An Implicit Matrix-Free Discontinuous Galerkin Solver for Turbulent Flows

Andrea Crivellini[1], Leonardo Pelagalli[1], Francesco Bassi[2]
[1]*Department of Energy, University Politecnica delle Marche, Italy*
E-mail: a.crivellini@univpm.it, l.pelagalli@univpm.it

[2]*Department of Industrial Engineering, University of Bergamo, Italy*
E-mail: francesco.bassi@unibg.it

Keywords: Discontinuous Galerkin, RANS Equations, Computational Fluid Dynamics, implicit method, matrix-free.

SUMMARY. This paper presents some recent advancements of the computational efficiency of a Discontinuous Galerkin (DG) solver for the Navier-Stokes (NS) and RANS (Reynolds Averaged Navier Stokes) equations. The implementation and the performances of a Newton-Krylov matrix-free method is presented and discussed. The solutions of the viscous flow around a Naca 0012 airfoil ($\mathrm{M}_\infty = 0.5$, $\alpha = 2°$, $\mathrm{Re} = 500$) and the turbulent flow around the L1T2 3-element airfoil ($\mathrm{M}_\infty = 0.197$, $\alpha = 20.18$, $\mathrm{Re} = 3.52 \cdot 10^6$) have been used to asses the algorithm.

1 INTRODUCTION

The growing interest that the DG method has been receiving in the recent past is due to various attractive features of the method. DG methods are in fact finite element methods which account for the physics of wave propagation by means of Riemann solvers as in upwind finite volume methods but, unlike the latter, can achieve higher-order accuracy on general unstructured grids using high degree polynomials as is customary in the classical (continuous) finite element method.

Despite these advantages, DG methods for real life applications still require to be assessed and improved in many respects, such as applicability to complex flow models, shock-capturing properties and computational efficiency. In this paper we will focus on the latter topic. The solution of the large block sparse linear system arising from an implicit time DG discretizzation of the NS and RANS equations becomes prohibitively expensive as the grid density and/or the order of polynomial approximation increases for this reason effective numerical strategies must be developed.

2 DG SPACE DISCRETIZATION

The complete set of compressible Navier-Stokes or RANS with k-ω turbulence model equations can be written in compact form as

$$\frac{\partial \mathbf{u}}{\partial t} + \boldsymbol{\nabla} \cdot \mathbf{F}_c(\mathbf{u}) + \boldsymbol{\nabla} \cdot \mathbf{F}_v(\mathbf{u}, \boldsymbol{\nabla}\mathbf{u}) + \mathbf{s}(\mathbf{u}, \boldsymbol{\nabla}\mathbf{u}) = \mathbf{0}, \tag{1}$$

where $\mathbf{u}, \mathbf{s} \in \mathbb{R}^M$ denote the vectors of the M conservative variables and source terms, $\mathbf{F}_c, \mathbf{F}_v \in \mathbb{R}^M \otimes \mathbb{R}^N$ denote the inviscid and viscous flux functions, respectively, and N is the space dimension. In turbulent simulations in order to deal with the stiffness of the k-ω equations we resorted to a non-standard implementation of the two equation differential model. The main features of this implementation consist in (i) using as variable $\log \omega$ rather than ω and (ii) fulfilling the realizability constraints and the Schwarz inequality on turbulent stress. A detailed description of the model can be found in [1].

Multiplying equation (1) by an arbitrary test function ϕ, integrating over the domain Ω and integrating by parts the divergence terms, the weak form of equation (1) reads:

$$\int_\Omega \phi \frac{\partial \mathbf{u}}{\partial t}\,\mathrm{d}\mathbf{x} - \int_\Omega \boldsymbol{\nabla}\phi \cdot \mathbf{F}(\mathbf{u}, \boldsymbol{\nabla}\mathbf{u})\,\mathrm{d}\mathbf{x} + \int_{\partial\Omega} \phi \mathbf{F}(\mathbf{u}, \boldsymbol{\nabla}\mathbf{u}) \cdot \mathbf{n}\,\mathrm{d}\sigma + \int_\Omega \phi \mathbf{s}(\mathbf{u}, \boldsymbol{\nabla}\mathbf{u})\,\mathrm{d}\mathbf{x} = \mathbf{0}, \quad (2)$$

where $\mathbf{F}$ is the sum of the inviscid and viscous fluxes.

