been considered in the past as an efficient way to solve both linear and non-linear problems arising from high-order discontinuous Galerkin discretizations. Such methods were first proposed in a dG context by Helenbrook *et al.* [14, 15], Bassi and Rebay [16], Fidkowski *et al.* [17]. Those authors focused on the analysis of a *p*-multigrid (*p*-MG) non-linear solver, proving convergence properties and performance in the context of compressible flows using element- or line-Jacobi smoothing. In particular, Ref. [15] analyses the effects of different coarse spaces, *i.e.*, those obtained from the re-discretization of the problem (inherited) from those generated through Galerkin projection (non-inherited). Several authors also considered multigrid operators built on agglomerated coarse grids, such as *h*-multigrid, see for example [18, 19, 20]. The possibility of using multigrid operators as a preconditioner of an iterative solver was also explored in the context of steady compressible flows, see for example [21, 22]. In these works, the algorithm is reported as the most efficient and scalable if compared to single-grid preconditioners, and a large reduction in the number of iteration to reach convergence was achieved. More recently, an *h*-multigrid preconditioner was proposed in [23] in the context of steady and unsteady incompressible flows. In this latter work a specific treatment for inherited dG discrezations of viscous terms on coarse levels was introduced, significantly improving the performance of the multigrid iteration.

The present work aims to devise a memory saving preconditioning strategy to be coupled with a matrix-free iterative solver for the solution of complex flow problems, extending and generalizing the techniques proposed in [9, 7, 24] to deal with stiff unsteady turbulent flow problems. The time-accurate numerical framework relies on linearly-implicit Runge–Kutta schemes of the Rosenbrock type. This class of schemes requires the solution of a linear systems at each stage while the matrix is assembled only once per time step. For the linear systems solution we rely on a matrix-free implementation of the Flexible GMRES (FGMRES) method, coupled with a memory saving *p*-MG preconditioner. In particular, the *p*-multigrid iteration is built using a memory saving smoother on finest level, and standard matrix-based GMRES smothers on coarse levels. The numerical experiments show that this technique allows to retain the memory footprint reduction presented in [9], while improving the computational efficiency on stiff problems. Finally, the use of a rescaled-inherited approach for the coarse space operators proposed in [23] in the context of *h*-multigrid is here assessed for the *p* version of the multigrid solver on the iterative

performance. While the use of rescaled coarse spaces is recommended to maintain acceptable convergence rates of h-multigrid solvers, smaller advantages have been observed in our experiments, which vanish for convection dominated cases such as the under-resolved simulations of turbulent flows.

The paper presents an extensive validation of the numerical strategy on test cases involving high-order accurate ILES of complex flow configurations using unstructured meshes made of severely stretched and curved elements. The effectiveness of the proposed coupling between a matrix-free linear solver and a matrix-free p-multigrid preconditioner is proved by comparing computational time, memory footprint of the solver as well as the algorithmic scalability of the preconditioning strategy on HPC facilities using a domain decomposition parallelization method.

The paper is structured as follows. Section 2 describes the space and time discretization and presents the multigrid framework here employed, with particular attention to the coarse spaces assembly and the intergrid transfer operators. Section 3 reports a thorough assessment of the stabilization scaling on test cases of growing complexity involving unsteady flows: the unsteady flow over a two dimensional cylinder at $Re = 200$, and the unsteady flow over a sphere at $Re = 300$. Section 4.1 demonstrates the advantages of using the proposed solver for the solution of the T3L1 flow problem of the ERCOFTAC test case suites, *i.e.*, the incompressible turbulent flow over a rounded leading-edge flat plate at $Re = 3450$ with different levels of free-stream turbulence. After a brief physical discussion of the solution accuracy, significant memory savings as well as improvements in computational efficiency with respect to matrix-based methods are documented. As proof of concept, the solution of the Boeing rudimentary landing gear test case at $Re = 10^6$ is reported in Section 4.2, including a favourable agreement with experimental data.

2. The numerical framework

In this section the space and time discretizations of the incompressible Navier–Stokes (INS) equations are briefly introduced together with a detailed description of the main building blocks of the p-multigrid preconditioner.

2.1. The dG discretization

We consider the unsteady INS equations in conservation form in a fixed Cartesian reference frame,

$$\partial_t \mathbf{u} + \boldsymbol{\nabla} \cdot (\mathbf{u} \otimes \mathbf{u} + p\mathbf{I}) - \nu \boldsymbol{\nabla} \cdot \left[\left(\nabla \otimes \mathbf{u} + (\nabla \otimes \mathbf{u})^t \right) - \frac{2}{3} (\nabla \cdot \mathbf{u}) \mathbf{I} \right] = 0 \qquad \text{in } \Omega \times (0, t_F), \tag{1a}$$

$$\boldsymbol{\nabla} \cdot \mathbf{u} = 0 \qquad \text{in } \Omega \times (0, t_F). \tag{1b}$$

equipped with suitable boundary conditions and initial value, where $\mathbf{u} \in \mathbb{R}^d$ is the velocity vector, $p \in \mathbb{R}$ is the pressure, ν denotes the (constant) viscosity, t_F is the final simulation time, $\mathbf{n}$ is the unit outward normal to $\partial\Omega$ and $\mathbf{I} = \delta_{ij}\, \mathbf{e}_i \otimes \mathbf{e}_j$, $i, j = 1, ..., d$, is the identity matrix. The density has been assumed to be uniform and equal to one and the Stokes hypothesis is used for the definition of viscous stresses. Introducing the convective and viscous flux functions

$$\begin{aligned} \mathbf{F}^{\nu}\left(\tfrac{\partial u_i}{\partial x_j}\right) &= \nu \left(\nabla \otimes \mathbf{u} + (\nabla \otimes \mathbf{u})^t \right) - \tfrac{2}{3} \nu (\nabla \cdot \mathbf{u}) \mathbf{I} &&= \nu \left(\frac{\partial u_j}{\partial x_i} + \frac{\partial u_i}{\partial x_j} - \frac{2}{3} \frac{\partial u_k}{\partial x_k} \delta_{ij} \right) \mathbf{e}_i \otimes \mathbf{e}_j \\ \mathbf{F}^{c}(u_i, p) &= \mathbf{u} \otimes \mathbf{u} + p\mathbf{I} &&= \left(u_i\, u_j + p \delta_{ij} \right) \mathbf{e}_i \otimes \mathbf{e}_j \end{aligned} \tag{2}$$

Eqs. (1a)-(1b) can be compactly rewritten in integral form as

$$\int_\Omega \partial_t \mathbf{u} + \int_\Omega \nabla \cdot (\mathbf{F}^c - \mathbf{F}^v) = \mathbf{0}, \tag{3a}$$

$$\int_\Omega \nabla \cdot \mathbf{u} = 0. \tag{3b}$$

In order to define the dG discretization we introduce a triangulation $\mathcal{T}_h$ of the computational domain Ω, that is the collection of disjoint mesh elements $\kappa \in \mathcal{T}_h$ such that $\bigcup_{\kappa \in \mathcal{T}_h} \kappa = \Omega_h$, where Ω_h is a suitable approximation of Ω. The mesh skeleton $\mathcal{F}_h$ is the collection of mesh faces σ. Internal faces $\sigma \in \mathcal{F}_h^{\mathrm{i}}$ are defined as the intersection of the boundary of two neighboring elements: $\sigma = \partial\kappa \bigcap \partial\kappa'$ with $\kappa \neq \kappa'$. Boundary faces $\sigma \in \mathcal{F}_h^{\mathrm{b}}$ reads $\sigma = \partial\kappa \bigcap \partial\Omega_h$. Clearly $\mathcal{F}_h = \mathcal{F}_h^{\mathrm{i}} \cup \mathcal{F}_h^{\mathrm{b}}$. Each component of the velocity vector and the pressure is sought (for $0 < t < t_F$) in the so called *broken polynomial spaces* defined over $\mathcal{T}_h$

$$\mathbb{P}_d^k(\mathcal{T}_h) = \left\{ v_h \in L^2(\Omega_h) \mid \forall \kappa \in \mathcal{T}_h, v_h|_\kappa \in \mathbb{P}_d^k(\kappa) \right\} \tag{4}$$

where $\mathbb{P}_d^k(\kappa)$ is the space of polynomial functions in d variables and total degree k defined over κ. Since no continuity requirements are enforced at inter-element boundaries, v_h admits two-valued traces on the partition of mesh skeleton $\mathcal{F}_h^{\mathrm{i}}$. Accordingly we introduce average and jump

operators over internal faces

$$\textbf{Average}: \{\!\{v_h\}\!\}_\sigma = \frac{1}{2}\left(v_h|_\kappa + v_h|_{\kappa'}\right), \qquad \textbf{Jump}: [\![v_h]\!]_\sigma = \left(v_h|_\kappa - v_h|_{\kappa'}\right). \tag{5}$$

Specific definitions of averages and jumps will be introduced over boundary faces to take into account Dirichlet and Neumann boundary conditions.

The dG discretization of the Navier-Stokes equations reads: find $(\boldsymbol{u}_h, p_h) \in [\mathbb{P}^k(\mathcal{T}_h)]^d \times \mathbb{P}^k(\mathcal{T}_h)$ such that, for all $(\boldsymbol{v}_h, q_h) \in [\mathbb{P}^k(\mathcal{T}_h)]^d \times \mathbb{P}^k(\mathcal{T}_h)$:

$$\begin{aligned}
\sum_{\kappa\in\mathcal{T}_h}\int_\kappa \partial_t \boldsymbol{u}_h\cdot\boldsymbol{v}_h \quad &- \sum_{\kappa\in\mathcal{T}_h}\int_\kappa \left(\mathbf{F}_h^c - \widetilde{\mathbf{F}}_h^v\right) : \nabla_h \boldsymbol{v}_h \quad + \sum_{\sigma\in\mathcal{F}_h}\int_\sigma \mathbf{n}^\sigma\cdot\left(\widehat{\mathbf{F}}_h^c - \widehat{\mathbf{F}}_h^v\right)\cdot[\![\boldsymbol{v}_h]\!] \quad = 0,\\
&- \sum_{\kappa\in\mathcal{T}_h}\int_\kappa \boldsymbol{u}_h\cdot\nabla_h q_h \qquad\quad + \sum_{\sigma\in\mathcal{F}_h}\int_\sigma \mathbf{n}^\sigma\cdot\widehat{\mathbf{u}}_h\ [\![q_h]\!] \qquad\quad = 0,
\end{aligned} \tag{6}$$

where $[\![\boldsymbol{v}_h]\!] = [\![v_{h,i}]\!]\,\mathbf{e}_i$ and $\mathbf{n}^\sigma$ is the normal vector with respect to σ. While obtaining (6) from (3) follows the standard dG FE practice (element-by-element integration by parts after having multiplied by a suitable test function), the dG method hinges on the definition of suitable numerical viscous $\widetilde{\mathbf{F}}_h^v$, $\widehat{\mathbf{F}}_h^v$ and inviscid fluxes $\widehat{\mathbf{F}}_h^c$, $\widehat{\mathbf{u}}_h$.

According to the BR1 scheme, proposed in [25], $\forall\boldsymbol{\tau}_h \in [\mathbb{P}_d^k(\mathcal{T}_h)]^d$, $v_h \in \mathbb{P}_d^k(\mathcal{T}_h)$, the consistent gradient $\mathbf{G}_h(v_h)$ is such that

$$\int_\Omega (\mathbf{G}_h(v_h) - \nabla v_h)\cdot\boldsymbol{\tau}_h = -\sum_{\sigma\in\mathcal{F}_h}\int_\sigma [\![v_h]\!]\,\{\!\{\boldsymbol{\tau}_h\}\!\}\cdot\mathbf{n}^\sigma := \sum_{\sigma\in\mathcal{F}_h}\int_\Omega \mathbf{r}^\sigma([\![v_h]\!])\cdot\boldsymbol{\tau}_h = \sum_{\kappa\in\mathcal{T}_h}\int_\kappa \mathbf{R}^\kappa(v_h)\cdot\boldsymbol{\tau}_h$$

where $\mathbf{r}^\sigma([\![v_h]\!]) : \mathbb{P}_d^k(\sigma) \to [\mathbb{P}_d^k(\mathcal{T}_h)]^d$ is the local lifting operator, $\mathbf{R}^\kappa(v_h) := \sum_{\sigma\in\mathcal{F}_{\partial\kappa}} \mathbf{r}^\sigma([\![v_h]\!])$ is the elemental lifting operator and $\mathcal{F}_{\partial\kappa}$ is the set of faces belonging to $\partial\kappa$. In this work we rely on the BR2 scheme, introduced to reduce the stencil of the BR1 discretization and analyzed in the context of the Poisson problem by [26] and [27]. The BR2 viscous fluxes are functions of elemental spatial derivatives corrected by suitable lifting operator contributions

$$\widetilde{\mathbf{F}}_h^v = \mathbf{F}^v\left(\frac{\partial u_{h,i}}{\partial x_j} - R_j^\kappa(u_{h,i})\right), \quad \text{and} \quad \widehat{\mathbf{F}}_h^v = \mathbf{F}^v\left(\left\{\!\!\left\{\frac{\partial u_{h,i}}{\partial x_j} - \eta_\sigma r_j^\sigma([\![u_{h,i}]\!])\right\}\!\!\right\}\right), \tag{7}$$

$\widehat{\mathbf{F}}^v$ ensures consistency and stability of the scheme and $\widetilde{\mathbf{F}}^v$ guarantees the symmetry of the formulation. As proved by Brezzi *et al.* [26], coercivity for the BR2 discretization of the Laplace equation holds provided that η_σ is greater than the maximum number of faces of the elements sharing σ. The inviscid numerical fluxes of the dG discretization result from the exact solution of local Riemann problems based on an artificial compressibility perturbation of the Euler equations, as proposed in [28].

We remark that, while for the continuous form of the INS the divergence of the velocity term in Eqs. (1a)-(1b) exactly vanishes, for the numerical problem this solenoidal constraint is satisfied at a discrete level [28, 29]. Numerical evidences, see [30], suggested that retaining the $\nabla \cdot \mathbf{u}$ term in the discrete form is beneficial in terms of robustness.

2.2. Implicit time accurate integration

Time integration of Equation (6) can be presented in compact form by collecting the velocity vector and the pressure polynomial expansions in the vector $\boldsymbol{w}_h \stackrel{\text{def}}{=} (u_{h,1}, ..., u_{h,d}, p_h) \in [\mathbb{P}_d^k(\mathcal{T}_h)]^{d+1}$ and identifying the unknown vector at time t_n with $\boldsymbol{w}_h^n$, that is $\boldsymbol{w}_h^n = [\boldsymbol{u}_h(t_n), p_h(t_n)]$. Moreover, we introduce the flux functions $\widetilde{\mathbf{F}}_h(\boldsymbol{w}_h) \stackrel{\text{def}}{=} \left[\mathbf{F}_h^c - \widetilde{\mathbf{F}}_h^v, \boldsymbol{u}_h\right] \in \mathbb{R}^d \otimes \mathbb{R}^{d+1}$ and $\widehat{\mathbf{F}}_h(\boldsymbol{w}_h) \stackrel{\text{def}}{=} \left[\widehat{\mathbf{F}}_h^c - \widehat{\mathbf{F}}_h^v, \widehat{\mathbf{u}}_h\right] \in \mathbb{R}^d \otimes \mathbb{R}^{d+1}$, collecting the viscous and inviscid flux contributions. For all $\boldsymbol{w}_h, \boldsymbol{z}_h \in [\mathbb{P}_d^k(\mathcal{T}_h)]^{d+1}$, we define the residual of the dG spatial discretization in (6) as follows

$$f_h(\boldsymbol{w}_h, \boldsymbol{z}_h) = -\sum_{\kappa \in \mathcal{T}_h} \int_\kappa \sum_{i=1}^{d+1} \sum_{p=1}^{d} \widetilde{F}_{p,i}(\boldsymbol{w}) \frac{\partial z_i}{\partial x_p} + \sum_{\sigma \in \mathcal{F}_h} \int_\sigma \sum_{i=1}^{d+1} \sum_{p=1}^{d} n_p^\sigma \widehat{F}_{p,i}(\boldsymbol{w}) \, [\![z_i]\!] \,, \tag{8}$$

where we dropped the mesh step size subscript when working in index notation. For all $\boldsymbol{w}_h, \delta\boldsymbol{w}_h, \boldsymbol{z}_h \in [\mathbb{P}_d^k(\mathcal{T}_h)]^{d+1}$, the linearization of the residual reads

$$j_h(\boldsymbol{w}_h, \delta\boldsymbol{w}_h, \boldsymbol{z}_h) = \frac{\partial f_h(\boldsymbol{w}_h, \boldsymbol{z}_h)}{\partial \boldsymbol{w}_h} \delta\boldsymbol{w}_h. \tag{9}$$

In particular we distinguish the inviscid $j_h^{\text{i}v}(\boldsymbol{w}_h, \delta\boldsymbol{w}_h, \boldsymbol{z}_h)$ and the viscous $j_h^v(\delta\boldsymbol{w}_h, \boldsymbol{z}_h)$ contributions

$$j_h^{\text{i}v}(\boldsymbol{w}_h, \delta\boldsymbol{w}_h, \boldsymbol{z}_h) = -\sum_{\kappa \in \mathcal{T}_h} \int_\kappa \sum_{i,j=1}^{d+1} \sum_{p=1}^{d} \frac{\partial \widetilde{F}_{p,i}}{\partial w_j}(\boldsymbol{w}_h) \, \delta w_j \, \frac{\partial z_i}{\partial x_p} + \sum_{\sigma \in \mathcal{F}_h} \int_\sigma \sum_{i,j=1}^{d+1} \sum_{p=1}^{d} n_p^\sigma \frac{\partial \widehat{F}_{p,i}}{\partial w_j}(\boldsymbol{w}_h) \, \delta w_j \, [\![z_i]\!] \,, \tag{10}$$

$$\begin{aligned}[1] j_h^v(\delta\boldsymbol{w}_h, \boldsymbol{z}_h) = &-\sum_{\kappa \in \mathcal{T}_h} \int_\kappa \sum_{i,j=1}^{d+1} \sum_{p,q=1}^{d} \frac{\partial \widetilde{F}_{p,i}}{\partial\left(\partial w_j/\partial x_q - R_q^\kappa(w_j)\right)} \left(\frac{\partial(\delta w_j)}{\partial x_q} - R_q^\kappa(\delta w_j)\right) \frac{\partial z_i}{\partial x_p} + \\ &+ \sum_{\sigma \in \mathcal{F}_h} \int_\sigma \sum_{i,j=1}^{d+1} \sum_{p,q=1}^{d} n_p^\sigma \frac{\partial \widehat{F}_{p,i}}{\partial\left(\partial w_j/\partial x_q - \eta_\sigma r_q^\sigma(w_j)\right)} \left\{\!\!\left\{ \frac{\partial(\delta w_j)}{\partial x_q} - \eta_\sigma \, r_q^\sigma\left([\![\delta w_j]\!]\right) \right\}\!\!\right\} [\![z_i]\!] \,. \end{aligned} \tag{11}$$

Note that, since $\mathbf{F}^v$ is a linear function, (11) is a bilinear form, while, by abuse of notation, (10) is a bilinear (resp. trilinear) when $F_{p,i}^c(\boldsymbol{w})$ is a linear (resp. non-linear) function of w_j.

