AIAA AVIATION Forum
17-21 June 2019, Dallas, Texas
AIAA Aviation 2019 Forum

10.2514/6.2019-3418

An entropy-adjoint p-adaptive discontinuous Galerkin method for the under-resolved simulation of turbulent flows

Francesco Bassi[1*], Alessandro Colombo[1†], Andrea Crivellini[2‡], Krzysztof J. Fidkowski[3§], Matteo Franciolini[3,4¶], Antonio Ghidoni[5‖], and Gianmaria Noventa[5**]

[1] *University of Bergamo, Dalmine (BG), 24044, Italy*

[2] *Polytechnic University of Marche, Ancona, 60131, Italy*

[3] *University of Michigan, Ann Arbor (MI), 48109, United States*

[4] *NASA Ames Research Center, Mountain View (CA), 94035, United States*

[5] *University of Brescia, Brescia, 25123, Italy*

The paper presents an approach to mesh adaptation suitable for scale resolving simulations. The methodology is based on the entropy adjoint approach, which corresponds to a standard output-based adjoint method with output functional targeting areas of spurious generation of entropy. The method shows several advantages over standard output-based error estimation: i) it is computationally inexpensive, ii) does not require the solution of a fine-space adjoint problem, and iii) is nonlinearly stable with respect to the primal solution for chaotic dynamical systems. In addition, the work reports on the parallel efficiency of the solver, which has been optimized through a multi-constraint domain decomposition algorithm available within the Metis 5.0 library.[1] The reliability, accuracy, and efficiency of the approach are assessed by computing three test cases: the two-dimensional, laminar, chaotic flow around a square at $Re = 3\,000$, the implicit Large Eddy Simulation (LES) of a circular cylinder at $Re = 3\,900$, and the ILES of a square cylinder at $Re = 22\,000$. The results show significant reduction in the number of DoFs with respect to uniform order-refinement, and good agreement with experimental data.

I. Introduction

The growing need for high-fidelity flow simulations and accurate problem-specific output quantities has paved the way for higher-order methods such as the discontinuous Galerkin (DG) method. Although for many problems, with typical engineering error tolerances, DG methods are still generally less efficient than standard industrial codes, the industrial interest is strongly fostering the scientific community to devise more efficient high-order CFD solvers. These could then reduce the computational time for high-accuracy solutions, enabling higher-fidelity simulations at acceptable costs.

Previous works contributed to the development of efficient high-order numerical methods for steady and unsteady flow problems involving adaptation of the space discretization by varying the order of polynomial approximation throughout the domain[2,3] or by performing mesh adaptation.[4–7] In this paper a p-adaptation strategy has been adopted to increase the computational efficiency of a DG solver[8] for scale-resolving simulations. The proposed strategy allows changing the polynomial order of the solution representation within each element according to an error estimate, reducing substantially the CPU time and memory usage, while not

*Full Professor, Department of Engineering and Applied Sciences, francesco.bassi@unibg.it
†Assistant Professor, Department of Engineering and Applied Sciences, alessandro.colombo@unibg.it
‡Associate Professor, Department of Industrial Engineering and Mathematical Sciences, a.crivellini@staff.univpm.it
§Associate Professor, Department of Aerospace Engineering, kfid@umich.edu
¶USRA Fellow, matteo.franciolini@nasa.gov
‖Associate Professor, Department of Mechanical and Industrial Engineering, antonio.ghidoni@unibs.it
**Research Fellow, Department of Mechanical and Industrial Engineering, gianmaria.noventa@unibs.it

spoiling the spectral resolution required by this class of simulations. In particular, as the work is focused on unsteady scale-resolving simulations, such as Large Eddy Simulation (LES), an adaptation procedure suitable for time-dependent problems is investigated. Previous research efforts[9] aimed at adapting the polynomial degree by combining two simple element-wise indicators based on interface pressure jumps and on the decay of the coefficients of the modal expansion. These sensors are coupled to guarantee a reasonable behavior both for high- and low-degree polynomial approximations. Despite not targeting any output quantity, the strategy showed satisfactory performance involving the solution of statistically steady turbulent flows.

The objective of the present work is to extend the previous approach by using error estimates provided by the so called entropy-adjoint approach introduced by Fidkowski and Roe.[10] The main idea of the method is to use the set of entropy variables associated with the state as the solution of the adjoint problem for an output functional that measures the balance, including the inflow, outflow, and generation, of entropy throughout the domain. By targeting the source of spurious entropy generation, which is closely related to numerical error, the approach overcomes the drawback of many heuristic sensors, which are generally not robust for controlling numerical discretization errors. The approach is demonstrated to be able to target areas of spurious entropy generation, which mainly affect the scalar output that measures net entropy balance throughout the domain. The method demonstrated several advantages over standard, output-based adaptive simulations for steady flows,[11,12] since the adjoint solution is easily obtained by manipulating the state vector. This property can be exploited also for unsteady flow problems, avoiding a backward-in-time integration, typical of unsteady adjoint problems, which is considerably expensive for three-dimensional simulations. Moreover, the adjoint solution is nonlinearly stable, meaning that the accuracy of the error estimate is not spoiled by the chaotic nature of the flow.