In order to construct a DG discretization of equation (2), we consider an approximation Ω_h of Ω and a triangulation $\mathcal{T}_h = \{K\}$ of Ω_h, *i.e.*, a set of non-overlapping elements K not necessarily simplices. We denote with $\mathcal{E}_h^0$ the set of internal element faces, with $\mathcal{E}_h^\partial$ the set of boundary element faces and with $\mathcal{E}_h = \mathcal{E}_h^0 \cup \mathcal{E}_h^\partial$ their union. We moreover set

$$\Gamma_h^0 = \bigcup_{e \in \mathcal{E}_h^0} e, \qquad \Gamma_h^\partial = \bigcup_{e \in \mathcal{E}_h^\partial} e, \qquad \Gamma_h = \Gamma_h^0 \cup \Gamma_h^\partial. \quad (3)$$

The solution is approximated on $\mathcal{T}_h$ as a piecewise polynomial function possibly discontinuous on element interfaces, *i.e.*, we assume the following space setting for each component $u_{h_i} = u_{h_1}, \ldots, u_{h_M}$ of the numerical solution $\mathbf{u}_h$:

$$u_{h_i} \in \Phi_h \overset{\text{def}}{=} \left\{ \phi_h \in L^2(\Omega) : \phi_h|_K \in \mathbb{P}_k(K)\ \forall K \in \mathcal{T}_h \right\} \quad (4)$$

for some polynomial degree $k \geq 0$, being $\mathbb{P}_k(K)$ the space of polynomials of global degree at most k on the element K. To overcome the ill-conditioning of elemental mass matrices for higher-order polynomials on high aspect ratio and curved elements we have chosen a hierarchical and orthogonal set of shape functions defined in physical space. This set is obtained using a modified Gram-Schmidt procedure considering as a starting point a set of monomial functions of the same degree k.

Following Brezzi et al. [2], it is also convenient to introduce the jump and average trace operators, which on a generic internal face $e \in \mathcal{E}_h^0$ are defined as:

$$[\![q]\!] \overset{\text{def}}{=} q^+\mathbf{n}^+ + q^-\mathbf{n}^-, \qquad \{\cdot\} \overset{\text{def}}{=} \frac{(\cdot)^+ + (\cdot)^-}{2}, \quad (5)$$

where q denotes a generic scalar quantity and the average operator applies to scalars and vector quantities. By definition, $[\![q]\!]$ is a vector quantity. These definitions can be suitably extended to faces intersecting $\partial\Omega$ accounting for the weak imposition of boundary conditions.

The discrete counterpart of equation (2) for a generic element $K \in \mathcal{T}_h$ then reads:

$$\int_K \phi_h \frac{\partial \mathbf{u}_h}{\partial t}\,\mathrm{d}\mathbf{x} - \int_K \boldsymbol{\nabla}_h\phi_h \cdot \mathbf{F}(\mathbf{u}_h, \boldsymbol{\nabla}_h\mathbf{u}_h)\,\mathrm{d}\mathbf{x} + \int_{\partial K} \phi_h \mathbf{F}(\mathbf{u}_h|_K, \boldsymbol{\nabla}_h\mathbf{u}_h|_K) \cdot \mathbf{n}\,\mathrm{d}\sigma + \int_K \phi_h \mathbf{s}(\mathbf{u}_h, \boldsymbol{\nabla}\mathbf{u}_h)\,\mathrm{d}\mathbf{x} = \mathbf{0}. \quad (6)$$

Due to the discontinuous functional approximation, the flux function in the third term of equation (6) is not uniquely defined. Moreover, a consistent and stable discretization of viscous fluxes must account for the effect of interface discontinuities on the gradient $\boldsymbol{\nabla}_h\mathbf{u}_h$. Uniqueness of interface fluxes is achieved by introducing a suitably defined numerical flux which couple the solution in adjacent elements and ensure local conservation.

The inviscid $\widehat{\mathbf{F}}_c$ and viscous $\widehat{\mathbf{F}}_v$ parts of the numerical flux are treated independently. For the former we have used the Godunov flux (alternatively, our code can use the van Leer vector split flux as modified by Hänel [3]), while for the viscous flux discretization we have adopted the BR2 scheme presented in[4, 5] and theoretically analyzed in [2, 6] (where it is referred to as BRMPS). The numerical viscous flux is given by:

$$\widehat{\mathbf{F}}_v\left(\mathbf{u}_h^{\pm}, (\boldsymbol{\nabla}_h \mathbf{u}_h + \eta_e \mathbf{r}_e(\llbracket \mathbf{u}_h \rrbracket))^{\pm}\right) \stackrel{\text{def}}{=} \{\mathbf{F}_v\left(\mathbf{u}_h, \boldsymbol{\nabla}_h \mathbf{u}_h + \eta_e \mathbf{r}_e(\llbracket \mathbf{u}_h \rrbracket)\right)\}, \tag{7}$$

where, according to [2, 6], the penalty factor η_e must be greater than the number of faces of the elements. A very interesting feature of the outlined viscous flux discretization scheme is that it couples only the unknowns already coupled by the inviscid flux discretization scheme, irrespective of the degree of polynomial approximation of the solution. This feature is obviously very attractive for an implicit implementation of the method.