In this work time integration is performed via the multi-stage linearly implicit (Rosenbrock-type) Runge-Kutta method. As an appealing feature the method requires the solution of a linear system at each stage $s = \{1, \cdots, n_s\}$, while the Jacobian matrix needs to be assembled only once

per time step. Prior to introducing the formulation for the temporal discretization we define the following mass bilinear form: for all $\boldsymbol{w}_h, \boldsymbol{z}_h \in [\mathbb{P}^k(\mathcal{T}_h)]^{d+1}$

$$m_h(\boldsymbol{w}_h, \boldsymbol{z}_h) = \sum_{\kappa \in \mathcal{T}_h} \int_\kappa \sum_{i=1}^{d} w_i \, z_i. \tag{12}$$

Given the initial condition $\boldsymbol{w}_h^0 = \boldsymbol{w}_h(t=0) \in [\mathbb{P}^k(\mathcal{T}_h)]^{d+1}$ we define the sequence $\boldsymbol{w}_h^{n+1}$ iteratively by means of the Rosenbrock scheme as described in Algorithm 1, where γ, a_{ij}, c_{ij} and m_i are real coefficients proper of the Rosenbrock scheme and $\delta\mathbf{w}_h^s$, with $s = \{1, \cdots, n_s\}$, the solutions at each stage of the scheme that are properly combined to compute the solution $\boldsymbol{w}_h^{n+1}$ at the next time level.

Algorithm 1 Multi-stage linearly implicit (Rosenbrock-type) Runge-Kutta method

set $\boldsymbol{w}_h^n = \boldsymbol{w}_h^0$, $n_F = \dfrac{t_F}{\delta t}$
for $n = 0, ..., n_F$ **do**
  for $s = 1, ..., n_s$ **do**
    set $\delta \boldsymbol{p}_h = \mathbf{0} \ \wedge \ \delta \boldsymbol{q}_h = \mathbf{0}$
    for $o = 1, ..., s-1$ **do**
      $\delta \boldsymbol{p}_h$ += $\mathsf{a}_{s,o} \, \delta \boldsymbol{w}_h^o$
      $\delta \boldsymbol{q}_h$ += $\mathsf{c}_{s,o} \, \delta \boldsymbol{w}_h^o$
    end for
    find $\delta \boldsymbol{w}_h^s \in [\mathbb{P}^k(\mathcal{T}_h)]^{d+1}$ such that, for all $\boldsymbol{z}_h \in [\mathbb{P}^k(\mathcal{T}_h)]^{d+1}$

$$\frac{1}{\gamma \delta t} m_h(\delta \boldsymbol{w}_h^s, \boldsymbol{z}_h) + j_h(\boldsymbol{w}_h^n, \delta \boldsymbol{w}_h^s, \boldsymbol{z}_h) = -f_h(\boldsymbol{w}_h^n + \delta \boldsymbol{p}_h, \boldsymbol{z}_h) - \frac{1}{\delta t} m_h(\delta \boldsymbol{q}_h, \boldsymbol{z}_h) \tag{13}$$

  end for
  for $o = 1, ..., s$ **do**
    set $\boldsymbol{w}_h^{n+1} = \boldsymbol{w}_h^n + \mathsf{m}_o \delta \boldsymbol{w}_h^o$
  end for
end for

The Rosenbrock time marching strategy in Algorithm 1 advances the solution in time by repeatedly solving the linearized system of equations (13), once for each stage of the Runge-Kutta method. Introducing the Jacobian and mass matrix operators

$$\begin{aligned} (\boldsymbol{J}_h \, \delta \boldsymbol{w}_h, \boldsymbol{z}_h)_{L^2(\Omega)} &= j_h(\boldsymbol{w}_h, \delta \boldsymbol{w}_h, \boldsymbol{z}_h) \quad \forall \, \boldsymbol{w}_h, \delta \boldsymbol{w}_h, \boldsymbol{z}_h \in [\mathbb{P}_d^{\kappa_\ell}(\mathcal{T}_h)]^{d+1}, \\ (\boldsymbol{M}_h \, \delta \boldsymbol{w}_h, \boldsymbol{z}_h)_{L^2(\Omega)} &= m_h(\delta \boldsymbol{w}_h, \boldsymbol{z}_h) \qquad \forall \, \delta \boldsymbol{w}_h, \boldsymbol{z}_h \in [\mathbb{P}_d^{\kappa_\ell}(\mathcal{T}_h)]^{d+1}, \end{aligned} \tag{14}$$

the equation system (13) can be compactly rewritten as follows:

$$\boldsymbol{G}_h \, \delta \boldsymbol{w}_h = \boldsymbol{g}_h \tag{15}$$

where $\boldsymbol{G}_h = \frac{1}{\gamma\delta t}\boldsymbol{M}_h + \boldsymbol{J}_h$ is the global matrix operator, and $\delta\boldsymbol{w}_h, \boldsymbol{g}_h \in [\mathbb{P}_d^k(\mathcal{T}_h)]^{d+1}$ are the unknown polynomial function and the right-hand side arising from the linearly-implicit Runge-Kutta time discretization, respectively. In this work the four stage, order three (ROSI2PW) scheme of Rang and Angermann [31] was employed. This scheme preserves its formal accuracy when applied to the system of DAEs arising form the spatial discretization of the INS equations as demonstrated in [9]. All the linear systems are solved using a flexible GMRES iterative scheme. The solution process terminates when a relative drop in the residual reaches 10^{-5}, which was found adequate for all the numerical experiments presented in this work. In order to report comparisons of different preconditioning strategies, the right preconditioning method is employed such that the residual drop definition is not affected by the choice of the operator.

2.3. Matrix-free iterative solver

The flexible GMRES can be implemented matrix-free following the approach of [32], where the product between the primal Jacobian and the defect vector is approximated by its first order Taylor expansion. Given $\boldsymbol{w}_h, \boldsymbol{d}_h \in [\mathbb{P}_d^{\kappa_\ell}(\mathcal{T}_h)]^{d+1}$, the Jacobian trilinear form can be expressed as

$$j_h(\boldsymbol{w}_h^n, \boldsymbol{d}_h, \boldsymbol{z}_h) = \frac{1}{\Delta}\left(f(\boldsymbol{w}_h^n + \Delta\,\boldsymbol{d}_h, \boldsymbol{z}_h) - f(\boldsymbol{w}_h^n, \boldsymbol{z}_h)\right),\ \forall \boldsymbol{z}_h \in [\mathbb{P}_d^{\kappa_\ell}(\mathcal{T}_h)]^{d+1}, \tag{16}$$

which involves bilinear form evaluations. According to [33],

$$\Delta = \epsilon \frac{\sqrt{1 + ||\boldsymbol{w}_h^n||_{L^2(\Omega)}}}{||\boldsymbol{d}_h||_{L^2(\Omega)}}, \tag{17}$$

with $\epsilon = 10^{-9}$ for all the computations [9, 10, 34]. We remark that the use of (16) does not change the behaviour of the iterative solver for relative tolerances of practical engineering interest, *i.e.* when those are greater than the numerical perturbation ϵ, and does not increase the cost-per-iteration at high order of accuracy, since the algorithm complexity of the residual evaluation scales equally to that of a matrix-vector product with the order of polynomial approximation. Since the global system matrix is not required for the time integration, the Jacobian matrix needs to be assembled for preconditioning purposes only, and such flexibility can be exploited to reduce the matrix-assembly time and memory footprint, for example by evaluating only parts of the Jacobian blocks and/or reusing those blocks for several successive iterations. See [9] for further details on the matrix-free implementation.

As preconditioners for GMRES iterators we consider the following options:

1. ASM(i,ILU(j)) - Additive Schwarz domain decomposition Method (ASM) preconditioners with i levels of overlap between sub-domains and an ILU decomposition for each sub-domain matrix with j levels of fill. In the overlap region, the residuals on the ghost points are used for the application of the incomplete LU factors, while the result values in the ghost points are discarded;

2. Block-Jacobi (BJ) or ASM(0,ILU(0)) - ASM preconditioner with no overlap between sub-domains (mesh partitions) and an ILU(0) decomposition for each sub-domain matrix with same level of fill of the original matrix;

3. EWBJ - Element-wise block Jacobi, equivalent to the BJ preconditioner applied to the block-diagonal iteration matrix. This preconditioner is implemented as the element-by-element LU factorization of the diagonal blocks.

Note that in serial computations ASM(i,ILU(j)) and BJ fall back to ILU(j) and ILU(0), respectively. ASM and BJ performance differ when the computation is performed in parallel, depending on the number of sub-domains. While efficiency of BJ decreases while increasing the number of sub-domains, ASM seeks to heal the convergence degradation at the expense of an increased memory footprint of the solver as the number of partitions rises, as part of the global matrix non-zeros entries are replicated in neighboring sub-domains. Conversely, the BJ preconditioner falls back to the EWBJ in the case of one element per partition. EWBJ has optimal scalability properties, involving local-to-each-element operations. In addition, when using a matrix-free iterative solver, only the use of EWBJ leads to a memory saving, as it allows to skip the computation of the off-diagonal contributions and therefore to reduce the matrix-assembly computation time. The code relies on the PETSc library to handle linear solvers and parallelism [35].

For the sake of compactness, in the remaining of the paper we will denote a solver-preconditioner couple following the convention:

SOLVER(MatVecOpt)[PREC(Opt)].

The MatVecOpt label describes how the matrix-vector products are performed, *i.e.* in a matrix-free (MF) or matrix-based (MB) fashion, while PREC(Opt) describes the type of preconditioning employed.

2.4. Multigrid preconditioners

To increase the performance of linear system solutions on stiff space discretizations, we investigate the use of multigrid preconditioning approaches to solve the global equation system (15). The basic idea is to exploit iterative solvers to smooth-out the high-frequency component of the error with respect to the unknown exact solution. Indeed, being iterative solvers not effective at damping low-frequency error components, the iterative solution of coarser problems is exploited to circumvent this issue, shifting the low-frequency modes towards the upper side of the spectrum. This simple and effective strategy allows to obtain satisfactory rates of convergence all along the iterative process.

As the work aims at obtaining solutions with high-order polynomials on rather few and possibly curved mesh elements, and targets the use of the solver on large HPC facilities, we build coarse spaces by reducing the degree of polynomial approximation of the solution of the dG discretization with respect to the original problem of degree k, commonly referred in the literature as p-multigrid method. The strategy show some advantages over h-multigrid approaches on agglomerated mesh elements other than the ease of implementation [36], as the cost of applying intergrid transfer operators is almost negligible. We consider L coarse levels spanned by the index $\ell = 1, ..., L$ and indicate the fine and coarse levels with $\ell = 0$ and $\ell = L$, respectively. The polynomial degree of level ℓ is k_ℓ and the polynomial degrees of the coarse levels are chosen such that $k_\ell < k_{\ell-1}$, $l = 1, ..., L$, with $k_0 = k$. Accordingly the polynomial spaces associated to the coarse levels read $\mathbb{P}_d^{\kappa_\ell}(\mathcal{T}_h)$. The coarse problems corresponding to (15) are in the form

$$\boldsymbol{G}_\ell \, \delta \boldsymbol{w}_\ell = \boldsymbol{g}_\ell \tag{18}$$

where $\boldsymbol{G}_\ell$ is the global matrix operator on level l and $\delta \boldsymbol{w}_\ell, \boldsymbol{g}_\ell \in [\mathbb{P}_d^{\kappa_\ell}(\mathcal{T}_h)]^{d+1}$ are the unknown function and the known right-hand side, respectively.

A crucial aspect for the efficiency of the p-multigrid iteration is related to the computational cost of building coarse grid operators $\boldsymbol{G}_\ell$. While it is possible to assemble the bilinear and trilinear forms j_h, f_h and m_h of Section 2.2 on each level ℓ with the corresponding polynomial functions $\boldsymbol{w}_\ell, \delta \boldsymbol{w}_\ell, \boldsymbol{z}_\ell \in \mathbb{P}_d^{k_\ell}(\mathcal{T}_h)$, significantly better performances are achievable by *restricting* the fine grid operator by means of so called Galerkin projections. The former and the latter strategies are named non-inherited and inherited p-multigrid, respectively. As will be clear in what follows the construction of coarse operators is trivial when polynomial expansions are based on hierarchical orthonormal modal basis functions.

2.4.1. Restriction and prolongation operators

In this section we describe the prolongation and restriction operators required to map polynomial functions on finer and coarser levels, respectively.

Since $\mathbb{P}_d^{\kappa_\ell}(\mathcal{T}_h) \supset \mathbb{P}_d^{\kappa_{\ell+1}}(\mathcal{T}_h)$, the *prolongation* operator $\mathcal{I}_{\ell+1}^{\ell} : \mathbb{P}_d^{\kappa_{\ell+1}}(\mathcal{T}_h) \rightarrow \mathbb{P}_d^{\kappa_\ell}(\mathcal{T}_h)$, is the injection operator such that

$$\mathcal{I}_{\ell+1}^{\ell} w_{\ell+1} = w_{\ell+1}, \quad \forall w_{\ell+1} \in \mathbb{P}_d^{\kappa_{\ell+1}}(\mathcal{T}_h), \tag{19}$$

which in practice computes $w_\ell \in \mathbb{P}_d^{\kappa_\ell}(\mathcal{T}_h)$ by padding $w_{\ell+1}$ with zeros on the higher-order modes. The prolongation operator from level ℓ to level 0 can be recursively defined by the composition of inter-grid prolongation operators: $\mathcal{I}_\ell^0 = \mathcal{I}_1^0 \, \mathcal{I}_2^1 \, ... \, \mathcal{I}_\ell^{\ell-1}$.

The (L^2 projection) *restriction* operator $\mathcal{I}_\ell^{\ell+1} : \mathbb{P}_d^{\kappa_\ell}(\mathcal{T}_h) \rightarrow \mathbb{P}_d^{\kappa_{\ell+1}}(\mathcal{T}_h)$, is such that

$$\sum_{\kappa \in \mathcal{T}_h} \int_\kappa (\mathcal{I}_\ell^{\ell+1} w_\ell - w_{\ell+1}) \, z_{\ell+1} = 0, \qquad \forall w_\ell \in \mathbb{P}_d^{\kappa_\ell}(\mathcal{T}_h), \ \forall z_{\ell+1} \in \mathbb{P}_d^{\kappa_{\ell+1}}(\mathcal{T}_h), \tag{20}$$

and the restriction operator from level 0 to level ℓ reads $\mathcal{I}_0^\ell = \mathcal{I}_{\ell-1}^{\ell} \, ... \, \mathcal{I}_1^2 \, \mathcal{I}_0^1$.

When applied to vector functions $\boldsymbol{w}_{\ell+1} \in [\mathbb{P}_d^{\kappa_{\ell+1}}(\mathcal{T}_h)]^{d+1}$ the interpolation operators act componentwise, *e.g.*, $\mathcal{I}_{\ell+1}^{\ell} \boldsymbol{w}_{\ell+1} = \sum_{i=1}^{d+1} \mathcal{I}_{\ell+1}^{\ell} w_i$. It is interesting to remark that using hierarchical orthonormal modal basis functions restriction and prolongation operators are trivial, in particular restriction from $\mathbb{P}_d^{\kappa_\ell}(\mathcal{T}_h)$ into $\mathbb{P}_d^{\kappa_{\ell+1}}(\mathcal{T}_h)$ is as simple as keeping the degrees of freedom of the modes of order $k \leq \kappa_{\ell+1}$ and discarding the remaining high-frequency modes.

2.5. Fine and coarse grid Jacobian operators

The non-inherited and the inherited version (denoted with superscript $\mathcal{I}$) of the inviscid and viscous Jacobian operators introduced in (10)-(11), can be defined as follows for $\ell = 1, ..., L$

$$\begin{aligned}
(\boldsymbol{J}_\ell^{!\nu} \, \delta \boldsymbol{w}_\ell, \boldsymbol{z}_\ell)_{L^2(\Omega)} &= j_\ell^{!\nu}(\boldsymbol{w}_\ell, \delta \boldsymbol{w}_\ell, \boldsymbol{z}_\ell) && \forall \, \boldsymbol{w}_\ell, \delta \boldsymbol{w}_\ell, \boldsymbol{z}_\ell \in [\mathbb{P}_d^{\kappa_\ell}(\mathcal{T}_h)]^{d+1} \\
(\boldsymbol{J}_\ell^{\nu} \, \delta \boldsymbol{w}_\ell, \boldsymbol{z}_\ell)_{L^2(\Omega)} &= j_\ell^{\nu}(\delta \boldsymbol{w}_\ell, \boldsymbol{z}_\ell) && \forall \, \delta \boldsymbol{w}_\ell, \boldsymbol{z}_\ell \in [\mathbb{P}_d^{\kappa_\ell}(\mathcal{T}_h)]^{d+1} \\
(\boldsymbol{J}_\ell^{!\nu,\mathcal{I}} \delta \boldsymbol{w}_\ell, \boldsymbol{z}_\ell)_{L^2(\Omega)} &= j_\ell^{!\nu}(\boldsymbol{w}_\ell, \mathcal{I}_\ell^0 \, \delta \boldsymbol{w}_\ell, \mathcal{I}_\ell^0 \boldsymbol{z}_\ell) && \forall \, \boldsymbol{w}_\ell, \delta \boldsymbol{w}_\ell, \boldsymbol{z}_\ell \in [\mathbb{P}_d^{\kappa_\ell}(\mathcal{T}_h)]^{d+1} \\
(\boldsymbol{J}_\ell^{\nu,\mathcal{I}} \delta \boldsymbol{w}_\ell, \boldsymbol{z}_\ell)_{L^2(\Omega)} &= j_\ell^{\nu}(\mathcal{I}_\ell^0 \, \delta \boldsymbol{w}_\ell, \mathcal{I}_\ell^0 \boldsymbol{z}_\ell) && \forall \, \delta \boldsymbol{w}_\ell, \boldsymbol{z}_\ell \in [\mathbb{P}_d^{\kappa_\ell}(\mathcal{T}_h)]^{d+1}
\end{aligned} \tag{21}$$

The main benefit of inherited algorithms is the possibility to efficiently compute coarse grid operators by means of the so called Galerkin projection, avoiding the cost of assembling bilinear and trilinear forms. The procedure is described in what follows, focusing on the benefits of using hierarchical orthonormal basis functions.