Output-based error estimation typically involves the computation of a fine-space adjoint solution[7,13–15] which can be accomplished solving a fine space problem (few smoothing iterations are typically involved), or reconstructing the solution. The latter approach has been considered and in this work we explore the use of a patch reconstruction scheme. The method aims at approximating the fine space adjoint on a $k+1$ space by interpolating solutions of polynomial orders k on an extended stencil. The reconstruction process takes advantage of the hierarchical and orthonormal basis functions.[16] Considering the reconstruction of the higher-order solution space ϕ_K on a given element K, the procedure relies on projecting solutions from the neighboring elements sharing a face with K to the extension of ϕ_K over the whole patch. The conservation of the mean value is also enforced during the reconstruction.

Part of the numerical experiments reported herein are devoted to show the parallel efficiency on multi-core machines, which is obtained through a dynamic load balancing based on the Metis library and its capability of generating weighted graphs. Such an implementation handles imbalances in the degrees of freedom on each partition due to the application of the adaptation algorithm, and provides an optimal scalability, comparable to that obtained using a fixed number of degrees of freedom.

The reliability, accuracy and efficiency of the approach are assessed by computing two test cases involving compressible, low Mach number, unsteady flows. First, two-dimensional, laminar chaotic flow around a square at $Re = 1000$ is studied. Second, the implicit LES of two span-wise periodic flows is studied: the flow around the span-wise periodic circular cylinder at $Re = 3\,900$, and around a square cylinder at $Re = 22000$. The results show significant reduction in the number of DoFs with respect to uniform order-refinement, and good agreement with experimental data.

II. The numerical framework

A. Space and time discretization

The governing equations are discretized in space according to the DG method.[8] The Navier-Stokes equations for m variables in d dimensions can be written in compact form as

$$\mathbf{P}\left(\mathbf{w}\right)\frac{\partial \mathbf{w}}{\partial t} + \nabla \cdot \mathbf{F}_{\mathrm{c}}\left(\mathbf{w}\right) + \nabla \cdot \mathbf{F}_{\mathrm{v}}\left(\mathbf{w}, \nabla \mathbf{w}\right) = \mathbf{0}, \tag{1}$$

where $\mathbf{w} \in \mathbb{R}^m$ is the unknown solution vector, $\mathbf{F}_c, \mathbf{F}_v \in \mathbb{R}^m \otimes \mathbb{R}^d$ are the convective and viscous flux functions, and $\mathbf{P}\left(\mathbf{w}\right) \in \mathbb{R}^m \otimes \mathbb{R}^m$ is a transformation matrix that takes into account the possible use of a set of unknowns $\mathbf{w}$ different frp, the conservative set $\mathbf{w}_c = [\rho, \rho E, \rho u_i]^T$. In particular, in this work, we use both the conservative and the primitive set of variables, the latter given by $\mathbf{w} = [p, T, u_i]^T$.

For the spatial discretization, the weak form of the governing equations is first obtained by multiplying Eq. (1) by an arbitrary, smooth test function and integrating by parts. The solution and the test function are then replaced with a finite-element approximation and a discrete test function, both belonging to the finite-dimensional set $\mathbf{V}_h := [\mathbb{P}_d^k(\mathcal{T}_h)]^m$, where $\mathbb{P}_d^k(\mathcal{T}_h) := \{v_h \in L^2(\Omega) \,|\, v_{h|K} \in \mathbb{P}_d^k(K), \forall K \in \mathcal{T}_h\}$ is the discrete polynomial space in physical (mesh) coordinates. $\mathbb{P}_d^k(K)$ denotes the restriction of the polynomial functions of $d = 2, 3$ variables and total degree k to element K belonging to a non-overlapping triangulation $\mathcal{T}_h = \{K\}$ of the computational domain. A set of hierarchical and orthonormal basis functions for the space $\mathbb{P}_d^k(K)$ is computed according to Bassi *et al.*[16] As the functional approximation space is discontinuous, the flux functions over mesh faces are not uniquely defined. The convective and viscous flux functions are then replaced with numerical counterparts according to the exact Riemann solver of Gottlieb and Groth[17] and the BR2 scheme of Bassi and Rebay,[18] respectively. In two dimensions, the approximate Riemann solver of Roe is used.[19]