Summing equation (6) over the elements and accounting for the above considerations, we obtain the DG formulation of problem (2), which then requires to find $u_{h_1}, \ldots, u_{h_M} \in \Phi_h$ such that:

$$\begin{aligned}\int_{\Omega_h} \phi_h \frac{\partial \mathbf{u}_h}{\partial t}\, \mathrm{d}\mathbf{x} - \int_{\Omega_h} \boldsymbol{\nabla}_h \phi_h \cdot \left(\mathbf{F}_c\left(\mathbf{u}_h\right) + \mathbf{F}_v\left(\mathbf{u}_h, \boldsymbol{\nabla}_h \mathbf{u}_h + \mathbf{r}\left(\llbracket \mathbf{u}_h \rrbracket\right)\right)\right)\, \mathrm{d}\mathbf{x} \\ + \int_{\Gamma_h} \llbracket \phi_h \rrbracket \cdot \left(\widehat{\mathbf{F}}_c\left(\mathbf{u}_h^{\pm}\right) + \widehat{\mathbf{F}}_v\left(\mathbf{u}_h^{\pm}, (\boldsymbol{\nabla}_h \mathbf{u}_h + \eta_e \mathbf{r}_e(\llbracket \mathbf{u}_h \rrbracket))^{\pm}\right)\right)\, \mathrm{d}\sigma \\ + \int_{\Omega_h} \phi_h \mathbf{s}\left(\mathbf{u}_h, \boldsymbol{\nabla}_h \mathbf{u}_h + \mathbf{r}(\llbracket \mathbf{u}_h \rrbracket)\right)\, \mathrm{d}\mathbf{x} = \mathbf{0},\end{aligned} \tag{8}$$

for all $\phi_h \in \Phi_h$. The lifting operator $\mathbf{r}_e$, which is assumed to act on the jumps of $\mathbf{u}_h$ componentwise, is defined as the solution of the following problem:

$$\int_{\Omega_h} \boldsymbol{\phi}_h \cdot \mathbf{r}_e\left(\mathbf{v}\right)\, \mathrm{d}\mathbf{x} = -\int_e \{\boldsymbol{\phi}_h\} \cdot \mathbf{v}\, \mathrm{d}\sigma, \forall \boldsymbol{\phi}_h \in \left[\Phi_h\right]^N,\ \mathbf{v} \in \left[L^1\left(e\right)\right]^N, \tag{9}$$

and the function $\mathbf{r}$ is related to $\mathbf{r}_e$ by the equation:

$$\mathbf{r}(\mathbf{v}) \stackrel{\text{def}}{=} \sum_{e \in \mathcal{E}_h} \mathbf{r}_e(\mathbf{v}). \tag{10}$$

At the boundary of the domain, the numerical flux function appearing in equation (8) must be consistent with the boundary conditions of the problem. In practice, this is accomplished by properly defining a boundary state which accounts for the boundary data and, together with the internal state, allows to compute the numerical fluxes and the lifting operator on the portion Γ_h^{∂} of the boundary Γ_h, see [7, 4].

2.1 Quadrature rules

All integrals appearing in equation (8) are computed by means of Gauss integration rules with a number of integration points suited for the required accuracy. Note that the exact integration must account for the degrees of (i) polynomial approximation, (ii) mapping, and (iii) Jacobian of the mapping. In this way the computational cost of numerical integration can be quite high. On the other hand, numerical experiments indicate that for curved elements reduced quadrature rules are perfectly suited to obtain accurate results, at least if the element aspect ratio is not extremely high. The reduced quadrature rules simple assume that the degree of the polynomial to be integrated on the reference element is equal to that of the polynomial defined in the physical space.

2.2 Time discretization and linear system solution

The discrete problem corresponding to equation (8) can be written as:

$$\mathbf{M}\frac{\mathrm{d}\mathbf{U}}{\mathrm{d}t} + \mathbf{R}\left(\mathbf{U}\right) = \mathbf{0}, \tag{11}$$

where $\mathbf{U}$ is the global vector of unknown degrees of freedom and $\mathbf{M}$ is the global block diagonal mass matrix. Equation (11) defines a system of (nonlinear) ODEs that in many applications requires implicit schemes in order to be solved efficiently. If a linearized backward Euler scheme is adopted the following equation is obtained:

$$\left[\frac{\mathbf{M}}{\Delta t} + \frac{\partial \mathbf{R}\left(\mathbf{U}^n\right)}{\partial \mathbf{U}}\right] \Delta\mathbf{U} = -\mathbf{R}\left(\mathbf{U}^n\right). \tag{12}$$