The matrix counterpart $\mathbf{J}_\ell^{\mathcal{I}}$ of the operator $\boldsymbol{J}_\ell^{\mathcal{I}} = \boldsymbol{J}_\ell^{\nu,\mathcal{I}} + \boldsymbol{J}_\ell^{!\nu,\mathcal{I}}$ is a sparse block matrix with block dimension $N_{\mathrm{DoF}}^{\kappa} = \dim\left(\mathbb{P}_d^{\kappa_\ell}(\kappa)\right)$ and total dimension $\mathrm{card}(\mathcal{T}_h)\, N_{\mathrm{DoF}}^{\kappa}\,(d{+}1)$. The matrix is composed of diagonal blocks $\mathbf{J}_{\kappa,\kappa}^{\ell,\mathcal{I}}$ and off-diagonal blocks $\mathbf{J}_{\kappa,\kappa'}^{\ell,\mathcal{I}}$, the latter taking care of the coupling between neighboring elements κ, κ' sharing a face σ. Once the fine system matrix $\mathbf{J}_0$ is assembled, the diagonal and off-diagonal blocks of the Jacobian matrix of coarse levels can be inherited recursively and matrix-free as follows

$$\mathbf{J}_{\kappa,\kappa}^{\ell+1,\mathcal{I}} = \mathbf{M}_{\ell+1,\ell}^{\kappa} \left(\mathbf{J}_{\kappa,\kappa}^{\ell,\mathcal{I}}\right) \left(\mathbf{M}_{\ell+1,\ell}^{\kappa}\right)^t, \qquad \mathbf{J}_{\kappa,\kappa'}^{\ell+1,\mathcal{I}} = \mathbf{M}_{\ell+1,\ell}^{\kappa} \left(\mathbf{J}_{\kappa,\kappa'}^{\ell,\mathcal{I}}\right) \left(\mathbf{M}_{\ell+1,\ell}^{\kappa'}\right)^t. \tag{22}$$

The projection matrices read

$$\mathbf{M}_{\ell+1,\ell}^{\kappa} = \left(\mathbf{M}_{\ell+1}^{\kappa}\right)^{-1} \int_\kappa \varphi^{\ell+1} \otimes \varphi^{\ell}, \quad \text{where} \quad \mathbf{M}_{\ell+1}^{\kappa} = \int_\kappa \varphi^{\ell+1} \otimes \varphi^{\ell+1}, \tag{23}$$

and φ^ℓ represents the set of basis functions spanning the space $\mathbb{P}_d^{\kappa_\ell}(\kappa)$. When using hierarchical orthonormal basis functions, $\mathbf{M}_{\ell+1}^{\kappa}$ is the unit diagonal elemental mass matrix and $\mathbf{M}_{\ell+1,\ell}^{\kappa} \in \mathbb{R}^{\dim\left(\mathbb{P}_d^{\kappa_{\ell+1}}(\kappa)\right) \times \dim\left(\mathbb{P}_d^{\kappa_\ell}(\kappa)\right)}$ is a unit diagonal rectangular matrix. Accordingly the Galerkin projection in (22) falls back to a trivial and inexpensive sub-block extraction.

Being $\mathbb{P}_d^{k_0}(\mathcal{T}_h) \supset \mathbb{P}_d^{\kappa_\ell}(\mathcal{T}_h)$, it can be demonstrated that inherited and non-inherited p-multigrid algorithms lead to the same inviscid Jacobian operators, that is $\boldsymbol{J}_\ell^{!\nu,\mathcal{I}} = \boldsymbol{J}_\ell^{!\nu}$. As opposite inherited and non-inherited viscous Jacobian differ because of the terms involving lifting operators. Note that inherited lifting operators act on traces of polynomial functions mapped into $\mathbb{P}_d^{k_0}(\mathcal{T}_h)$, accordingly

$$\text{inherited } p\text{-multigrid lifting operators}, \quad \mathbf{r}^{\sigma}([\![z_h]\!]) : \mathbb{P}_d^{k_0}(\sigma) \to [\mathbb{P}_d^{k_0}(\mathcal{T}_h)]^d, \tag{24}$$

$$\text{non-inherited } p\text{-multigrid lifting operators}, \quad \mathbf{r}^{\sigma}([\![z_h]\!]) : \mathbb{P}_d^{\kappa_\ell}(\sigma) \to [\mathbb{P}_d^{\kappa_\ell}(\mathcal{T}_h)]^d. \tag{25}$$

Interestingly, using the definitions of the global and local lifting operators, the bilinear form

can be rewritten as follows

$$
\begin{aligned}
j_h^{\nu}(\delta \boldsymbol{w}_h, \boldsymbol{z}_h) = &- \sum_{\kappa \in \mathcal{T}_h} \int_{\kappa} \sum_{i,j=1}^{d+1} \sum_{p,q=1}^{d} \frac{\partial \widetilde{F}_{p,i}}{\partial\Big(\partial w_j/\partial x_q - R_q^{\kappa}(w_j)\Big)} \frac{\partial(\delta w_j)}{\partial x_q} \frac{\partial z_i}{\partial x_p} + \\
&+ \sum_{\sigma \in \mathcal{F}_h} \int_{\sigma} \sum_{i,j=1}^{d+1} \sum_{p,q=1}^{d} n_q^{\sigma} \frac{\partial \widetilde{F}_{p,i}}{\partial\Big(\partial w_j/\partial x_q - R_q^{\kappa}(w_j)\Big)} [\![\delta w_j]\!] \left\{\!\!\left\{ \frac{\partial z_i}{\partial x_p} \right\}\!\!\right\} + \\
&+ \sum_{\sigma \in \mathcal{F}_h} \int_{\sigma} \sum_{i,j=1}^{d+1} \sum_{p,q=1}^{d} n_p^{\sigma} \frac{\partial \widehat{F}_{p,i}}{\partial\Big(\partial w_j/\partial x_q - \eta_{\sigma} r_q^{\sigma}(w_j)\Big)} \left\{\!\!\left\{ \frac{\partial(\delta w_j)}{\partial x_q} \right\}\!\!\right\} [\![z_i]\!] + \\
&- \sum_{\sigma \in \mathcal{F}_h} \int_{\sigma} \sum_{i,j=1}^{d+1} \sum_{p,q=1}^{d} \eta_{\sigma} \frac{\partial \widehat{F}_{p,i}}{\partial\Big(\partial w_j/\partial x_q - \eta_{\sigma} r_q^{\sigma}(w_j)\Big)} r_q^{\sigma}\left([\![\delta w_j]\!]\right) r_p^{\sigma}\left([\![z_i]\!]\right) \qquad (26)
\end{aligned}
$$

showing that only the last term, *i.e.*, the stabilization term, cannot be reformulated lifting-free. In particular, it can be demonstrated that the inherited stabilization term introduces an excessive amount of stabilization with respect to its non-inherited counterpart, see [36] for the theoretical estimates. In the context of *h*-multigrid solution strategies, this showed to be detrimental for multigrid algorithm performance, see [23], where the authors consider dG discretizations of the INS equations, and [37], where preconditioners for weakly over-penalized symmetric interior penalty dG discretization of elliptic problems are devised. In those works, the use of rescaled-inherited coarse space operators was proposed in order to recover the correct amount of stabilization of the viscous operator and the optimal multigrid efficiency. In the numerical results, the use of rescaled inherited coarse grid operators extended to the *p*-version of the multigrid linear solver [36] is also assessed. A rather limited benefits was observed on practical three dimensional simulations involving convection-dominated regimes, which justifies the use of standard inherited approaches for production runs involving the under-resolved direct numerical simulations of turbulent flows.

2.5.1. The p*-multigrid iteration*

In this section we provide an overlook of the sequence of operations involved in *p*-multigrid iterations. The recursive *p*-multigrid $\mathcal{V}$-cycle and full *p*-multigrid $\mathcal{V}$-cycle for the problem $\boldsymbol{G}_{\ell}\, \delta \boldsymbol{w}_{\ell} = \boldsymbol{g}_{\ell}$ on level ℓ reads:

Algorithm 2 $\overline{\boldsymbol{w}}_\ell = \mathrm{MG}_{\mathcal{V}}(\ell, \boldsymbol{g}_\ell, \boldsymbol{w}_\ell)$

if $(\ell = L)$ **then**
  $\overline{\boldsymbol{w}}_\ell = \mathrm{SOLVE}(\boldsymbol{G}_\ell, \boldsymbol{g}_\ell, 0)$
end if
if $(\ell < L)$ **then**
  Pre-smoothing:
  $\overline{\boldsymbol{w}}_\ell = \mathrm{SMOOTH}(\boldsymbol{G}_\ell, \boldsymbol{g}_\ell, \boldsymbol{w}_\ell)$
  Coarse grid correction:
  $\boldsymbol{d}_\ell = \boldsymbol{g}_\ell - \boldsymbol{G}_\ell \overline{\boldsymbol{w}}_\ell$
  $\boldsymbol{d}_{\ell+1} = \mathcal{I}_\ell^{\ell+1} \boldsymbol{d}_\ell$
  $\boldsymbol{e}_{\ell+1} = \mathrm{MG}_{\mathcal{V}}(\ell+1, \boldsymbol{d}_{\ell+1}, 0)$
  $\widehat{\boldsymbol{w}}_\ell = \overline{\boldsymbol{w}}_\ell + \mathcal{I}_{\ell+1}^{\ell} \boldsymbol{e}_{\ell+1}$
  Post-smoothing:
  $\overline{\boldsymbol{w}}_\ell = \mathrm{SMOOTH}(\boldsymbol{G}_\ell, \boldsymbol{g}_\ell, \widehat{\boldsymbol{w}}_\ell)$
end if
return $\overline{\boldsymbol{w}}_\ell$

Algorithm 3 $\overline{\boldsymbol{w}}_\ell = \mathrm{MG}_{\mathrm{full}}(\ell, \boldsymbol{g}_\ell, \boldsymbol{w}_\ell)$

if $(\ell = L)$ **then**
  $\overline{\boldsymbol{w}}_\ell = \mathrm{SOLVE}(\boldsymbol{G}_\ell, \boldsymbol{g}_\ell, 0)$
end if
if $(\ell < L)$ **then**
  $\boldsymbol{g}_{\ell+1} = \mathcal{I}_\ell^{\ell+1} \boldsymbol{g}_\ell$
  $\widehat{\boldsymbol{w}}_{\ell+1} = \mathrm{MG}_{\mathrm{full}}(\ell+1, \boldsymbol{g}_{\ell+1}, 0)$
  $\mathcal{V}$*-cycle correction:*
  $\widehat{\boldsymbol{w}}_\ell = \mathcal{I}_{\ell+1}^{\ell} \widehat{\boldsymbol{w}}_{\ell+1}$
  $\boldsymbol{d}_\ell = \boldsymbol{g}_\ell - \boldsymbol{G}_\ell \widehat{\boldsymbol{w}}_\ell$
  $\boldsymbol{e}_\ell = \mathrm{MG}_{\mathcal{V}}(\ell, \boldsymbol{d}_\ell, 0)$
  $\overline{\boldsymbol{w}}_\ell = \widehat{\boldsymbol{w}}_\ell + \boldsymbol{e}_\ell$
end if
return $\overline{\boldsymbol{w}}_\ell$

To obtain an application of the p-multigrid preconditioner the multilevel iteration is invoked on the problem $\boldsymbol{G}_h\ \delta\boldsymbol{w}_h = \boldsymbol{g}_h$. While one $\mathrm{MG}_{\mathcal{V}}$ iteration requires two applications of the smoother on the finest level (one pre- and one post-smoothing) and one application of the coarse level smoother independently from the number of levels, one $\mathrm{MG}_{\mathrm{full}}$ iteration requires one application of the finest level smoother and L applications of the coarse level smoother.

In this work the p-multigrid Full-$\mathcal{V}$ cycle iteration will be applied for the numerical solution of linearized equations systems arising in Rosenbrock time marching strategies for dG discretizations of incompressible flow problems. In the context of such problems we seek for the best performance employing full p-multigrid and tuning preconditioners and smoothing options. We remark that, in this work, all the smoothers of the multigrid strategy are chosen to be GMRES, and thus two nested Krylov iterative solvers are invoked. In this setting, the outer solver follows the flexible GMRES implementation [38], since the action of the multigrid linear solver needs to be stored at each iteration. In order to reduce the overall memory footprint, a matrix-free implementation of the p-multigrid linear solver needs to be employed. To this end, we recall what was reported in Section 2.6 regarding matrix-free iterative strategies, and we remark that only if EWBJ preconditioner is coupled with a matrix-free GMRES smoother on the finest space, an overall reduction of memory footprint of the solver can be achieved. As demonstrated in the results section, such strategy can be conveniently used for practical simulations without spoiling

the convergence rates of the multigrid iteration. Another important aspect regarding the choice of the smoothers involves the coarse space. To this end, the use of more powerful preconditioners like BJ or ASM for the smoothers on the coarsest space is typically suggested to ensure a satisfactory convergence rate not polluted by the domain decomposition. While matrix-free iterative solvers can be employed at no additional cost at high order, they are more expensive for low order polynomials with respect to matrix-based methods, and thus the use of the latter would speed-up the solution process. In this configuration, the off-diagonal blocks needed by the coarse level operators can be computed at the coarsest polynomial degree for the sake of efficiency. In the rest of the paper several combinations of preconditioners are investigated with particular attention to the quantification of the parallel performance and of the memory savings. We point out that, since the number of non-zeros of the primal Jacobian matrix scales as k^{2d}, the memory footprint of a coarse space matrix can be less than 1% that of the EWBJ used on the fine space, and thus the overall memory saving is not compromised.

2.6. Memory footprint considerations

In this section an estimate of the memory usage of all the strategies employed in this paper is devised to fully appreciate the memory savings achievable through a matrix-free solver preconditioned with p-multigrid. We observe that the memory footprint of the global block matrix scales as

$$\mathrm{card}(\mathcal{T}_h)\ (\overline{\mathrm{card}}(\mathcal{F}_{\partial\kappa}) + 1)\ ((d+1)\ \dim(\mathbb{P}^k_d))^2,$$

where $\overline{\mathrm{card}}(\mathcal{F}_{\partial\kappa})$ is the average number of element's faces, $d+1$ is the number of variables in d space dimensions and $((d+1)\ \dim(\mathbb{P}^k_d))^2$ is the number of non-zeros in each matrix block. While for a matrix-based implementation we assume that both the global system matrix and the preconditioner are stored in memory, for a matrix-free approach only the preconditioner is stored. The preconditioner's memory occupation is carefully estimated:

1. for EWBJ, we consider only the non-zero entries of a block-diagonal matrix.
2. For BJ we consider the storage of ILU(0) factorization applied to the domain-wise portion of the iteration matrix, which neglects the off diagonal blocks related to faces residing on a partition boundary.
3. For ASM(q,ILU(0)), we assume that the preconditioner applies the ILU(0) decomposition to a larger matrix, bigger than the sub-domain matrix of the BJ precondtioner. The exact

number of additional non-zero blocks is difficult to estimate for general unstructured grids since it depends on mesh topology, the partitioning strategy, and element types in case of a hybrid grid. Nevertheless, an estimation can be done assuming a square and cubical domain discretized by uniformly distributed quadrilateral and hexahedral elements in $2d$ and $3d$, respectively. Periodic boundary conditions are also assumed at the domain boundaries. Accordingly the number of non-zero blocks in each sub-domain matrix can be estimated as follows

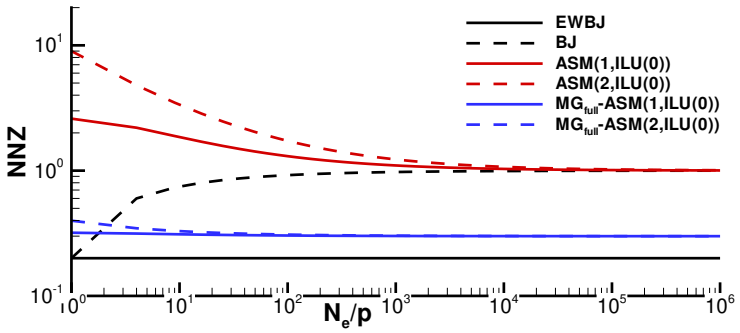
$$(2d+1)\left(\frac{N_e}{p}+2dq\left(\frac{N_e}{p}\right)^{\frac{d-1}{d}}+2^{d-1}d\sum_{i=1}^{q}(i-1)\right)-2d\left(\left(\left(\frac{N_e}{p}\right)^{\frac{1}{d}}+2q\right)^{d-1}-2^{d-1}(d-2)\sum_{i=0}^{q}(q-i)\right) \tag{27}$$

where q is the number (or depth) of overlapping layers, p is the number of processes, $N_e = \text{card}(\mathcal{T}_h)$ is the number of mesh elements and $\overline{\text{card}}(\mathcal{F}_{\partial\kappa}) = 2d$. In Eq. (27), the first term takes into account that each element of a partition, which is widened with the overlapping elements, contributes to the Jacobian matrix with $(1 + 2d)$ blocks, with $2d$ the number of faces of an element, while the second term subtracts the blocks not considered by the preconditioner, *i.e.* those due to the connection between elements at the boundary faces of the augmented partition. For $q = \{0, 1, 2\}$ we get an estimation of the number of non-zero blocks corresponding to BJ, ASM(1,ILU(0)) and ASM(2,ILU(0)) preconditioners, respectively. The number of non-zero entries of the global matrix is obtained multiplying the number of non-zero blocks by the size of each block, *i.e.*, the number of DoFs per element squared.