By assembling together all the elemental contributions of the DG discretization, the system of ordinary differential equations (ODEs) governing the evolution in time of the discrete solution can be written as

$$\frac{d\mathbf{W}}{dt} + \widetilde{\mathbf{R}}(\mathbf{W}) = \mathbf{0}, \quad \text{with} \quad \widetilde{\mathbf{R}}(\mathbf{W}) = \mathbf{M_P}^{-1}(\mathbf{W})\,\mathbf{R}(\mathbf{W}), \tag{2}$$

where $\mathbf{W}$ is the global vector of unknown degrees of freedom, $\mathbf{M_P}$ is the global block diagonal mass matrix, and $\mathbf{R}(\mathbf{W})$ is the vector of spatial residuals. For sets of variables different than $\mathbf{w}_c$, the matrix $\mathbf{P}$ couples the degrees of freedom of the variables within each block of $\mathbf{M_P}$, so that this matrix is not diagonal, even using a set of orthogonal basis functions. Accurate high-order time integration is performed by means of multi-stage, linearly implicit, Rosenbrock-type, Runge-Kutta schemes. Such schemes require the solution of a linear system per stage, while the Jacobian matrix needs to be assembled only once per time step:

$$\mathbf{W}^{n+1} = \mathbf{W}^n + \sum_{j=1}^{s} m_j \mathbf{Y}_j, \tag{3}$$

$$\left(\frac{\mathbf{I}}{\gamma \Delta t} + \widetilde{\mathbf{J}}\right)^n \mathbf{Y}_i = -\widetilde{\mathbf{R}}\left(\mathbf{W}^n + \sum_{j=1}^{i-1} a_{ij}\mathbf{Y}_j\right) + \sum_{j=1}^{i-1} \frac{c_{ij}}{\Delta t}\mathbf{Y}_j, \quad i = 1, \ldots, s, \tag{4}$$

where, omitting the dependence on $\mathbf{W}$ for the sake of notation compactness,

$$\mathbf{J} = \frac{\partial \mathbf{R}}{\partial \mathbf{W}}, \qquad \widetilde{\mathbf{R}} = \mathbf{M_P}^{-1}\mathbf{R}, \qquad \widetilde{\mathbf{J}} = \frac{\partial \widetilde{\mathbf{R}}}{\partial \mathbf{W}} = \mathbf{M_P}^{-1}\left(\mathbf{J} - \frac{\partial \mathbf{M_P}}{\partial \mathbf{W}}\widetilde{\mathbf{R}}\right),$$

and m_i, a_{ij}, c_{ij} are real coefficients. Among the variety of Rosenbrock schemes available in the literature, the three-stage, third-order ROS3P scheme of Lang and Verwer[20] is used. An extended review of Rosenbrock schemes is reported by Bassi *et al.*[21] In our implementation, the Jacobian matrix $\mathbf{J}$ is computed analytically and the preconditioned GMRES algorithm[22] is used to solve Eq. (3) at each time step.

B. Evaluation of the basis functions in physical space

The three-dimensional DG solver relies on orthonormal and hierarchical basis functions defined in the physical space. This particular choice improves approximation properties over basis functions defined in reference space, and increases the flexibility of the discretization when dealing with agglomerated elements. However, an efficient evaluation of basis functions to compute the integrals is mandatory when dealing with adaptive, unsteady, memory-intensive computations. In particular, when relying on a set of orthonormal and hierarchical basis functions,[16] several options can be considered for an efficient implementation. In practice, this basis is built from a set of monomials defined in a reference frame relocated in the element barycenter and aligned with the principal axes of inertia, by applying the modified Gram-Schmidt (MGS) orthonormalization procedure. The MGS procedure can be expressed as the application of a set of coefficients, whose definition involves the calculation of some volume integrals, to the monomial basis when evaluated at any quadrature point. As these coefficients may be computed once during a pre-processing stage, at least three possible implementations can be devised (here listed for decreasing memory usage and increasing CPU time):

for each mesh element and face, *i*) the basis functions and their derivatives are evaluated and stored at each quadrature point during pre-processing; *ii*) the coefficients are evaluated and stored during pre-processing, while monomials are evaluated and orthonormalized on-the-fly at any quadrature point during operators assembly; *iii*) both the monomials and the orthonormalization coefficients are computed on-the-fly when needed during operator assembly. According to our numerical experiments, the second approach, *i.e.* computing and storing in memory the coefficients during pre-processing and performing the orthonormalization of the monomials basis on-the-fly at quadrature points during operators assembly, proved to be the best compromise between CPU time and the memory footprint.[9]

C. Efficient adaptive quadrature rules

Another key aspect to improve the solver efficiency is represented by a proper choice of the degree of exactness of the quadrature rules. In fact, the number of integration points rapidly increases when dealing with high-degree polynomial approximations and curved mesh elements. It is worth noticing that the use of curved mesh elements is typically limited to the vicinity of a curved boundary, while in the remaining of the domain straight-sided cells and faces are employed. To avoid over-integration on those elements, it is of primary importance to recognize the minimum quadrature requirements of each element and face. To this end we introduce an algorithm to identify the element curvature within the domain by measuring the element-wise integration error. The algorithm computes, for the diagonal entries of the elemental mass matrix, the relative error between integrating using a quadrature rule with the theoretical degree of exactness, accounting for the non-linear mapping, and a quadrature rule with a reduced degree of exactness. According to this definition and to a user-defined tolerance, a proper set of "reduced" quadrature points is used with significant CPU time savings when dealing with curved geometries for similar accuracy on the integral evaluation.