The Jacobian matrix $\frac{\partial \mathbf{R}(\mathbf{U}^n)}{\partial \mathbf{U}}$ can be regarded as an $N_K \times N_K$ block sparse matrix where N_K is the number of elements in $\mathcal{T}_h$ and the rank of each block is $M \times N_{DOF}^K$, being N_{DOF}^K the number of degrees of freedom for each variable in the generic element K. Thanks to the DG discretization here adopted the degrees of freedom of a generic element K are only coupled with those of the neighbouring elements and the number of nonzero blocks for each (block) row K of the Jacobian is therefore equal to the number of elements surrounding the element K plus one. Solving the equation (12) amounts to solve a linear system of the form $\mathbf{Ax} = \mathbf{b}$ at each time step. Note that in the limit $\Delta t \to \infty$ the linearized backward Euler scheme is in fact identical to one step of the Newton method (quadratic convergence in the computation of steady state solutions).

The linear algebraic system $\mathbf{Ax} = \mathbf{b}$ is solved with either direct multifrontal and iterative preconditioned GMRES methods (Generalized Minimal RESidual). In the latter case both block diagonal and ILU factorization preconditioners have been used. For real life applications the standard GMRE-S approach can be prohibitively expensive as it requires the full storage of the matrix $\mathbf{A}$. However, the huge memory requirement can be reduced by observing that the GMRES algorithm only requires the evaluation of matrix-vectors products $\mathbf{Ax}$, and that these can be approximated by the following matrix-free formula

$$\left[\frac{\mathbf{M}}{\Delta t} + \frac{\partial \mathbf{R}\left(\mathbf{U}^n\right)}{\partial \mathbf{U}}\right] \Delta\mathbf{U} \simeq \frac{\mathbf{M}}{\Delta t}\Delta\mathbf{U} + \frac{\mathbf{R}(\mathbf{U}^n + h\Delta\mathbf{U}) - \mathbf{R}(\mathbf{U}^n)}{h}, \tag{13}$$

where the Jacobian matrix has been replaced by its first order finite difference approximation (see [8] for a comprehensive review of the method) and the parameter h can be computed as proposed by Brown and Saad [9].

For the purpose of computational efficiency equation (13) must be preconditioned by means of a matrix $\mathbf{P}$ that should be a good approximation of $\mathbf{A}^{-1}$. For sequential application an effective preconditioning is achieved using the incomplete lower-upper factorization, ILU(0), of the analytically computed matrix $\mathbf{A}$. Unlike the matrix-based GMRES algorithm the matrix free approach enjoys more flexibility because it allows to freeze the evaluation of the preconditioner matrix for a number of steps without affecting the quadratic convergence of the Newton algorithm. Numerical investigations have shown that this approach gives a clear computational advantage with respects to the standard GMRES, ILU(0) matrix-based, iterative scheme.

For steady computations we have implemented the pseudo-transient continuation technique whereby the CFL number is related to the L_2 and L_∞ norms of the residuals according the following relations:

$$\begin{aligned}\mathrm{CFL} &= \min\left(\frac{\mathrm{CFL}_{\min}}{r^{\beta}}, \mathrm{CFL}_{\max}\right) \\ r &= \max\left(\frac{\|\mathbf{R}\left(\mathbf{U}^{n}\right)\|_{L_2}}{\|\mathbf{R}\left(\mathbf{U}^{0}\right)\|_{L_2}}, \frac{\|\mathbf{R}\left(\mathbf{U}^{n}\right)\|_{L_\infty}}{\|\mathbf{R}\left(\mathbf{U}^{0}\right)\|_{L_\infty}}\right),\end{aligned} \qquad (14)$$

where $\mathrm{CFL}_{\min}$, $\mathrm{CFL}_{\max}$ and β are user-defined input parameters.

The linear algebra is handled through the PETSc[10] library (Portable Extensible Toolkit for Scientific Computations) and the parallelization is based on grid partitioning accomplished by means of the METIS[11] package. Each processor owns the data related to its local portion of the grid and the data on remote processors are accessed through MPI, the standard for message-passing communication. Thanks to the compactness of our DG method only the data owned by the boundary elements of adjacent partitions need to be shared among different processes. Note that since the global ILU(0) is not a highly scalable algorithm for parallel computations the block Jacobi (BJ) method, with one block per process each of which is solved with ILU(0), is used in the parallel context. Several computations have shown that the additive Schwarz method (ASM), for which the local block is enlarged with the matrix entries corresponding to one strip of overlapping elements, is more effective than the BJ method.