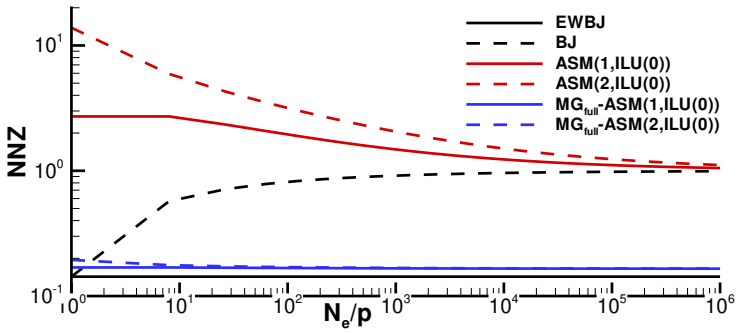
Figures 1(a) and 1(b) report the ratio between the estimated number of non-zeros of the preconditioner and the system matrix with respect to the number of sub-domains. For $(N_e/p) = 1$, corresponding to one element per partition, BJ reduces to EWBJ, which provides a $1/(2d+1)$ decrease of the number of non-zeros. On the other hand, for both ASM(1,ILU(0)) and ASM(2,ILU(0)) the number of non-zero entries grows as (N_e/p) approaches one, which is explained by the growth of the number of overlapping regions.

In the same manner the memory footprint of p-multigrid preconditioners can be estimated. We consider as reference the three-level p-multigrid strategy whose specs reads: $k = 6$, FGMRES(MF) outer solver, GMRES(MB)[ASM(1,ILU(0)] smoothing on the coarsest level ($k = 1$), GMRES(MF)[EWBJ] on the finest level ($k = 6$) and GMRES(MB)[EWBJ] on the intermediate

level ($k = 2$). We remark that the memory allocation of coarse levels preconditioners has as a small impact on the total number of non-zeros, due the reduction of block size. For instance, in three space dimensions, $\mathbf{G}_2$ and $\mathbf{G}_1$ have 440 and 70 time less non-zeros that the $\mathbf{G}_0$ matrix, respectively. By the analysis of Figure 1, it is clear that the most significant memory savings can be obtained via the use of single-grid element-wise block Jacobi preconditioning coupled with a matrix-free implementation of the iterative solver. However, as it will be shown in the remaining of the paper, the use of such strategy is highly inefficient from the computation time viewpoint leading to unreasonably large performance penalties with respect to BJ and ASM preconditioners. Code profiling on benchmark test cases demonstrated that EWBJ preconditioned GMRES can be efficiently employed as a fine-level smoother of a multilevel strategy, providing execution time gains and, at the same time, memory savings which reach the 92% as demonstrated in Figure 1.



(a) $d = 2$



(b) $d = 3$

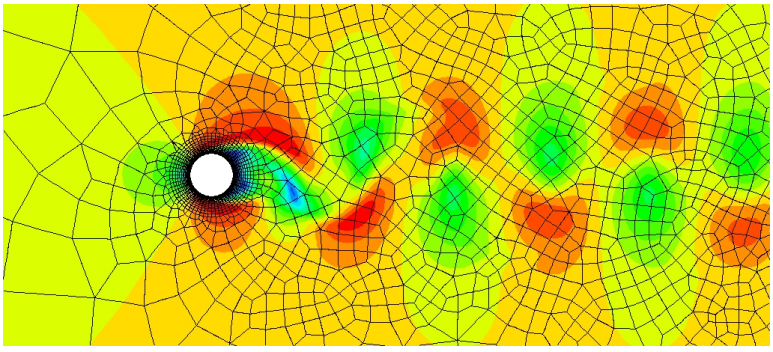
Figure 1: Estimated relative number of non-zeros (NNZ) of the preconditioner with respect to the non-zeros of the Jacobian as a function of the number of elements per partition for a two dimensional ($d = 2$) and three-dimensional case ($d = 3$). See text for details.

3. Numerical results

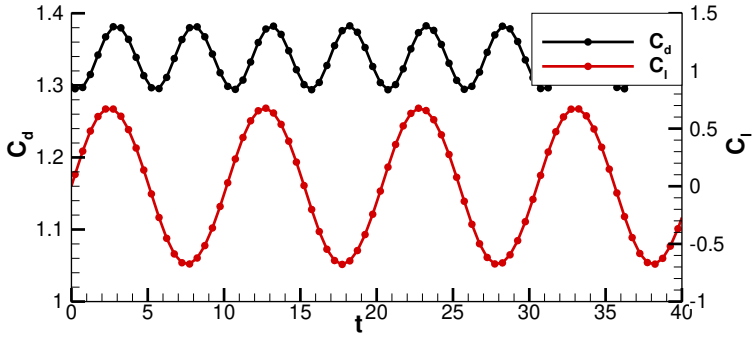
In this section the performance of the p-multigrid matrix-free preconditioner is compared to state-of-the-art single-grid strategies in the context of unsteady flow simulations. In all the numerical experiments the matrix-based, block-Jacobi preconditioned GMRES solver is assumed as the reference solution strategy. Two incompressible flow problems of increasing complexity are considered: i) the two-dimensional laminar flow around a circular cylinder at $Re = 200$; ii) the three-dimensional laminar flow around a sphere at $Re = 300$.

3.1. Laminar flow past a two-dimensional circular cylinder at Re = 200

The laminar flow around a circular cylinder at $Re = 200$ is solved with $k = 6$ on a computational grid made of 4710 elements with curved edges represented by cubic Lagrange polynomials. The domain extension in non-dimensional units is $[-50, 100]\times[-50, 50]$, with reference length equal to the cylinder diameter. We remark that the grid was deliberately generated with



(a) Velocity magnitude iso-contours



(b) C_d and C_l coefficients history

Figure 2: Laminar flow around a circular cylinder at $Re = 200$.

a severe refinement in the wake region, as well as large elements at the far-field, in order to challenge the solution strategy on a stiff space discretization. Dirichlet and Neumann boundary conditions are imposed at the inflow and outflow boundaries, respectively, while symmetry flow conditions are employed at the top and bottom boundaries. The no-slip Dirichlet boundary condition is imposed on the cylinder wall. A snapshot of the mesh and the velocity magnitude contours is shown in Fig. 2(a). Time marching is performed using a non-dimensional fixed time step $\Delta t = 0.25$, corresponding to 1/20 of the shedding period, which was found adequate to describe the flow physics and large enough to stress the importance of an efficient solution strategy. Fig. 2(b) reports lift and drag coefficients history. The drag coefficient $C_d = 1.335$ and the Strouhal number St = 0.1959 are in good agreement with [39] and references therein.

Performance assessment. The p-multigrid preconditioner approach seeks to minimize the number of GMRES iterations on the fine grid by means of a full p-multigrid solution strategy. A full $\mathcal{V}$-cycle p-multigrid iteration (see Algorithm 3) has many parameters to tune in order to get the best performance, among the others we mention the following: i) the choice of the smoother and its preconditioner on each level, ii) the number of smoothing iteration on each level, iii) the forcing term (controlling the exit condition based on the relative defect drop) of the coarse solver and its maximum number of iterations. Accordingly the combination of these parameters lead to a multitude of different configurations that is hard to explore comprehensively. Nevertheless, we provide some useful indications that can be directly applied to the simulation of realistic flow problems. In general, with respect to the linear test cases, such as that presented in [36], the use of a full p-multigrid strategy together with an increased number of smoothing iterations proved to deal more efficiently with the solution of the incompressible Navier–Stokes equations. The experiments are devoted to show the benefits of i) the use of a memory saving smoother on the fine space to reduce the memory footprint and increase the computational efficiency; ii) the use of a rescaled-inherited approach for the coarse space operators to improve the convergence rates of the iterative solver. Code profiling is applied for 10 time steps starting from a fully developed flow solution obtained with the same polynomial degree, same time step size and solving linear system up to the same tolerance. In practice, the performance of the preconditioners is averaged on the solution of 40 linear systems. The efficiency of each setting is monitored in parallel, to assess the behavior of different preconditioners in a realistic setup for this kind of computations. The runs are performed on a computational node made by two sixteen-core AMD Opteron CPUs.

First, we report in Table 1 the results obtained using single-grid preconditioners. As expected, the numerical experiments show a sub-optimal parallel efficiency for the BJ preconditioner, indeed the average number of linear iterations increases while increasing the number of sub-domains. The iterations number increase tops at 62% when comparing the simulation on 16 cores against the serial one. The ASM(1,ILU(0)) preconditioner partially heals the performance degradation providing a 10% increase of the iterations number at the expenses of a higher memory requirement, as explained in Section 2.6. The matrix-based and the matrix-free versions of GMRES provide a similar number of iterations with a CPU time that is in favour of the former. This can be explained by the high quadrature cost associated to non-affine mesh elements, see *e.g.*[40]. In fact, while Franciolini *et al.*[9] demonstrated that the residual computation and a matrix-vector product has similar costs when dealing with high-order discretizations on affine elements, in this case the numerical quadrature expense has a higher relative cost on residual evaluation. We remark that no attempt was made to optimize numerical quadrature, in particular elements located far from curved boundaries are still treated as high-order non-affine elements for the sake of simplicity. Even if matrix-free iterations can be further improved in term of efficiency in realistic applications, this is beyond the scope of the present work. Table 2 allows to evaluate the impact of the fine smoother preconditioner on the computational efficiency. We report two parameters of interest, the average number of FGMRES iterations (ITs) and the speed-up with respect to the best performing single-grid preconditioner, $\mathrm{SU_{MB}}$ = TotTime/$\mathrm{TotTime_{ref}}$, where $\mathrm{TotTime_{ref}}$ is the total CPU time of the GMRES(MB)[BJ] method. The specs of the *p*-multigrid iteration setup are reported in the top of the table. We also compare the standard-inherited approach (*scaling off*) with the rescaled-inherited one (*scaling on*).

FGMRES[$\mathrm{MG_{full}}$] with 3 GMRES(MB)[BJ] smoothing iterations provides a speed-up of about 2 in serial computations with respect to the reference strategy. Although the solver is still faster

Solver	GMRES(MB)		GMRES(MB)		GMRES(MF)		GMRES(MF)		GMRES(MF)	
Prec	BJ		ASM(1,ILU(0))		EWBJ		BJ		ASM(1,ILU(0))	
nProcs	ITs	TotTime	ITs	TotTime	ITs	TotTime	ITs	TotTime	ITs	TotTime
1	123.5	3805	123.5	3805	542.8	36700	112.2	9860	112.2	9860
2	108.0	1756	121.3	1917	538.7	17980	102.4	4547	109.9	4782
4	105.3	859	123.9	982	543.9	9281	103.5	2325	111.2	2426
8	138.0	543	120.4	515	542.2	4615	121.4	1333	111.2	1253
16	199.7	497	134.7	378	554.4	2934	171.3	995	122.8	750

Table 1: Two-dimensional cylinder test case. Single-grid parallel performances, matrix-based and matrix-free implementations. Comparison of the average number of GMRES iterations (ITs) and the whole elapsed CPU time (solution plus assembly) TotTime.

than the reference one, the parallel performance is not satisfactory being an increase in the number of iterations observed. As expected a better scalability can be obtained with 3 GMRES(MB)[ASM(1,ILU(0))] smoothing iterations. The numerical experiment revealed that to increase the number of iterations from 3 to 8 is mandatory to maintain the smoothing efficiency of GMRES(MB)[EWBJ], and to achieve a satisfactory performance in parallel. Indeed, despite being less performing in serial runs, the number of iterations is almost independent from the number of processes. Moreover, increasing the number of iterations of GMRES(MB)[BJ] does not pay off in terms of speedup. The number of FGMRES iterations is significantly reduced in all the cases when considering rescaled-inherited coarse grid operators. However, it can be seen that the strategy does not always pay off in terms of speedup, as an increased cost for the matrix assembly is required to compute the rescaled stabilization terms. The GMRES(MB)[EWBJ] smoother, despite being less powerful per-iteration, is competitive with more expensive preconditioners in parallel computations. Interestingly, the EWBJ preconditioner is also the cheapest from the memory footprint viewpoint.

Table 3 compares the computational efficiency when varying the preconditioner on the coars-

Solver	ℓ	κ_ℓ	ITs	Smoother		Prec	
FGMRES[MG_{full}]	0,1	6, 2	$*$	GMRES(MB)		‡	
	2	1	40	GMRES(MB)		ASM(1,ILU(0))	
scaling off	^{*3} ‡BJ		^{*3} ‡ASM(1,ILU(0))		^{*8} ‡BJ		^{*8} ‡EWBJ	
nProcs	ITs	SU_{MB}	ITs	SU_{MB}	ITs	SU_{MB}	ITs	SU_{MB}
1	4.10	2.02	4.10	1.98	2.98	1.76	5.48	1.56
2	4.63	1.67	4.05	1.74	3.10	1.51	5.48	1.36
4	5.73	1.51	4.05	1.74	3.63	1.40	5.63	1.36
8	7.20	1.47	4.35	1.83	3.85	1.48	5.63	1.48
16	8.63	1.80	5.38	2.25	5.05	1.68	5.70	2.01
scaling on	^{*3} ‡BJ		^{*3} ‡ASM(1,ILU(0))		^{*8} ‡BJ		^{*8} ‡EWBJ	
nProcs	ITs	SU_{MB}	ITs	SU_{MB}	ITs	SU_{MB}	ITs	SU_{MB}
1	3.43	2.11	3.43	2.11	2.48	1.88	3.50	1.92
2	3.68	1.82	3.55	1.80	2.80	1.58	3.53	1.69
4	4.78	1.64	3.60	1.79	2.55	1.65	3.83	1.64
8	5.13	1.59	3.60	1.83	2.58	1.66	3.68	1.69
16	7.65	1.58	4.10	2.20	2.85	1.96	3.50	2.21

Table 2: Two-dimensional cylinder test case. Effects of the smoother type and the rescaled-inherited coarse spaces on parallel performance. Comparison of the average number of FGMRES iterations (ITs) and the speed-up (SU_{MB}) of the p-multigrid preconditioner with respect to the GMRES(MB)[BJ]. The asterisk and the double dagger symbols in the solver specs row are placeholders for the number of iterations (ITs) and coarse solver type of each column, respectively.

est level. We fix the GMRES(MB)[EWBJ] smoother on all the other levels, with the idea of exploiting its performance on parallel runs. The top and bottom of the table include results for

Solver	ℓ	κ_ℓ	ITs	Smoother	Prec
FGMRES[MG_{full}]	0	6	8	$*$	EWBJ
	1	2	8	GMRES(MB)	EWBJ
	2	1	40	GMRES(MB)	‡

scaling off	*GMRES(MB) ‡BJ		*GMRES(MB) ‡ASM(1,ILU(0))		*GMRES(MB) ‡ASM(1,ILU(1))	
nProcs	ITs	SU_{MB}	ITs	SU_{MB}	ITs	SU_{MB}
1	5.48	1.56	5.48	1.56	4.73	1.64
2	5.63	1.35	5.48	1.36	4.73	1.43
4	5.43	1.39	5.63	1.36	4.73	1.44
8	5.90	1.46	5.63	1.48	4.75	1.55
16	6.85	1.8	5.70	2.01	4.95	2.13
scaling on	*GMRES(MB) ‡BJ		*GMRES(MB) ‡ASM(1,ILU(0))		*GMRES(MB) ‡ASM(1,ILU(1))	
nProcs	ITs	SU_{MB}	ITs	SU_{MB}	ITs	SU_{MB}
1	3.50	1.92	3.50	1.92	3.15	1.96
2	3.55	1.69	3.53	1.69	3.15	1.72
4	3.35	1.75	3.83	1.64	3.15	1.73
8	3.33	1.79	3.68	1.69	3.15	1.76
16	3.48	2.24	3.50	2.21	3.18	2.27
scaling off	*GMRES(MF) ‡BJ		*GMRES(MF) ‡ASM(1,ILU(0))		*GMRES(MF) ‡ASM(1,ILU(1))	
nProcs	SU_{MB}	SU_{MF}	SU_{MB}	SU_{MF}	SU_{MB}	SU_{MF}
1	0.61	1.59	0.61	1.59	0.69	1.80
2	0.55	1.41	0.56	1.45	0.63	1.63
4	0.55	1.42	0.54	1.41	0.63	1.63
8	0.58	1.43	0.60	1.48	0.70	1.70
16	0.80	1.61	0.96	1.92	1.01	2.03
scaling on	*GMRES(MF) ‡ BJ		*GMRES(MF) ‡ASM(1,ILU(0))		*GMRES(MF) ‡ASM(1,ILU(1))	
nProcs	SU_{MB}	SU_{MF}	SU_{MB}	SU_{MF}	SU_{MB}	SU_{MF}
1	0.89	2.31	0.89	2.30	0.98	2.54
2	0.84	2.18	0.82	2.11	0.9	2.32
4	0.85	2.21	0.80	2.06	0.89	2.30
8	0.91	2.23	0.90	2.20	0.97	2.39
16	1.39	2.80	1.41	2.81	1.53	3.05

Table 3: Two-dimensional cylinder test case. Effects of the coarse level solver on parallel performance. Comparison of the average number of FGMRES iterations (ITs) and the speed-up of the *p*-multigrid preconditioner with respect to the best performing single-grid preconditioner in its matrix-based and matrix-free implementation (SU_{MB} and SU_{MF}, respectively). The asterisk and the double dagger symbols in the solver specs row are placeholders for the smoother and coarse solver types of each column, respectively.

a matrix-based and a matrix-free approach, respectively. Note that only on the finest level the matrix-vector products are performed matrix-free, both within the outer FGMRES iteration and the fine GMRES smoother. Indeed, since the coarse levels operators are fairly inexpensive to store in memory, the moderate memory savings of a matrix-free implementation would not justify the increased computational costs at low polynomial orders. The results highlight that a further improvement in computational efficiency is achieved by means of a [ASM(1,ILU(1))] preconditioner for the coarsest smoother: the number of FGMRES iterations decreases while maintaining optimal scalability. In addition, the speedup values for the matrix-free approach increase at large number of cores. We remark that, due to low polynomial degree of the coarsest level, the additional level of fill of the ILU factorization is not significant from the memory footprint viewpoint. Interestingly, the increased robustness of the rescaled-inherited p-multigrid approach results in similar speedups for all the coarse level solver options, but also shows that with a powerful smoother on the coarsest level the standard inherited approach shows competitive speedup values as well.