3 NUMERICAL RESULTS

In this section we present high-order DG results for 2-dimensional, steady, subsonic problems proposed within the ADIGMA (Adaptive Higher-Order Variational Methods for Aerospace Applications) European project. The first test case considers the viscous flow around the classical NACA0012 airfoil while the second computes the turbulent flow around the L1T2 3-element airfoil.

3.1 Viscous test case

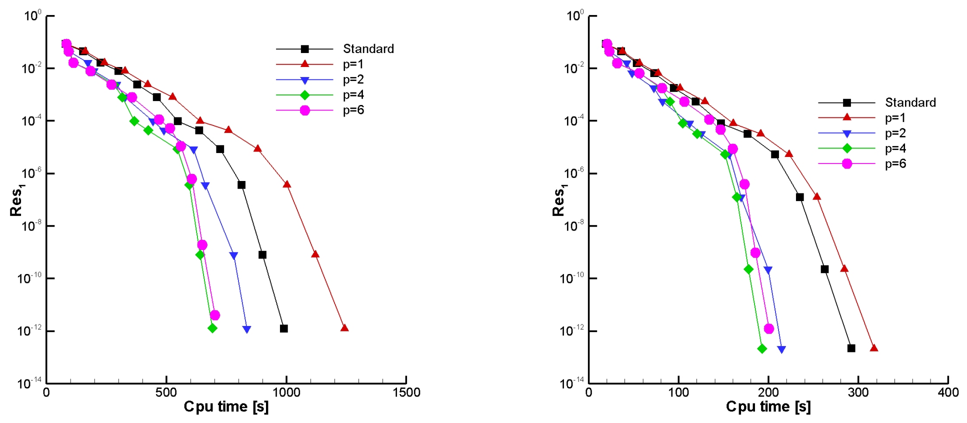
In this test case the compressible viscous flow around a Naca 0012 airfoil at $\mathrm{M}_\infty = 0.5$, $\mathrm{Re} = 500$ and $\alpha = 2^\circ$ has been considered. The investigation of this airfoil aims at assessing the performance of the matrix-free approach with respect to the standard matrix-based algorithm, and at evaluating the efficiency of freezing the analytically computed Jacobian matrix for p Newton steps (Jacobian lagging). This test case has been computed by using $\mathbb{P}_1, \ldots, \mathbb{P}_6$ elements, with both the full and the reduced quadrature techniques, on a suite of 5 unstructured grids, with a number of elements between $ne = 578$ and $ne = 8946$. In all these grids a piecewise cubic representation of the airfoil profile has been adopted.

The results on the first and second grid level were obtained with a sequential computation and a ILU(0) preconditioner, while for finer grids parallel computations were performed using a BJ preconditioner. For all the computations $\mathrm{CFL}_{\min} = 1$, $\mathrm{CFL}_{\max} = 10^{20}$ and $\beta = 1$ were set in equation (14). As of the linear solver a relative residual convergence equal to 10^{-5} was adopted and both the number of GMRES iterations and the dimension of the Krylov subspace were set to 120. Note that the residual tolerance is usually satisfied only for low order polynomial and/or for coarse grids while for the high order case it is reached only during the first Newton iterations, when the CFL number is small and therefore the matrix is less ill-conditioned. Indeed the method can be considered an inexact Newton algorithm.

The memory saving achieved for different polynomial approximation on a fixed computational grid ($ne = 2113$) are reported in Table 1. Note that the different memory requirements for the full and reduced quadrature rules implementations is related to the storage of the shape functions as well as the shape function derivatives at each Gauss quadrature point of each element. Indeed, the choice of shape functions defined in the physical space usually requires a pre-processing phase to compute and store the basis function values once and for all. The comparison between the matrix-free method and the standard method for the full quadrature implementation indicates a relative saving in storage that is almost independent of the grid size while it increases rising the polynomial approximations, from about 13% for $\mathbb{P}_1$ to 29% for $\mathbb{P}_6$. When the reduced quadrature technique is adopted the results are even better for the matrix-free algorithm, being the memory saving in the range $14 \div 37\%$.

Table 1: Memory requirement for each MPI thread and relative saving for turbulent test case, $ne = 4740$.

	Full Quadrature			Reduced Quadrature		
	matrix based [Mb]	matrix free [Mb]	saving [%]	matrix based [Mb]	matrix free [Mb]	saving [%]
$\mathbb{P}_1$	77.9	67.3	13.6	73.3	62.7	14.5
$\mathbb{P}_2$	233.0	187.3	19.6	189.0	143.1	24.3
$\mathbb{P}_3$	545.8	416.2	23.8	430.0	300.0	30.2
$\mathbb{P}_4$	1183.8	891.8	24.8	887.8	596.0	32.9
$\mathbb{P}_5$	2245.0	1669.8	25.6	1625.8	1050.4	35.4
$\mathbb{P}_6$	3550.2	2522.4	28.9	2771.8	1744.2	37.1



(a) Standard quadrature.