Table 3 reports numerical experiments using the matrix-free solver on the fine space. Since for this test case the cost-per-iteration of the matrix-free solver is higher than that of a matrix-based, reducing the number of FGMRES iterations does pay off. Accordingly, the benefits of rescaled-inherited coarse grid operators are more evident: the total execution time is comparable with GMRES(MB)[ASM(1,(ILU(0))] and almost three times faster than GMRES(MF)[BJ]. However, as already mentioned, further optimizations on the quadrature rules would reduce considerably this penalty.

Table 4 compares three- and four-levels p-multigrid preconditioners. The additional level significantly reduces the number of FMGRES iterations at the expense of storing a fourth degree coarse grid operator. The use of a rescaled-inherited approach reduce the number of iterations further, but the CPU time compares similarly for a matrix-based solution strategy. On the other hand, the benefits in terms of speedup are most significant in the matrix-free framework, where solution times dominates assembly times.

To summarize the results, we report in Figure 3 the performance of the algorithm compared to those of the single grid GMRES(MB)[BJ]. The Figure 3(a) shows the maximum theoretical gain of the p-multigrid algorithm, expressed as fine space iteration ratio speed-up, *i.e.* ITs_{ref}/ITs, as a function of the number of computational cores. We point out that the iteration ratio measures

the algorithmic reduction of the operations spent on the most expensive part of the code, *i.e.*, the number of GMRES iterations performed with the highest polynomial order of the simulation. As the number of computational core increases, the iteration ratio grows in all cases, revealing the superior scalability of the preconditioning strategy. In particular, it can be seen that increasing the number of levels of the multigrid operator, as well as using the rescaled-inherited approach, provide a reduction of the overall iterations of the algorithm. We remark that no distinction between matrix-based and matrix-free is required, as the number of iterations required by GMRES are identical. Figure 3(b) shows the performance from the CPU time point of view. Trends similar to that of the iteration ratio can be observed, as the speed-up values increase by increasing the number of computational cores. In particular, they reach 2 for the matrix-based algorithm with scaling on and $L = 3$, while the matrix-free algorithm peaks at 3.5 for the same settings.

In conclusion, it has been demonstrated how the use of cheap preconditioners like EWBJ on the fine space smoothers of multigrid cycle, coupled with a matrix-free implementation of the iterative solver, can be used to devise an efficient and memory saving solution strategy. The op-

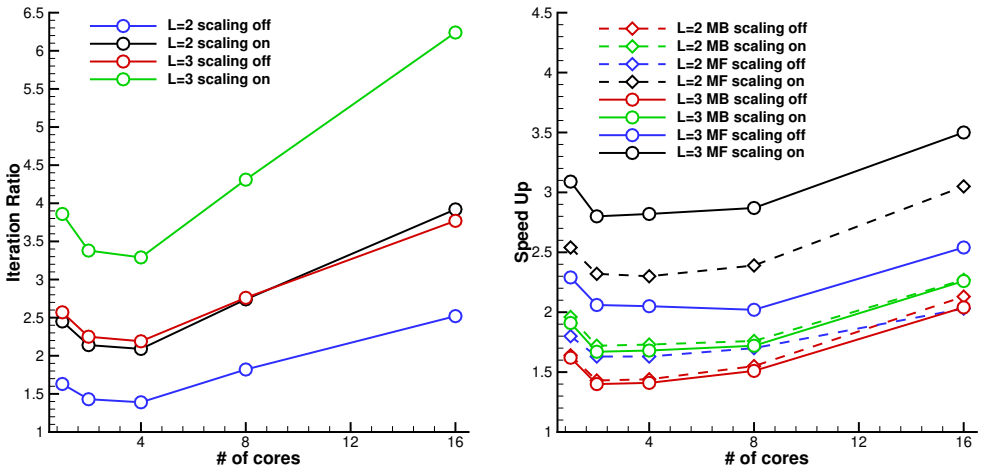
Solver	ℓ	κ_ℓ	ITs	Smoother			Prec	
	0	6	8	$*$			EWBJ	
FGMRES[MG_{full}]	1,...,L-1	‡	8	GMRES(MB)			EWBJ	
	L	1	40	GMRES(MB)			ASM(1,ILU(1))	
	GMRES(MB)*		GMRES(MB)*		GMRES(MF)*		GMRES(MF)*	
scaling off	$2^‡$ (L=2)		$4,2^‡$ (L=3)		$2^‡$ (L=2)		$4,2^‡$ (L=3)	
nProcs	ITs	SU_{MB}	ITs	SU_{MB}	SU_{MB}	SU_{MF}	SU_{MB}	SU_{MF}
1	4.73	1.64	3.00	1.62	0.69	1.80	0.88	2.29
2	4.73	1.43	3.00	1.40	0.63	1.63	0.80	2.06
4	4.73	1.44	3.00	1.41	0.63	1.63	0.80	2.05
8	4.75	1.55	3.10	1.51	0.70	1.70	0.82	2.02
16	4.95	2.13	3.33	2.04	1.01	2.03	1.28	2.54
	GMRES(MB)*		GMRES(MB)*		GMRES(MF)*		GMRES(MF)*	
scaling on	$2^‡$ (L=2)		$4,2^‡$ (L=3)		$2^‡$ (L=2)		$4,2^‡$ (L=3)	
nProcs	ITs	SU_{MB}	ITs	SU_{MB}	SU_{MB}	SU_{MF}	SU_{MB}	SU_{MF}
1	3.15	1.96	2.00	1.91	0.98	2.54	1.19	3.09
2	3.15	1.72	2.00	1.67	0.90	2.32	1.08	2.80
4	3.15	1.73	2.00	1.68	0.89	2.30	1.09	2.82
8	3.15	1.76	2.00	1.72	0.97	2.39	1.17	2.87
16	3.18	2.27	2.00	2.26	1.53	3.05	1.75	3.50

Table 4: Two-dimensional cylinder test case. Comparison of a three-level and four-level p-multigrid strategy in optimal settings for matrix-based and matrix-free fine level options in terms of SU_{MB} and SU_{MF}, respectively). The asterisk and the double dagger symbols in the solver specs row are placeholders for the smoother type and (number, order) of the coarse levels, respectively.

timal scalability properties of the EWBJ, which involve local-to-each element operations, can be conveniently coupled with a more powerful preconditioning strategy on the coarse space smoothers, for example an Additive Schwarz method, which helps to maintain an optimal solution of the coarse space problem even in the context of highly parallel runs. In the next few sections, test cases of increasing complexity will be presented to further assess the performance of the devised strategy. We remark that similar specs to those reported herein will be employed for the multigrid iteration, which proved to be optimal in the context of the solution of incompressible Navier–Stokes equations.

3.2. Three-dimensional laminar flow past a sphere at Re = 300

As a three-dimensional validation test case we computed the unsteady laminar flow past a sphere at $Re = 300$. The solution is characterised by a perfectly periodic behaviour, with the flow maintaining a plane of symmetry [41, 42, 43, 29]. In our computations, the symmetry plane was enforced by defining an appropriate boundary condition. The mesh is made of 3560 elements with a bi-quadratic geometrical representation of the wall boundary, see Fig. 4(a). The computational domain is obtained via extrusion of the wall surface discretization. While the no-slip condition is set at the wall, velocity inflow and pressure outflow boundary conditions are

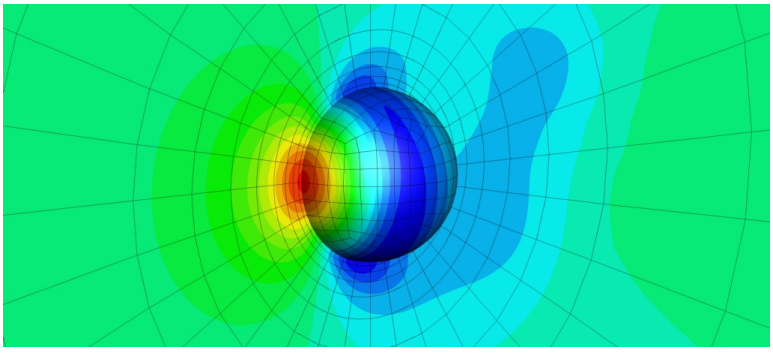


(a) Iteration ratio (b) Speed-up

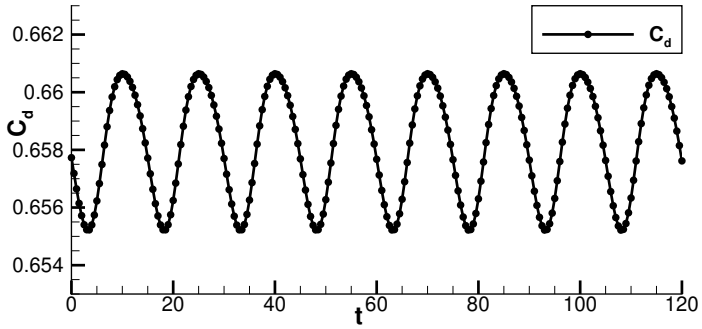
Figure 3: Two-dimensional cylinder test case. Comparison of a three-level and four-level p-multigrid strategies in optimal settings. Iteration ratio is computed setting numerator (denominator) terms equal to the number of GMRES iterations with the single grid (p-multilevel) preconditioner of reference, that is ITs_{ref}/ITs. The speed-up is defined as above.

imposed on the spherical farfield located at 50 diameters. A $k = 6$ representation of the solution was employed for all computations presented hereafter. We remark that the small number of mesh elements together with the lack of a refined region in the wake of the sphere reduce the stiffness of the problem. The parallel performance is evaluated running on the Marconi-A1 HPC platform hosted by CINECA, the Italian supercomputing center. Scalability is assessed on a single-node base, as the CPU time of the serial computation exceeded the maximum wall-clock time allowed by CINECA. The number of mesh elements is optimised to ensure that all the solution strategies fit the memory of a single node (118 GB). Despite the small size of the problem the following numerical experiments aim at providing reliable indications on the parallel performance that can be extended to real-size production runs.

The solution is integrated in time with a fixed non-dimensional time step $\delta t = 0.5$. The drag coefficient time history is shown in Fig. 4(b), its mean value reads 0.659, and the Strouhal num-



(a) C_p iso-contours



(b) C_D

Figure 4: Laminar flow around a Sphere at $Re = 300$. Pressure coefficient iso-contours (top) and drag coefficient history (bottom).

ber is St = 0.133, in agreement with the published literature, see [29]. Despite the geometry is represented with second degree polynomial spaces, the degree of exactness of quadrature rules does not consider the degree of mappings from reference to physical mesh elements. Accordingly, bilinear forms are exactly integrated only over affine mesh elements, located in general outside the boundary layer region of the mesh. We numerically verified that, for this test case, this practice does not compromise accuracy while significantly improving the matrix-free computational time, see [9]. To maximise parallel efficiency, the mesh has been partitioned using the *local* two-level partitioning strategy described in [44]. The first-level decomposition is performed according to the number of compute nodes, and the second-level decomposition acts over each node-local partition according to the number of CPU-cores per node, such that the extra-node MPI communications are minimized.

Performance assessment. Table 5 reports the parallel performance of the single-grid matrix-based and matrix-free solvers running in parallel up to 576 cores, *i.e.* the domain is decomposed using 6 elements per partition on average. Increasing the number of sub-domains from 36 to 576 leads to an increased number of GMRES iterations: 42% and 20% up when employing a BJ and an ASM(1,ILU(0)) preconditioner, respectively. Thanks to the use of quadrature rules suited for affine meshes, the CPU time of the matrix-free solver is similar to the matrix-based one.

Results reported in Table 6 for a three- and four-levels p-multigrid strategy and the exact same setup of two-dimensional computations confirm the efficacy of the multigrid preconditioner: the number of iterations stays the same up to 576 cores and the speedup are maintained in this largely-parallelized scenario. The use of a matrix-free implementation reduce the CPU time over its matrix-based counterpart, since the matrix assembly time is reduced as discussed in

Solver	GMRES(MB)		GMRES(MB)		GMRES(MF)		GMRES(MF)	
Prec	BJ		ASM(1,ILU(0))		BJ		ASM(1,ILU(0))	
nProcs	ITs	TotTime	ITs	TotTime	ITs	TotTime	ITs	TotTime
36	78.5	1448.1	34.5	1245.9	77.1	1365.9	34.7	1220.7
72	86.7	774.0	35.0	675.2	87.0	743.9	35.0	671.9
144	84.9	380.1	38.2	386.7	85.2	370.3	38.2	381.9
288	102.7	226.7	40.2	233.1	102.4	221.3	40.3	228.5
576	111.8	126.0	41.3	150.6	113.6	129.2	41.3	133.8

Table 5: Three dimensional incompressible flow around a sphere. Single-grid parallel performances, matrix-based and matrix-free implementations. Comparison of the average number of GMRES iterations (ITs) and the whole elapsed CPU time (solution plus assembly) TotTime. Computations performed on Marconi-A1@CINECA.

Section 2.3. Moreover, the stabilization scaling provides only slight improvements in terms of FGMRES iterations, while the speedup values looks almost similar to those obtained using standard-inheritance, especially for the most parallelized cases. We finally remark that the four-level p-multigrid preconditioner, with the same settings on the fine/coarse space smoothers, is almost two times faster than the best single-grid setup.

4. Application to the under-resolved simulation of incompressible turbulent flows

In this section the devised p-multigrid matrix-free implementation is applied to the implicit LES of two test cases. The first is the transitional flow around a flat plate with a semi-circular leading edge of diameter d at $Re_d = 3450$ and a low level of free-stream turbulence intensity (Tu = 0.2%). The second is the turbulent flow around the Boeing rudimentary landing gear test case at $Re = 10^6$. Those two test cases are representative of the target applications for the solution strategy here proposed.

Solver	ℓ	κ_ℓ	ITs	Smoother		Prec	
	0	6	8	$*$		EWBJ	
FGMRES[MG_{full}]	1,...,L-1	‡	8	GMRES(MB)		EWBJ	
	L	1	40	GMRES(MB)		ASM(1,ILU(0))	
	*GMRES(MB)		*GMRES(MB)		*GMRES(MF)		*GMRES(MF)	
scaling off	‡2 (L=2)		‡4,2 (L=3)		‡2 (L=2)		‡4,2 (L=3)	
nProcs	ITs	SU_{MB}	ITs	SU_{MB}	SU_{MB}	SU_{MF}	SU_{MB}	SU_{MF}
36	4.00	1.29	2.00	1.61	1.37	1.29	2.28	2.15
72	4.00	1.37	2.00	1.68	1.52	1.46	2.38	2.29
144	4.00	1.29	2.00	1.43	1.45	1.41	2.07	2.02
288	4.00	1.37	2.00	1.67	1.56	1.52	2.14	2.09
576	4.00	1.28	2.00	1.55	1.46	1.50	1.55	1.59
	GMRES(MB)*		*GMRES(MB)		*GMRES(MF)		*GMRES(MF)	
scaling on	‡2 (L=2)		‡4,2 (L=3)		‡2 (L=2)		‡4,2 (L=3)	
nProcs	ITs	SU_{MB}	ITs	SU_{MB}	SU_{MB}	SU_{MF}	SU_{MB}	SU_{MF}
36	3.0	1.43	2.00	1.51	1.53	1.44	2.09	1.97
72	3.0	1.52	2.00	1.61	1.62	1.56	2.18	2.09
144	3.0	1.36	2.00	1.50	1.55	1.51	1.93	1.88
288	3.0	1.56	2.00	1.60	1.71	1.67	2.02	1.97
576	3.0	1.45	2.00	1.43	1.63	1.67	1.73	1.78

Table 6: Three dimensional incompressible flow around a Sphere. Efficiency of a three and four level p-multigrid strategy varying the fine level smoother. Comparison of the average number of FGMRES iterations (ITs) and the speed-up of the p-multigrid preconditioner with respect to the best performing single-grid preconditioner in its matrix-based and matrix-free implementation ($SU_{MB}$and SU_{MF},respectively). The asterisk and the double dagger symbols in the solver specs row are placeholders for the smoother and coarse solver types of each column, respectively. Computations performed on Marconi-A1@CINECA.

4.1. ERCOFTAC T3L1 test case

This test case, named T3L1, is part of the ERCOFTAC test case suite. The solution exhibits at leading edge a laminar separation bubble and, downstream the transition, an attached turbulent boundary layer. Those complex flow features are perfectly suited to evaluate the efficiency of the solver and highlight the advantages of using a dG-based ILES approach. ILES naturally resolves all the flow scales (in a DNS-fashion) if the numerical resolution is enough to do so, while the numerical dissipation plays the role of a sub-grid scale model for the spatially under-resolved regions of the domain. In this test case, the laminar region is fully resolved, while in the turbulent region the dissipation of the numerical scheme dumps the under-resolved scales. We remark that in all the computations the same settings of Table 6 are employed for the p-multigrid iteration.

The simulations were performed in parallel using 540 cores on a hybrid mesh of 38320 elements with curved edges. The unstructured grid is strongly coarsened moving away from the plate, while a structured-like boundary layer is used at the wall. The first cell height is $10^{-2}d$, with d the leading edge diameter, and the mesh is refined near the reattachment region, where the minimum dimension along x axis is $2 \cdot 10^{-2}d$, see Fig. 5. The domain extension on the $x - y$ plane, with origin located in the leading edge semi-circumference center, is taken from [45], *i.e.*, $28d \times 17d$. The two-dimensional domain is extruded using 10 elements along the span-wise direction z for a length of $2d$, as in [46]. To our best knowledge, only those two works report a LES simulation of this ERCOFTAC test case. In both the cases, the numerical method was based on a standard second-order scheme and a dynamic subgrid scale model. On the other hand, we here employ sixth-order polynomials, *i.e.*, seventh-order accurate space discretizations. A direct

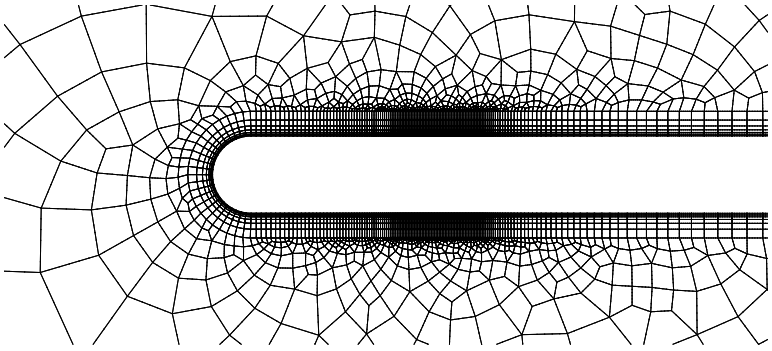


Figure 5: T3L1 test case. Near-wall detail of the computational grid.

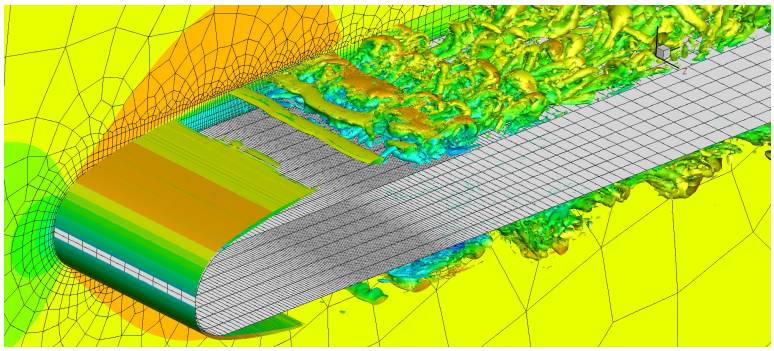
comparison of the present computations with previously published works in terms of DoFs is not trivial due to the differences of the computational domains. In [46] the DoFs count per variable is on the order of $1.88 \cdot 10^6$ and the domain extension in the $x - y$ plane is 1.9 times smaller. In [45] the DoFs count is on the order of $4.39 \cdot 10^6$ and the domain is 4 times larger in the span-wise direction. The result with the lowest resolution presented in this paper has about $1.39 \cdot 10^6$ DOFs and takes advantage of the unstructured nature of the grid by increasing the mesh density during the turbulent transition and reattachment, that is for $x/d \gtrapprox 4.5$ up to the outflow.

Physical discussion. This type of flow problem is reported to be very sensitive to the free-stream turbulence at the inlet (Tu). The free-stream flow was carefully manipulated to reproduce those reported by ERCOFTAC as well as previous numerical computations [46, 45]. In those works, a white-noise random perturbation was added at the inflow velocity to mimic the low experimental turbulence level, *i.e.*, Tu $< 0.2\%$. In the present work, due to an aggressive mesh coarsening in the far-field regions, the generation of a free-stream turbulence at inlet is unfeasible. In fact, the coarse spatial discretization at far-field would rapidly damp any random perturbation introduced upstream. Accordingly, the turbulent fluctuations were synthetically injected via a spatially-supported random forcing term in those regions of the domain where the mesh density is enough not to dissipate small scales. The random discrete forcing vector $\mathbf{f}_h$ is applied as a source term of the discrete momentum equations (6) and computed by multiplying a randomly generated three-dimensional discrete vector $\mathbf{r}_h$ to a function of the spatial coordinates x, y, z. This function is chosen to be a Gaussian distribution in the x direction and homogeneous in y−z such that

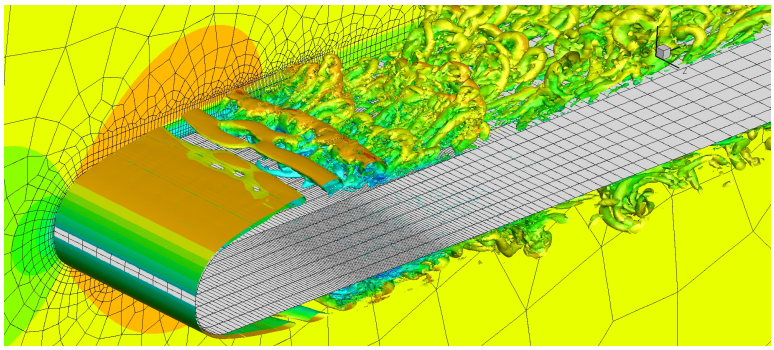
$$\mathbf{s}_h = \int_\Omega \mathbf{v}_h \cdot \mathbf{f}_h d\Omega = \int_\Omega \mathbf{v}_h \cdot \left(A e^{-\left(\frac{x_1 - \overline{x}_1}{2\mu}\right)^2} \right) \mathbf{r}_h d\Omega \tag{28}$$

where A, $\overline{x}_1$ and μ are, respectively, the amplitude coefficient, the location of the forcing plane, and the amplitude of the Gaussian support. We remark that the vector component is normalized such that $\|\mathbf{r}_h\| = 1$. The random vector is computed at each Gauss-quadrature point separately during the residual assembly. Since this vector changes not only in space, but also in time, it has been verified that the results are insensitive with respect to temporal refinements. The Gaussian function was centered in $\overline{x}_1/d = -3$, and the constants A, μ were adjusted, via a trial and error approach, to meet the experimental Tu. We avoid a fine control algorithm of the turbulent length-scale since the reattachment length is pretty insensitive to this value, see [47, 48, 49]. In the present configuration the expected turbulence intensity is met setting $A = 0.06$ and $\mu = 0.01$.

We remark that perturbing the velocity field through a forcing term in the momentum equations guarantees a divergence-free perturbation.



(a) Tu = 0%



(b) Tu = 0.2%

Figure 6: T3L1 test case, $k = 6$ solutions for different Tu levels. $\lambda_2 = -1$ iso-contour and periodic plane coloured by the streamwise velocity.

Fig. 6 depicts the instantaneous flow fields computed with Tu = 0% and Tu = 0.2%. In both cases, the quasi two-dimensional Kelvin-Helmholtz instabilities in the shear-layer region above the separation bubble and their convection downstream are observed. As expected, the low free-stream turbulence intensity value promotes the instability of the quasi two-dimensional structures arising from the upstream flow separation. For both the conditions, hairpin vortices developing after flow reattachment and the breakdown to turbulence are similar. Distortion along the spanwise direction is anticipated upstream in the Tu = 0.2% case.

As reported in previous studies, the bubble length is found to be very sensitive to the inlet turbulence intensity, see for example [50]. In particular, when increasing the Tu from 0% to 0.2% the bubble length reduces from $x_R/d = 3.90$ to 2.69, as shown by the statistically-converged

time and spanwise averaged velocity contours in Figs. 7 and 8. The length predicted for the Tu = 0.2% case is in a better agreement with the experimental data ($l/d = 2.75$) than other numerical computations [46, 45] (2.59 and 3.00, respectively). Moreover, we verified by lowering the polynomial degree of the dG discretization that our statistical average x_R/d is almost converged with respect to the spatial resolution: for $k = 5$ and $k = 4$ we obtain 2.70 and 2.73, respectively. Convergence of the statistics is also confirmed by polynomial degree independence observed for the skin friction coefficients, see Fig. 9. As opposite to the behavior documented in [46], no hysteresis effects are observed in our computations. Accordingly, if the random source term generating small turbulent perturbations is suddenly suppressed in the fully developed flow field at Tu = 0.2%, the solution of a zero free-stream turbulence case is quickly recovered.

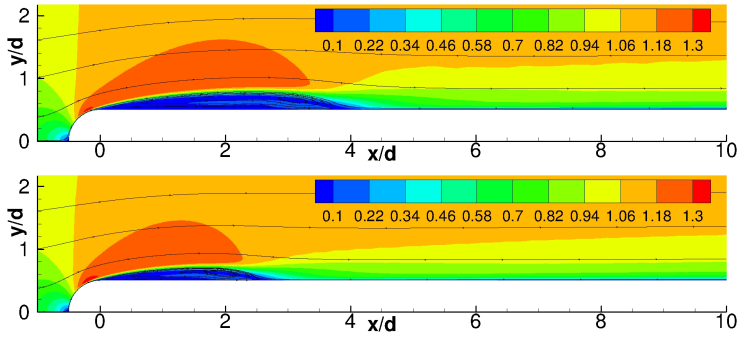


Figure 7: T3L1 test case. Effect of the Tu level. Average velocity magnitude iso-contours, $k = 6$ solutions. Top: Tu = 0.0%; Bottom: Tu = 0.2%.

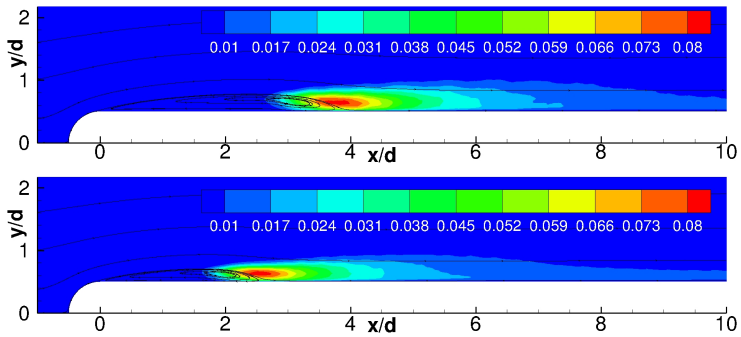


Figure 8: T3L1 test case. Effect of the Tu level. Turbulent kinetic energy iso-contours, $k = 6$ solutions. Top: Tu = 0.0%; Bottom: Tu = 0.2%.

Figure 10 compares velocity profiles with the experimental ones. We consider the mean stream-

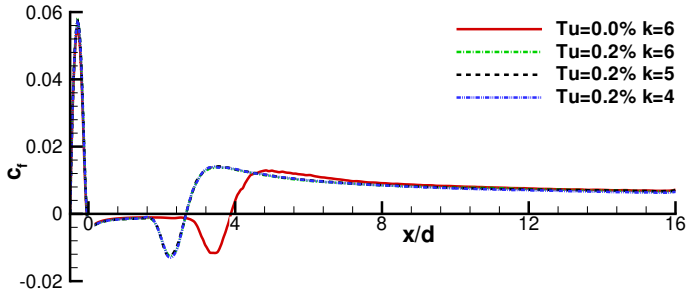


Figure 9: T3L1 test case. Effect of the Tu level on the skin friction coefficient c_f.

wise velocity $<u>$, and the velocity fluctuation (or velocity RMS), $<u'u'>$, as a function of the normal direction for different stations. Velocity is normalized by the local maximum velocity u_{max}, computed independently for each of the stations. The random forcing efficacy is demonstrated by the very good agreement with experimental data close to the plate stagnation point. We point out that for $x_1/l < 1.64$ improvements with respect to previous computational investigations are difficult to evaluate. As opposite, for $x_1/l > 1.64$ our results still compare favourably with the experimental data, while the matching is less evident in [45]. We stress that our velocity fluctuations compare favourably with the experiments up to $3.45l$, which was omitted in previous works [46, 45], see Figures 10 and 11. This supports the claim that present computations provide a larger fully resolved region, for all the polynomial degrees here considered. Note that some jumps at inter-element boundaries are still noticeable, especially for $k = 4$. Fig. 12 reports the computed averaged velocity profiles in wall units, for different stations located downstream to the reattachment region. For $x/l \geq 3.45$ the profile approaches the turbulent law of the wall, showing some discrepancies with respect to the equilibrium boundary layer in the outer layer. For the sake of comparison, the zero pressure gradient flat plate DNS result at $\mathrm{Re}_\theta = 300$ of Spalart [51] is reported together with the numerical solution at $x/l = 4.55$, which shows almost the same Re_θ. We remark the very good agreement between the experimental and numerical data at station $x/l = 3.45$, corresponding to $\mathrm{Re}_\theta \approx 270$.

Performance assessment. Table 7 reports the computational performances of the different solution strategies obtained for a dG approximation with $k = 6$, the same polynomial degree of reference employed in previous sections. The table aims at comparing the performance of the solution strategies accounting for CPU time, memory footprint as well as average number of GM-

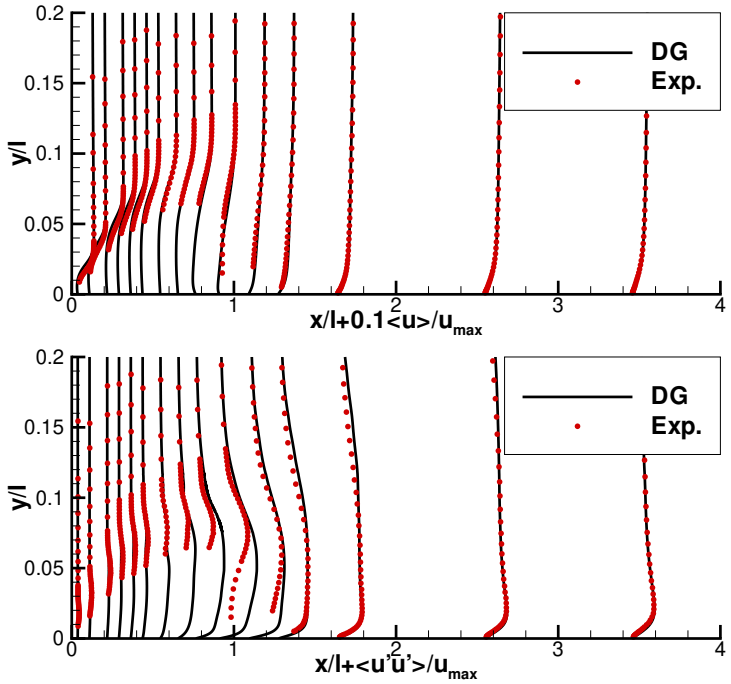


Figure 10: T3L1 test case, $k = 6$ solution for $Tu = 0.2\%$. Time- and spanwise-averaged velocity profiles in comparison with experimental data [52]. Mean (top) and RMS (bottom) flow velocity. Abscissas are non-dimensionalized using the experimental reattachment length $l/d = 2.75$.

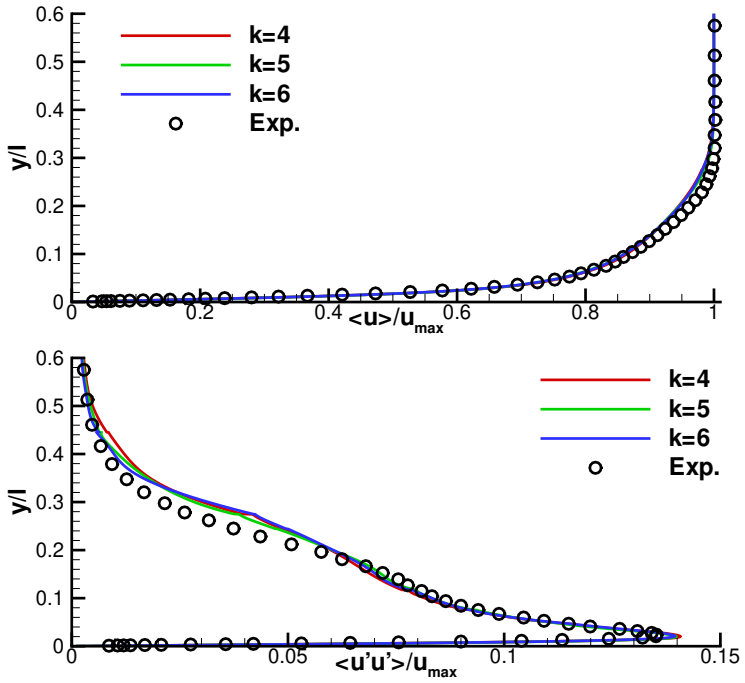


Figure 11: T3L1 test case, $k = 6$ solution for $Tu = 0.2\%$. Time- and spanwise-averaged velocity profiles in comparison with experimental data [52] at $x/l = 3.45$. Mean (top) and RMS (bottom) velocity.

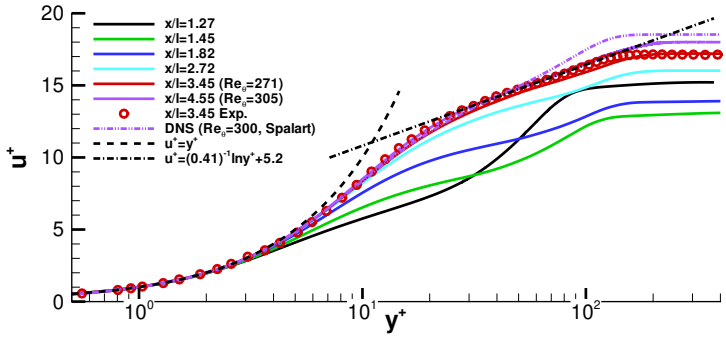


Figure 12: T3L1 test case, $k = 6$ solution for $Tu = 0.2\%$. Non dimensional stream-wise velocity profile for different values of x/l after reattachment in comparison to the theoretical law of the wall and DNS data [51].

RES iterations. The time step size of the third-order accurate linearly-implicit Rosenbrock-type time integration scheme was 16 and 8 times larger than those used in [45] and [46], respectively.

We point out that, as the number of curved elements small when compared to the number of affine elements, the degrees of exactness of quadrature rules neglects the second degree geometrical representation of cells close to leading edge. Since the boundary layer is still laminar at the leading edge, this under-integration does not affect the stability of the scheme as well as the accuracy of the numerical results.