(b) Reduced quadrature.

Figure 1: Convergence history, effect of Jacobian lagging with p freezing steps, $ne = 578$.

Analysing the convergence histories for the $\mathbb{P}_6$ computations, see for example Figures 1, it is possible to notice a clear saving of CPU time freezing the preconditioner for a few Newton iterations. In this test case, considering the optimal value $p = 4$, the CPU time is almost the same of the matrix based algorithm in the worst case, while an improvement of 30% is achieved in the best case. Rising further the value of p is not advisable because the number of Newton iterations needed for convergence up to machine precision increases too much. Once more the reduced quadrature technique is more effective for the matrix-free approach spotting a CPU time reduction of $24 \div 34\%$ with $p = 4$.

3.2 Turbulent test case

The reliability and robustness of the method have been tested by computing the turbulent flow around the L1T2 3-element airfoil at $\mathrm{M}_\infty = 0.197$, $\mathrm{Re} = 3.52 \cdot 10^6$, $\alpha = 20.16°$. This problem is characterized by low free-stream velocity and high incidence which develops very steep gradients of velocity and turbulence around the slat. The interaction of different turbulent shear layers further complicates the flow development. This computation turns out to be very challenging due to the numerical stiffness related to the high order discretization of the coupled RANS and $k - \omega$ turbulent model equations and to the highly stretched and curved elements of the grid. This test case has been computed up to $\mathbb{P}_6$ polynomial approximation on a grid consisting of $4{,}740$ quadrangular elements each of them obtained by agglomerating 16 elements of a multi-block structured finer grid. The position of the finer grid points has been used to define a higher-order representation of the geometry of the elements, which in fact display curved edges in Figures 2(a) and 2(b). Due to the grid characteristics only the full quadrature approach has been considered in these computations. All the solution have been computed using 4 ($\mathbb{P}_1$, $\mathbb{P}_2$ and $\mathbb{P}_3$) and 8 processes ($\mathbb{P}_4$, $\mathbb{P}_5$ and $\mathbb{P}_6$) on a small Linux Cluster with 4 Opteron based nodes for a total of 32 cores operating at 2.3 GHz.

Figure 3(a) shows the residuals convergence history (note that for computational efficiency higher-order solutions are started from the lower-order ones) using the BJ preconditioner. The method proved to be suitable and robust even for this test case but, unlike the laminar case, a small adjustment of the pseudo-time continuation law parameters has been adopted. In particular the β value in equation (14) has been slightly decreased rising the polynomial degree, from $\beta = 1$ for $\mathbb{P}_1$ to $\beta = 0.7$ for $\mathbb{P}_6$. In all the cases but the $\mathbb{P}_1$, for which a $\mathrm{CFL}_{\min} = 3$ is used, we have set $\mathrm{CFL}_{\min} = 1$, $\mathrm{CFL}_{\max} = 10^{20}$, relative tolerance and maximum iterations for the restarted GMRES(60) equal to 10^{-5} and 120 respectively.

The practice of Jacobian lagging for the parallel computations of this test case reveals a high sensitivity of the Newton method to the type of preconditioning. For BJ preconditioning the technique has been effective only up to $\mathbb{P}_4$ approximation while for higher order approximations the computation sometimes blew-up. On the other hand the Jacobian lagging was found to work fine if coupled to an ASM preconditioner even allowing to reduce the maximum number of iterations to 60. Thanks to the better ASM preconditioning properties a significant improvement of the overall execution time has been observed, see Figure 3(b) where the Jacobian is computed every two steps. It is worth noting that the current PETSc implementation of ASM precontioning is not optimal because it keeps in memory both the full Jacobian matrix and its ASM preconditioned counter part thus limiting the memory saving that in principle could be obtained.

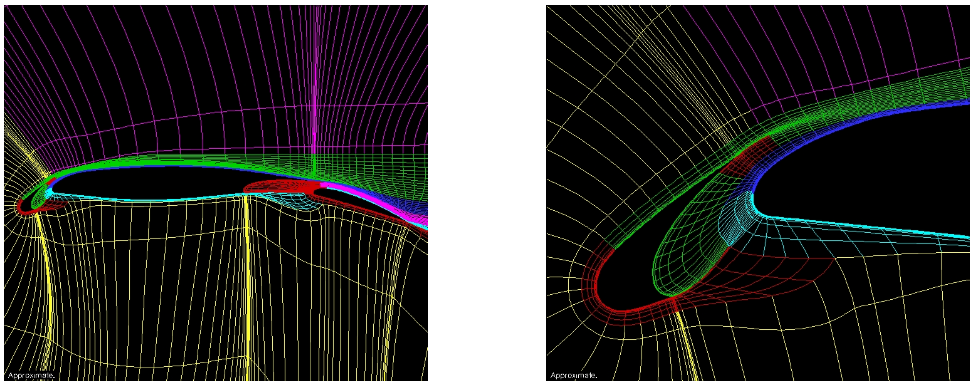
Comparing Tables 1 and 2 one can appreciate that the memory saving of the matrix-free method with BJ preconditioner for the RANS equations is greater if compared to a NS laminar computation since the number of equations and unknowns of the PDE system increases from four to six. Concerning the CPU time the performance of the standard and of the matrix-free algorithm with a $p = 2$ Jacobian lagging are almost identical.

Figure 4 summarizes the DG results for this test case. Rising the degree of the polynomial approximation greatly improves the pressure distribution on the most critical part of the airfoil, *i.e.* on the slat and on the front part of the main profile.

The computational experience on the turbulent test case suggests that achieving an optimal efficiency strongly depends on a subtle balance between robustness and performance whereby the pseudo-transient continuation technique to determine the CFL number plays a crucial role that needs to be further investigated.

Table 2: Memory requirement for each MPI thread and relative saving for turbulent test case, $ne = 4740$.

	matrix based [Mb]	matrix free [Mb]	saving [%]
$\mathbb{P}_1$	63.4	49.9	21.6
$\mathbb{P}_2$	195.7	139.1	28.9
$\mathbb{P}_3$	481.9	322.2	33.1
$\mathbb{P}_4$	535.3	356.5	33.3
$\mathbb{P}_5$	1027.4	675.5	34.3
$\mathbb{P}_6$	1701.9	1074.7	36.9



(a) grid around the L1T2 airfoil

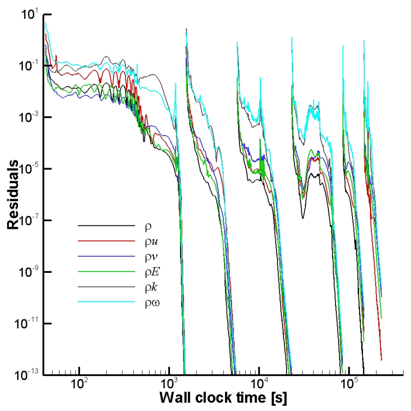
(b) grid around the slat

Figure 2: Computational grid, L1T2 3-element airfoil.

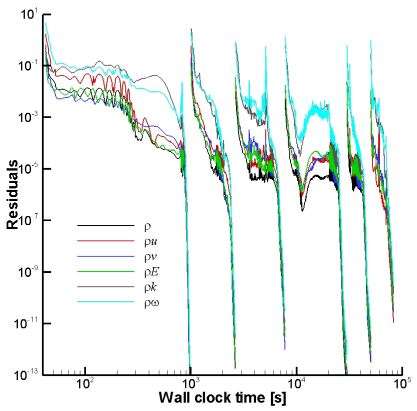
4 CONCLUSION

A matrix-free Newton-Krylov algorithm has been implemented in a high-order parallel DG code for computational fluid dynamics, the proposed approach has been validated and the computational efficiency verified. The method yields a significant memory saving with respect to the standard matrix-based approach, while the lagging of the Jacobian matrix, used only as preconditioner, can improve the performance of the code, *i.e.*, the execution time. A complex turbulent flow problem has been used to assess the method for real life simulations.

Further developments will include its implementation in a 3D code and the investigation of different types of preconditioners, using approximated and cheaper forms for the Jacobian matrix.



(a) BJ preconditioner, 2310 Newton iterations.



(b) ASM preconditioner, 1570 Newton iterations.

Figure 3: Convergence history, L1T2 3-element airfoil, $\mathrm{M}_\infty = 0.197$, $\alpha = 20.16°$, $\mathrm{Re} = 3.52\times10^6$.

ACKNOWLEDGEMENT

The first and the third authors acknowledge the financial support of the European Union, under the ADIGMA project.

References

[1] F. Bassi, A. Crivellini, S. Rebay, and M. Savini. Discontinuous Galerkin solution of the Reynolds-averaged Navier-Stokes and k-ω turbulence model equations. *Comput. & Fluids*, 34:507–540, 2005.