Comparing the matrix-based (GMRES(MB)[BJ]) to the matrix-free solver (GMRES(MF)[BJ]), we observe the same computational efficiency but 40% less memory usage. On the other hand, the use of Additive Schwarz preconditioned GMRES (GMRES[ASM(1,ILU(0))]) decreases the overall parallel efficiency of the method: the CPU time increases by the 11% and the memory requirements raises by 60%, due to the increased number of overlapping block elements employed. The p-multigrid precondioned solver specs are as follows: FGMRES(MB or MF) as outer solver, a full p-multigrid interation with $L = 3$, $\kappa_\ell = 6, 2, 1$, GMRES(MB or MF)[EWBJ] smoother for $\ell = 0$ (8 iterations), GMRES(MB)[EWBJ] smoother for $\ell = 1$ (8 iterations) and GMRES(MB)[ASM(1,ILU(0))] smoother on the coarsest level ($\ell = 2$, 40 iterations). Note that for the finest level outer solver and smoother we consider both matrix-based and matrix-free implementations for the sake of comparison. The computational efficiency of the method improves considerably with respect to the reference (first column of Tab. 7): i) in a matrix-based framework a 50% decrease of the CPU time is observed and the memory requirements reduce by 35%, ii) in a matrix-free framework the CPU times gains are unaltered and the memory savings reach 85%. Such significant memory footprint reductions are mainly due the finest level strategy: only a block diagonal matrix is allocated and a small number of Krylov subspaces is employed for the GMRES algorithm. We remark that in this case the actual memory

Solver Prec	GMRES(MB) BJ	GMRES(MB) ASM(1,ILU(0))	GMRES(MF) BJ	FGMRES(MB) MG_{full}	FGMRES(MF) MG_{full}	FGMRES(MF) MG_{full} (LAG=3)
CPU Ratio	1	1.11	0.95	0.50	0.47	0.31
Memory Ratio	1	1.6	0.6	0.65	0.15	0.15
ITs	115	72	115	3.0	3.0	3.31

Table 7: Performance comparison of the solver on the T3L1 test case. Computational time, total memory footprint non-dimensionalized with the GMRES(MB)[BJ] solver, and average number of GMRES iterations per time step, for the BJ, ASM(1,ILU(0)) and p-multigrid preconditioners (see text for settings details). Results obtained on 540 Intel Xeon CPUs of Marconi-A1@CINECA.

footprint has been computed through the PETSc library, and it is in line with the values that can be estimated a-priori using the model of Section 2.6.

As a further optimization of the matrix-free approach we consider the possibility to lag the computation of the system matrix employed for preconditioning purposes, this means that the preconditioner is frozen for several time steps. Clearly this strategy reduces assembly times but degrades convergence rates if the discrepancy between the matrix and the preconditioner gets too severe. In the present computation optimal performance are achieved by lagging the operators evaluation for 3 time steps. By doing so, the CPU time is further reduced to the 0.31 of the baseline, see Tab. 7, which corresponds to a speed-up of 3.22. As a side effect, the average number of GMRES iterations slightly increases, from 3.0 to 3.31, due to a loss of efficiency of the multigrid preconditioner. We remark that, in a matrix-free framework, lagging the preconditioners only acts on the coarse space multigrid operators, while the finest space still gets updated thanks to the matrix-free approach.

4.2. Boeing rudimentary landing gear test case

The final validation case reported in this work deals with the implicit LES of the incompressible flow around the Boeing rudimentary landing gear (RLG). The purpose of the test is to demonstrate the applicability of the solution strategies proposed in this work within an industrially relevant test case.

The RLG was designed by Spalart *et al.* [53], and experimentally studied in [54], to become a benchmark for testing turbulence modelling approaches. The test case was also included within the test case suite of the ATAAC EU-funded project [55]. The flow conditions involve a Reynolds number of 10^6, based on the freestream velocity $V_\infty = 40\ m/s$ and on the wheel diameter $D = 0.406\ m$. The Mach number of the experiments was $M = 0.12$, which represents almost an incompressible flow problem. While the structural elements of the landing gear are rectangular to fix the location of the separation points, the boundary layers on the wheels are tripped, see [53]. The structure of the unsteady flow around the landing gear is mainly characterized by large separated and recirculating regions on the wheels and axles, as well as by the front-rear wheel interaction, which make the use of unsteady scale-resolving simulation mandatory.

The computational domain is delimited by one symmetry plane, three inviscid walls, one inflow, one outflow, and the landing gear wall surface according to [55]. A snapshot of the grid showing the wall surface (red), the symmetry plane discretization (black) and an internal slice

(blue) is reported in Figure 13. The mesh employed takes advantage of the symmetry of the problem and it discretizes only the half of the domain. The grid was made by $115 \cdot 10^3$ hexahedral elements with second-order geometrical representation of the curved boundaries. It shows a severe wall refinement to accommodate a suitable wall resolution given the high Reynolds number of the case. The first cell height is $\delta y = 6.054 \cdot 10^{-5} D$, which provides an equivalent wall normal resolution of $1.851 \cdot 10^{-5}$, obtained by dividing for $(n_v)^{1/3}$, with n_v the number of DoFs per element. Exploiting the maximum value of the skin friction coefficient over the entire wall boundary obtained during the post-processing phase, the grid allows a maximum wall normal resolution of $y^+ = 2.8$. The estimated minimum aspect ratio of the cell is of the order of 10^4, which increases considerably the condition number of the iteration matrix.

The solution has been obtained by using $k = 4$ polynomials, providing a total of $4.025 \cdot 10^6$ degrees of freedom, while the four-stage, order-three ROSI2PW scheme was employed for time integration. Using as reference quantities the free stream velocity and the wheel diameter, the non-dimensional time step size was $\delta t = 0.001$. To compute the average fields, the solution was advanced in time for roughly 50 convective times, some of them performed using a lower-order space discretization. The average process lasted roughly $T = 23$ convective times, which may be not enough to have converged first-order statistics. However, it has been verified that the average quantities did not change sensibly from $T > 17.5$.

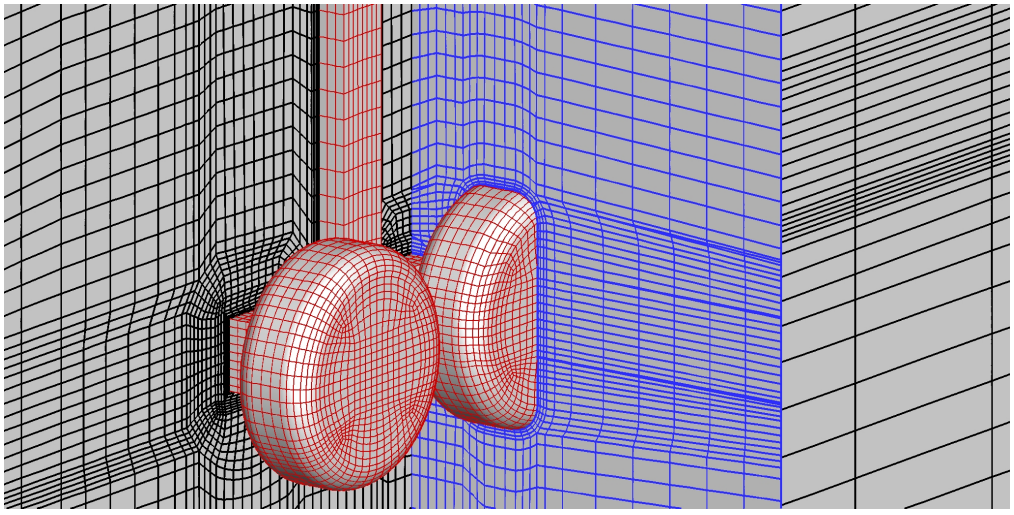


Figure 13: Boeing rudimentary landing gear test case. Details of the multiblock structured grid provided by DLR under the ATAAC and TILDA European projects.

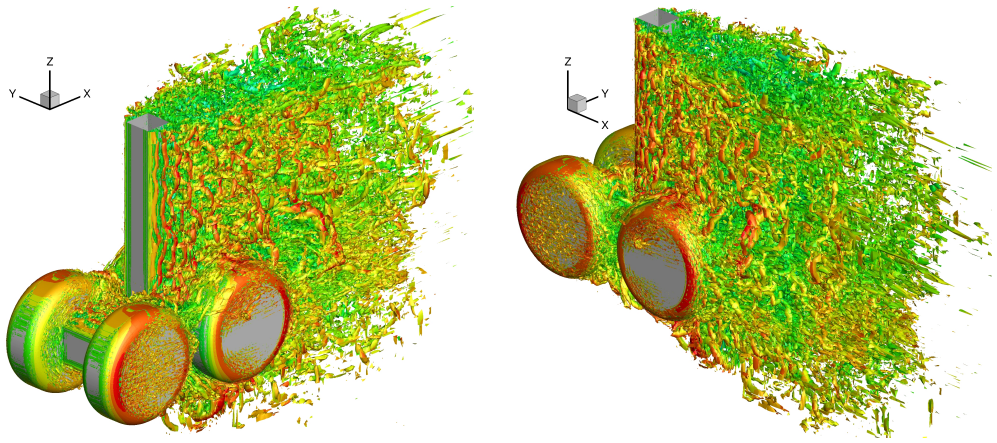


Figure 14: RLG test case at $Re = 10^6$. Incompressible flow solution using $k = 4$ polynomials. λ_2 iso-contour coloured by stream-wise velocity magnitude. Front view (left) and rear view (right).

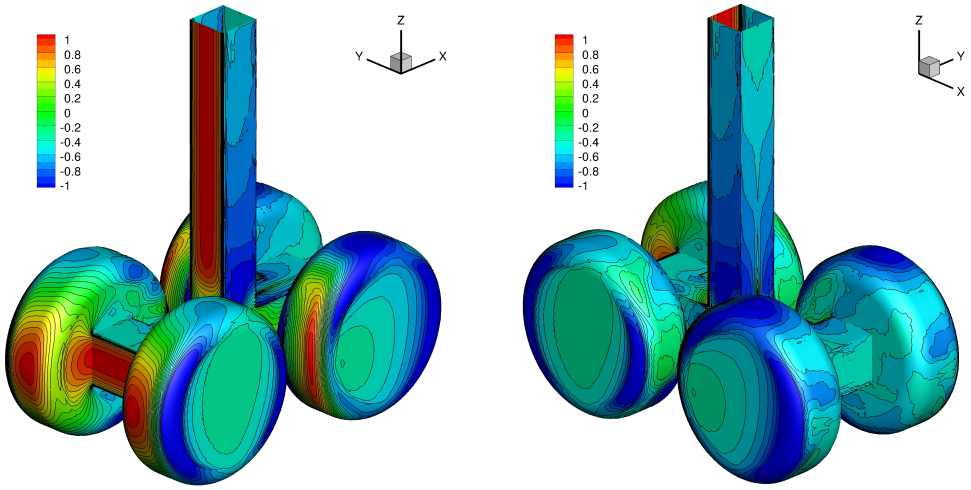


Figure 15: RLG test case at $Re = 10^6$. Incompressible flow solution using $k = 4$ polynomials. Mean pressure coefficient C_p contours on the wall surface. Front view (left) and rear view (right).

Physical discussion. Figure 14 shows the features of the flow field through the instantaneous λ_2 iso-contour plot coloured by the stream-wise velocity magnitude. The shape of the iso-contours suggests that the flow is mainly attached to the wheels, although a very small laminar separation can be observed on the fore side of the wheel.

The averaged fields in terms of pressure coefficient C_p, the root mean square value of the pressure coefficient C_p^{RMS} on the landing gear are reported in Figure 15, 16. A qualitative agreement with the surface plots reported in [53, 56], obtained through a hybrid RANS/LES approach, can

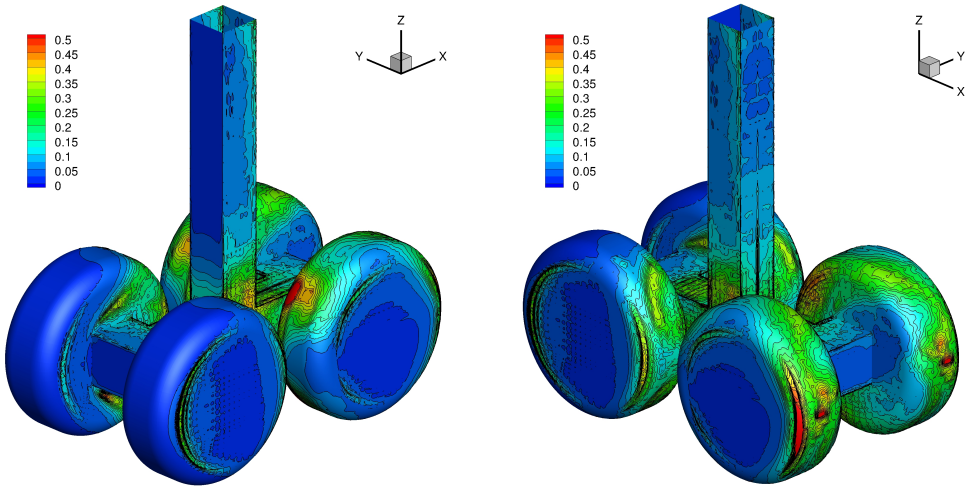


Figure 16: RLG test case at $Re = 10^6$. Incompressible flow solution using $k = 4$ polynomials. Pressure coefficient RMS, C_p^{RMS} contours on the wall surface. Front view (left) and rear view (right).

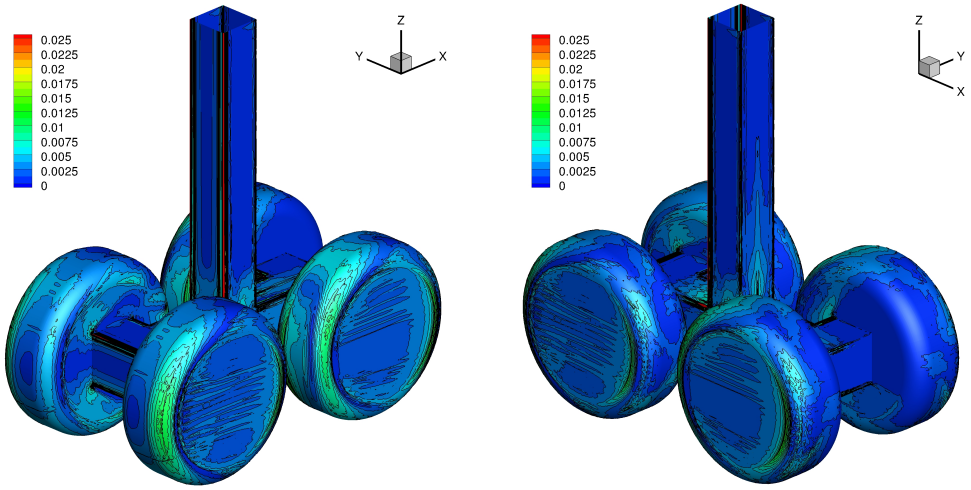


Figure 17: RLG test case at $Re = 10^6$. Incompressible flow solution using $k = 4$ polynomials. Mean skin friction coefficient C_f contours on the wall surface. Front view (left) and rear view (right).

be observed especially as regards the front views. On the other hand, the rear view highlights the presence of pressure oscillations that suggest a very coarse space resolution. Those oscillations are even more evident if first order statistics (*e.g.* C_p^{RMS}) or the skin friction coefficient (which involve state derivatives) are considered. Figure 17 reports the surface plot of the skin friction coefficient $C_f = 2\tau_w/\rho V_\infty^2$, which looks qualitatively similar to those reported in previous numerical simulations [53, 56].

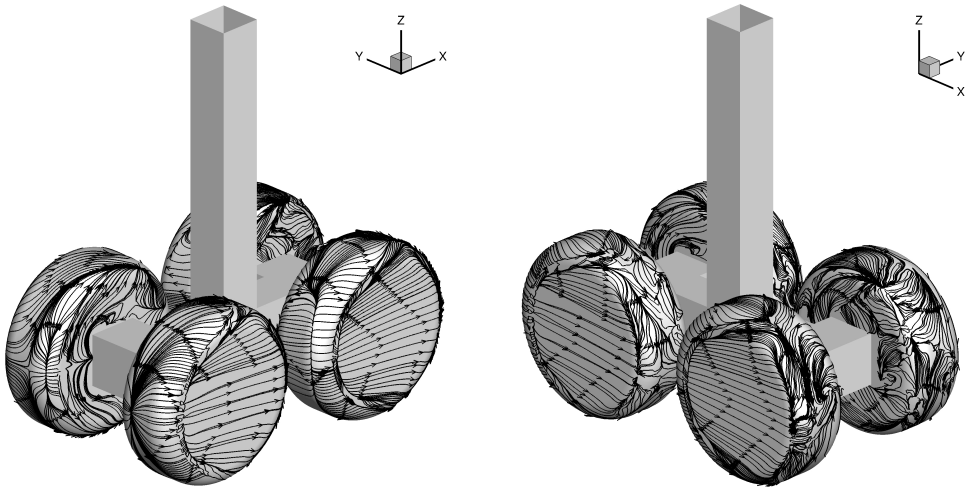
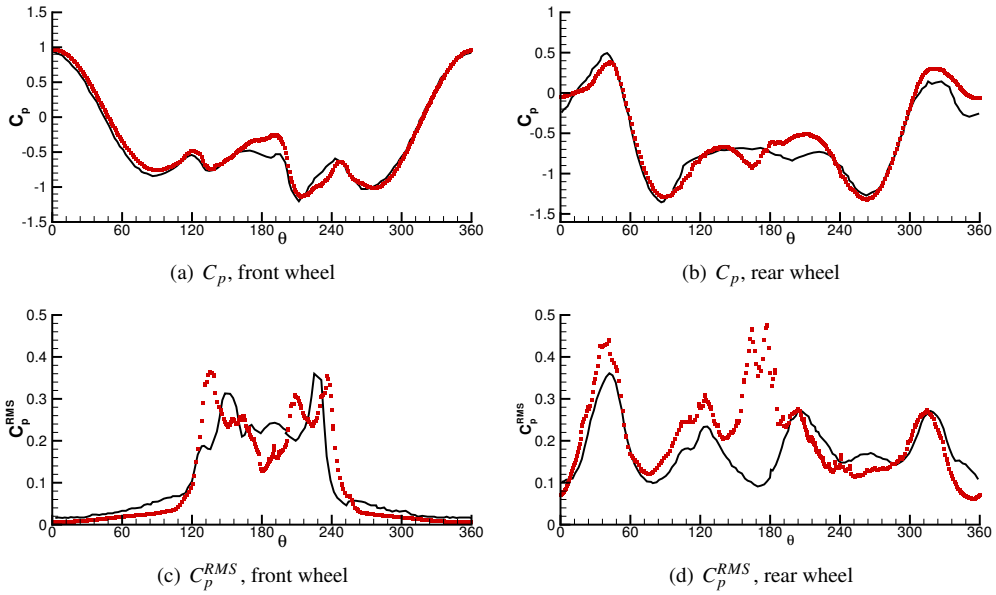


Figure 18: RLG test case at $Re = 10^6$. Incompressible flow solution using $k = 4$ polynomials. Average wall streamline path. Front view (left) and rear view (right).

Figure 18 show the streamline patterns on the wheels. The patterns show the bifurcation line of separation and reattachment on the front and rear wheels as reported in [56]. However, the simulation shows on both sides of the wheels a region with separated flow and reversed streamline patterns. Such a region seems to be different to that reported by the experiments [54] as well as previous numerical simulations. No transition tripping was employed in the current simulation, differently to what has been done for the experiments and previous numerical simulations based on RANS and hybrid RANS/LES modeling.

Figure 19 reports the average pressure coefficient C_p and its root mean square value C_p^{RMS} on the mid-line of the wheels, versus the azimuthal angle θ, in comparison with experimental data from NAL [54]. While the pressure coefficient compare favourably with experimental data, its root mean square value shows a more oscillatory behaviour, originating from low spatial resolution and possibly a too short averaging time. However, the locations of the peaks of fluctuation as well as its value is pretty well captured.

Performance assessment. Linear systems arising from the time integration were solved using FGMRES preconditioned with a p-multigrid strategy with similar settings to that of Table 6, employing an additive Schwarz preconditioned smoothers on the coarsest level of the multigrid iteration, and an element-wise block Jacobi method on the other levels to maximise the scalability of the algorithm. Despite working with an average of 19 elements per partition in such a complex



(a) C_p, front wheel

(b) C_p, rear wheel

(c) C_p^{RMS}, front wheel

(d) C_p^{RMS}, rear wheel

Figure 19: RLG test case at $Re = 10^6$. Incompressible flow solution using $k = 4$ polynomials. Average pressure coefficient C_p and RMS C_p^{RMS} distribution on the mid-line of the fore and back wheels versus the azimuthal angle θ. Numerical simulations (red dots) compared to experimental data (black solid lines).

test case, the iterative solver converged using an average of 3.5 iterations per stage, confirming the efficacy of this preconditioning approach. The present computation has been performed on roughly $6 \cdot 10^3$ cores of the Marconi A1 cluster hosted by CINECA, using a total wall clock time of 94 hours. The average wall time per convective time unit was roughly 4 hours.

5. Conclusion

The paper presents a p-multigrid preconditioner strategy applied to the solution of linear system arising from linearly-implicit Rosenbrock-type time discretizations. The algorithm relies on a matrix-free implementation of both the outer FGMRES solver as well as the finest level smoother, while matrix-based GMRES smoothers are employed on coarse levels. Coarse operators of lower polynomial degree are built using a subspace inherited approach.

The performance of the algorithm has been evaluated on test cases of growing complexity. First, we deal with unsteady laminar flows, *i.e.*, the flow around a two-dimensional cylinder and a sphere, showing that the p-multigrid preconditioned solver can be used to achieve optimal convergence rates, outperforming standard single-grid preconditioned iterative solvers from a CPU time viewpoint in practical parallel computations. In particular, we consider and parallel computations (up to 576 cores and 6 elements per partition), with speed-ups roughly ranging from 1.5 to 3.5. Finally, we perform ILES of the incompressible turbulent flow over a rounded-leading edge plate with different free-stream turbulent intensities. High-order accurate $k = 6$ solutions are compared with published numerical results and wind tunnel experiments. The solver strategy is profiled and compared with state-of-the-art single-grid solvers running on large HPC facilities. We show that, if a block-diagonal preconditioner is employed on the finest level, the algorithm reduces the memory footprint of the solver of about 92% of the standard matrix-based implementation. Interestingly, besides the memory savings, the p-multigrid preconditioned FGMRES solver is also three times faster than the best performing single-grid solver. As proof of concept, we report the ILES of the Boeing rudimentary landing gear at $Re = 10^6$. Increasing the complexity of the problems has not required tuning of p-multigrid parameters confirming the robustness of the proposed approach.

Future works involve the implementation of an adaptive strategy for the choice of the quadrature degree of exactness, which can be adapted in view of the actual amount of curvature of mesh faces, in order to optimise the computation of the residuals vector and thus the overall performance of the matrix-free algorithm.

Acknowledgements

We acknowledge the CINECA award, under the ISCRA initiative (grant numbers HP10BEE6C4, HP10CPSWZ2, HP10BMA1AP, HP10CKYRYE, HP10CE90VW), for the avail-

ability of high performance computing resources and support. Professor Michele Napolitano (Politecnico di Bari) and the co-authors of [52] are also gratefully acknowledged for sharing the experimental data of the T3L1 test case. Dr. Ralf Hartman from DLR, the EU-funded projects "Advanced Turbulence Simulation for Aerodynamic Application Challenges" (ATAAC) and "Towards Industrial LES/DNS in Aeronautics – Paving the Way for Future Accurate CFD" (TILDA) are also acknowledged for sharing the mesh of the Boeing rudimentary landing gear test case.

Appendix A. ROSI2PW scheme

We here report the coefficients of the ROSI2PW scheme [31] used for the time integration of the set of incompressible Navier–Stokes equations. Coefficients $\hat{\mathsf{m}}$, reported for completeness, are those of the embedded ROSI2PW Runge–Kutta scheme.

$\gamma = 4.3586652150845900E-01$	
$\mathsf{a}_{21} = 8.7173304301691801E-01$	$\mathsf{c}_{21} = -8.7173304301691801E-01$
$\mathsf{a}_{31} = -7.9937335839852708E-01$	$\mathsf{c}_{31} = 3.0647867418622479E+00$
$\mathsf{a}_{32} = -7.9937335839852708E-01$	$\mathsf{c}_{32} = 3.0647867418622479E+00$
$\mathsf{a}_{41} = 7.0849664917601007E-01$	$\mathsf{c}_{41} = -1.0424832458800504E-01$
$\mathsf{a}_{42} = 3.1746327955312481E-01$	$\mathsf{c}_{42} = -3.1746327955312481E-01$
$\mathsf{a}_{43} = -2.5959928729134892E-02$	$\mathsf{c}_{43} = -1.4154917367329144E-02$
$\mathsf{m}_1 = 6.0424832458800504E-01$	$\hat{\mathsf{m}}_1 = 4.4315753191688778E-01$
$\mathsf{m}_2 = -3.6210810811598324E-32$	$\hat{\mathsf{m}}_2 = 4.4315753191688778E-01$
$\mathsf{m}_3 = -4.0114846096464034E-02$	$\hat{\mathsf{m}}_3 = 0.0000000000000000E+00$
$\mathsf{m}_4 = 4.3586652150845900E-01$	$\hat{\mathsf{m}}_4 = 1.1368493616622447E-01$

Table A.8: ROSI2PW scheme coefficients from [31].

References

References

[1] F. Bassi, L. Botti, A. Colombo, A. Crivellini, A. Ghidoni, F. Massa, On the development of an implicit high-order Discontinuous Galerkin method for DNS and implicit LES of turbulent flows, European Journal of Mechanics-B/Fluids 55 (2016) 367–379.

[2] J.-B. Chapelier, M. De La Llave Plata, F. Renac, E. Lamballais, Evaluation of a high-order discontinuous Galerkin method for the DNS of turbulent flows, Computers & Fluids 95 (2014) 210–226.

[3] C. C. Wiart, K. Hillewaert, L. Bricteux, G. Winckelmans, Implicit LES of free and wall-bounded turbulent flows based on the discontinuous Galerkin/symmetric interior penalty method, International Journal for Numerical Methods in Fluids 78 (6) (2015) 335–354.

[4] F. Bassi, A. Crivellini, D. Di Pietro, S. Rebay, An implicit high-order discontinuous Galerkin method for steady and unsteady incompressible flows, Computers & Fluids 36 (10) (2007) 1529–1546.

[5] A. Uranga, P.-O. Persson, M. Drela, J. Peraire, Implicit large eddy simulation of transition to turbulence at low reynolds numbers using a discontinuous Galerkin method, International Journal for Numerical Methods in Engineering 87 (1-5) (2011) 232–261.

[6] F. Bassi, L. Botti, A. Colombo, A. Ghidoni, F. Massa, Linearly implicit Rosenbrock-type Runge–Kutta schemes applied to the discontinuous Galerkin solution of compressible and incompressible unsteady flows, Computers & Fluids 118 (2015) 305–320.

[7] A. Crivellini, F. Bassi, An implicit matrix-free discontinuous Galerkin solver for viscous and turbulent aerodynamic simulations, Computers & Fluids 50 (1) (2011) 81 – 93.

[8] A. Sarshar, P. Tranquilli, B. Pickering, A. McCall, C. J. Roy, A. Sandu, A numerical investigation of matrix-free implicit time-stepping methods for large CFD simulations, Computers & Fluids 159 (2017) 53–63.

[9] M. Franciolini, A. Crivellini, A. Nigro, On the efficiency of a matrix-free linearly implicit time integration strategy for high-order discontinuous Galerkin solutions of incompressible turbulent flows, Computers & Fluids 159 (2017) 276 – 294. doi:https://doi.org/10.1016/j.compfluid.2017.10.008.

[10] A. Crivellini, M. Franciolini, A. Nigro, A matrix-free incompressible DG algorithm for the simulation of turbulent flows, in: Progress in Turbulence VII, Springer, 2017, pp. 153–159.

[11] L. T. Diosady, S. M. Murman, Tensor-product preconditioners for higher-order space–time discontinuous galerkin methods, Journal of Computational Physics 330 (2017) 296–318.

[12] W. Pazner, P.-O. Persson, Approximate tensor-product preconditioners for very high order discontinuous Galerkin methods, Journal of Computational Physics 354 (2018) 344–369.

[13] L. T. Diosady, S. M. Murman, Scalable tensor-product preconditioners for high-order finite-element methods: Scalar equations, Journal of Computational Physics (2019).

[14] B. Helenbrook, D. Mavriplis, H. Atkins, Analysis of *p*-multigrid for continuous and discontinuous finite element discretizations, in: 16th AIAA Computational Fluid Dynamics Conference, 2003, p. 3989.

[15] B. T. Helenbrook, H. L. Atkins, Application of *p*-multigrid to discontinuous Galerkin formulations of the poisson equation, AIAA journal 44 (3) (2006) 566–575.

[16] F. Bassi, S. Rebay, Numerical solution of the Euler equations with a multiorder discontinuous finite element method, in: Computational Fluid Dynamics 2002, Springer, 2003, pp. 199–204.

[17] 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, Journal of Computational Physics 207 (1) (2005) 92–113.

[18] F. Prill, M. Lukáčová-Medviďová, R. Hartmann, Smoothed aggregation multigrid for the discontinuous Galerkin method, SIAM Journal on Scientific Computing 31 (5) (2009) 3503–3528.

[19] M. Wallraff, R. Hartmann, T. Leicht, Multigrid solver algorithms for DG methods and applications to aerodynamic flows, in: IDIHOM: Industrialization of High-Order Methods-A Top-Down Approach, Springer, 2015, pp. 153–178.

[20] 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.

[21] 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.

[22] L. T. Diosady, D. L. Darmofal, Preconditioning methods for discontinuous Galerkin solutions of the Navier–Stokes equations, Journal of Computational Physics 228 (11) (2009) 3917–3935.

[23] L. Botti, A. Colombo, F. Bassi, *h*-multigrid agglomeration based solution strategies for discontinuous Galerkin discretizations of incompressible flow problems, Journal of Computational Physics 347 (2017) 382–415.

[24] A. Crivellini, M. Franciolini, A. Nigro, An implicit discontinuous Galerkin method with reduced memory footprint for the simulation of turbulent flows, in: Direct and Large-Eddy Simulation XI, Springer, 2019, pp. 69–74.

[25] F. Bassi, S. Rebay, A high-order accurate discontinuous finite element method for the numerical solution of the compressible Navier–Stokes equations, J. Comput. Phys. 131 (1997) 267–279.
[26] 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.
[27] D. N. Arnold, F. Brezzi, B. Cockburn, L. D. Marini, Unified analysis of discontinuous Galerkin methods for elliptic problems, SIAM J. Numer. Anal. 39 (5) (2002) 1749–1779.
[28] F. Bassi, A. Crivellini, D. Di Pietro, S. Rebay, An artificial compressibility flux for the discontinuous Galerkin solution of the incompressible Navier-Stokes equations, Journal of Computational Physics 218 (2) (2006) 794–815. doi:10.1016/j.jcp.2006.03.006.
[29] A. Crivellini, V. D'Alessandro, F. Bassi, Assessment of a high-order discontinuous Galerkin method for incompressible three-dimensional Navier–Stokes equations: Benchmark results for the flow past a sphere up to Re=500, Computers & Fluids 86 (0) (2013) 442 – 458. doi:http://dx.doi.org/10.1016/j.compfluid.2013.07.027.
[30] F. Bassi, A. Ghidoni, A. Perbellini, S. Rebay, A. Crivellini, N. Franchina, M. Savini, A high-order Discontinuous Galerkin solver for the incompressible RANS and k-ω turbulence model equations, Computers & Fluids 98 (2014) 54–68. doi:10.1016/j.compfluid.2014.02.028.
[31] J. Rang, L. Angermann, New Rosenbrock methods of order 3 for PDAEs of index 2, in: Proceedings of Equadiff-11 2005, Comenius University Press (Bratislava), 2007, pp. 385–394.
[32] D. Knoll, D. Keyes, Jacobian-free Newton-Krylov methods: a survey of approaches and applications, J. Comput. Phys. 193 (2) (2004) 357–397.
[33] M. Pernice, H. F. Walker, NITSOL: A Newton iterative solver for nonlinear systems, SIAM Journal on Scientific Computing 19 (1) (1998) 302–318.
[34] A. Crivellini, F. Bassi, An implicit matrix-free discontinuous Galerkin solver for viscous and turbulent aerodynamics simulations, Comput. & Fluids 50 (1) (2011) 81 – 93.
[35] S. Balay, J. Brown, K. Buschelman, W. D. Gropp, D. Kaushik, M. G. Knepley, L. C. McInnes, B. F. Smith, H. Zhang, PETSc Web page, http://www.mcs.anl.gov/petsc (2013).
[36] L. Botti, A. Colombo, A. Crivellini, M. Franciolini, {h-p-hp}-Multilevel discontinuous galerkin solution strategies for elliptic operators, International Journal of Computational Fluid Dynamics (2019) 1–9.
[37] P. F. Antonietti, B. Ayuso de Dios, S. Brenner, L. Sung, Schwarz methods for a preconditioned WOPSIP method for elliptic problems, Computational Methods in Applied Mathematics 12 (3) (2012) 241–272. doi:10.2478/cmam-2012-0021.
[38] Y. Saad, A flexible inner-outer preconditioned GMRES algorithm, SIAM Journal on Scientific Computing 14 (2) (1993) 461–469.
[39] J. R. Meneghini, F. Saltara, C. L. R. Siqueira, J. A. Ferrari Jr, Numerical simulation of flow interference between two circular cylinders in tandem and side-by-side arrangements, Journal of fluids and structures 15 (2) (2001) 327–350.
[40] 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.
[41] T. A. Johnson, V. C. Patel, Flow past a sphere up to a Reynolds number of 300, Journal of Fluid Mechanics 378 (1999) 19–70.
[42] A. G. Tomboulides, S. A. Orszag, Numerical investigation of transitional and weak turbulent flow past a sphere, Journal of Fluid Mechanics 416 (2000) 45–73.
[43] P. Ploumhans, G. Winckelmans, J. K. Salmon, A. Leonard, M. S. Warren, Vortex methods for direct numerical simulation of three-dimensional bluff body flows: application to the sphere at Re= 300, 500, and 1000, Journal of Computational Physics 178 (2) (2002) 427–463.
[44] A. Crivellini, M. Franciolini, A. Colombo, F. Bassi, OpenMP parallelization strategies for a discontinuous Galerkin solver, International Journal of Parallel Programming 178 (2) (2018) 427–463.
[45] M. Langari, Z. Yang, Numerical study of the primary instability in a separated boundary layer transition under elevated free-stream turbulence, Physics of Fluids 25 (7) (2013) 074106. doi:10.1063/1.4816291.
[46] Z. Yang, P. R. Voke, Large-eddy simulation of boundary-layer separation and transition at a change of surface curvature, Journal of Fluid Mechanics 439 (2001) 305–333. doi:10.1017/S0022112001004633.
[47] R. Hillier, N. Cherry, The effects of stream turbulence on separation bubbles, Journal of Wind Engineering and Industrial Aerodynamics 8 (1) (1981) 49 – 58. doi:https://doi.org/10.1016/0167-6105(81)90007-6.
[48] Y. Nakamura, S. Ozono, The effects of turbulence on a separated and reattaching flow, Journal of Fluid Mechanics 178 (1987) 477–490. doi:10.1017/S0022112087001320.
[49] Z. Yang, I. E. Abdalla, Effects of free-stream turbulence on a transitional separated-reattached flow over a flat plate with a sharp leading edge, International Journal of Heat and Fluid Flow 30 (5) (2009) 1026 – 1035, the 3rd International Conference on Heat Transfer and Fluid Flow in Microscale. doi:https://doi.org/10.1016/j.ijheatfluidflow.2009.04.010.
[50] E. Lamballais, J. Silvestrini, S. Laizet, Direct numerical simulation of flow separation behind a rounded leading

edge: Study of curvature effects, International Journal of Heat and Fluid Flow 31 (3) (2010) 295–306.
[51] P. R. Spalart, Direct simulation of a turbulent boundary layer up to $Re_\theta = 1410$, Journal of Fluid Mechanics 187 (1988) 61–98. doi:10.1017/S0022112088000345.
[52] L. Cutrone, P. De Palma, G. Pascazio, M. Napolitano, Predicting transition in two-and three-dimensional separated flows, International Journal of Heat and Fluid Flow 29 (2) (2008) 504–526.
[53] P. Spalart, M. Shur, M. Strelets, A. Travin, Initial rans and ddes of a rudimentary landing gear, in: Progress in Hybrid RANS-LES Modelling, Springer, 2010, pp. 101–110.
[54] L. Venkatakrishnan, N. Karthikeyan, K. Mejia, Experimental studies on a rudimentary four-wheel landing gear, AIAA journal 50 (11) (2012) 2435–2447.
[55] S.-H. Peng, M. Strelets, Test case ST12, Boeing rudimentary landing gear., in: ATAAC.
[56] Q.-L. Dong, H.-Y. Xu, Z.-Y. Ye, Numerical investigation of unsteady flow past rudimentary landing gear using DDES, LES and URANS, Engineering Applications of Computational Fluid Mechanics 12 (1) (2018) 689–710.