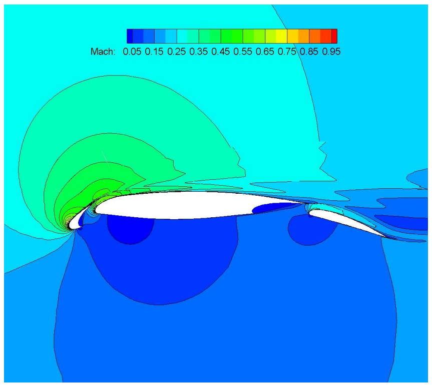
[2] F. Brezzi, G. Manzini, D. Marini, P. Pietra, and A. Russo. Discontinuous Galerkin approximations for elliptic problems. *Numer. Methods Partial Differential Equations*, 16:365–378, 2000.

[3] D. Hänel, R. Schwane, and G. Seider. On the accuracy of upwind schemes for the solution of the Navier–Stokes equations. AIAA Paper 87-1105 CP, AIAA, July 1987. Proceedings of the AIAA 8th Computational Fluid Dynamics Conference.

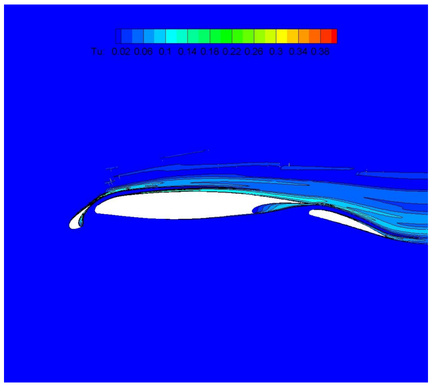
[4] F. Bassi, S. Rebay, G. Mariotti, S. Pedinotti, and M. Savini. A high-order accurate discontinuous finite element method for inviscid and viscous turbomachinery flows. In R. Decuypere and G. Dibelius, editors, *2nd European Conference on Turbomachinery Fluid Dynamics and Thermodynamics*, pages 99–108, Antwerpen, Belgium, March 5–7 1997. Technologisch Instituut.

[5] F. Bassi and S. Rebay. A high order discontinuous Galerkin method for compressible turbulent flows. In *Discontinuous Galerkin Methods. Theory, Computation and Applications*, volume 11 of *Lecture Notes in Computational Science and Engeneering*. Springer-Verlag, 2000. *First Internation Symposium on Discontinuous Galerkin Methods,* May 24–26, 1999, Newport, RI, USA.

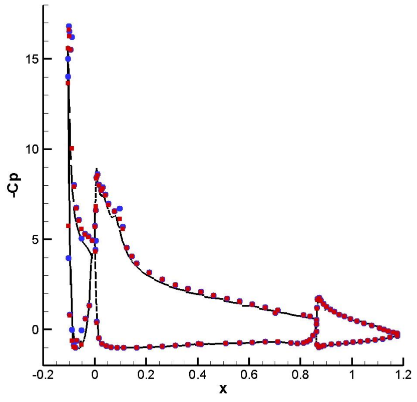
[6] D. N. Arnold, F. Brezzi, B. Cockburn, and D. Marini. Unified analysis of discontinuous Galerkin methods for elliptic problems. *SIAM J. Numer. Anal.*, 39(5):1749–1779, 2002.



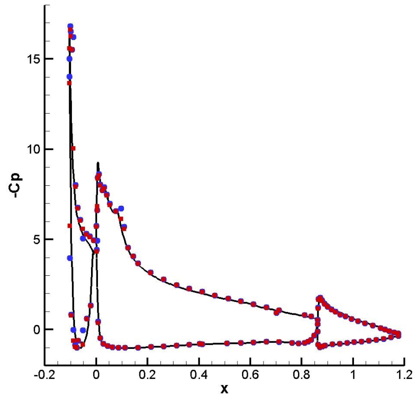
(a) Mach contours, $\mathbb{P}_5$ solution



(b) turbulence intensity contours, $\mathbb{P}_5$ solution



(c) pressure coefficient, $\mathbb{P}_2$ solution



(d) pressure coefficient, $\mathbb{P}_5$ solution

Figure 4: L1T2 3-element airfoil, $\mathrm{M}_\infty = 0.197$, $\alpha = 20.16°$, $\mathrm{Re} = 3.52{\times}10^6$.

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

[8] D.A. Knoll and D.E. Keyes. Jacobian-free Newton-Krylov methods: a survey of approaches and applications. *J. Comput. Phys.*, 193(2):357–397, 2004.

[9] P.N. Brown and Y. Saad. Hybrid Krylov methods for nonlinear systems od equations. *SIAM J. Sci. and Stat. Comput.*, 11(3):450–481, 1990.

[10] Satish Balay, Kris Buschelman, William D. Gropp, Dinesh Kaushik, Matthew G. Knepley, Lois Curfman McInnes, Barry F. Smith, and Hong Zhang. PETSc Web page, 2001. http://www.mcs.anl.gov/petsc.

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