III. Adaptation strategy

A. Output-adjoint sensitivity

On practical engineering problems an unsteady output quantity can be generally computed as

$$\bar{J} = \int_0^T J(\mathbf{W}(t), t)dt + J_T(\mathbf{W}(T)), \tag{5}$$

where $J(\mathbf{W}(t), t)$, J_T are functionals of the unsteady state vector $\mathbf{W}(t)$, with T the size of the time window and J_T computed at the final time. The continuous-in-time adjoint solution $\mathbf{\Psi}(t)$ is the sensitivity of the output functional $\bar{J}$ with respect to perturbations on the unsteady residual

$$\overline{\mathbf{R}}(\mathbf{W}(t), t) = \mathbf{M_P}\frac{d\mathbf{W}}{dt} + \mathbf{R}\left(\mathbf{W}\right) = \mathbf{0}. \tag{6}$$

An equation for $\mathbf{\Psi}(t)$ can be obtained by imposing the stationarity of the Lagrangian,

$$\mathcal{L} = \bar{J} + \int_0^T \mathbf{\Psi}^T\overline{\mathbf{R}}(\mathbf{W}(t), t)dt = \bar{J} + \int_0^T \mathbf{\Psi}^T\left(\mathbf{M_P}\frac{d\mathbf{W}}{dt} + \mathbf{R}(\mathbf{W})\right)dt, \tag{7}$$

with respect to the permissible state variations $\delta\mathbf{W}$, which reads

$$\delta\mathcal{L} = \frac{dJ_T}{d\mathbf{W}}\delta\mathbf{W}|_{t=T} + \mathbf{\Psi}^T\mathbf{M_P}\delta\mathbf{W}|_{t=T} - \mathbf{\Psi}^T\mathbf{M_P}\delta\mathbf{W}|_{t=0} + \int_0^T\left[\frac{\partial J}{\partial\mathbf{W}} - \frac{d\mathbf{\Psi}^T}{dt}\mathbf{M_P} + \mathbf{\Psi}^T\frac{\partial\mathbf{R}}{\partial\mathbf{W}}\right]\delta\mathbf{W}dt = \mathbf{0}. \tag{8}$$

Note that the third term evaluated at $t = 0$ vanishes, as the primal solution $\delta\mathbf{W}$ is constrained by the initial condition of the problem. By taking the transpose of the integrand, the unsteady adjoint differential equation reads

$$-\mathbf{M_P}\frac{d\mathbf{\Psi}}{dt} + \left(\frac{\partial\mathbf{R}}{\partial\mathbf{W}}\right)^T\mathbf{\Psi} + \left(\frac{\partial J}{\partial\mathbf{W}}\right)^T = \mathbf{0}, \tag{9}$$

with the terminal condition at $t = T$,

$$\mathbf{\Psi}(T) = -\mathbf{M_P}^{-1}\frac{dJ_T^T}{d\mathbf{W}}, \tag{10}$$

which requires to march backwards in time. To this end, it is useful to define $\tau = T - t$ and rewrite the discrete unsteady adjoint equation as

$$\mathbf{M_P}\frac{d\mathbf{\Psi}}{d\tau} + \left(\frac{\partial \mathbf{R}}{\partial \mathbf{W}}\right)^T \mathbf{\Psi} + \left(\frac{\partial J}{\partial \mathbf{W}}\right)^T = \mathbf{M_P}\frac{d\mathbf{\Psi}}{d\tau} + \mathbf{R}^{\mathbf{\Psi}}(\mathbf{W}, \mathbf{\Psi}) = \mathbf{0}. \tag{11}$$

This technique proved to be efficient and reliable for sensitivity analysis of some classes of unsteady problems.[5, 23] However, it has been shown that adjoint approaches compute very large sensitivities when applied to chaotic dynamical systems such as those arising from scale resolving simulations. Such behaviour is not only driven by turbulence itself, but it is also observed in two dimensional flow simulations. The reason for this resides in the very high sensitivity with respect to the initial conditions in the context of chaotic dynamical systems,[24–26] leading to sensitivity parameters and error estimates which are not anymore effective. Additionally, the backward time integration increases considerably the computational costs of the the procedure, which then increases dramatically the overall CPU time of the simulation.

To circumvent those problems, we propose to compute an adjoint by neglecting the unsteady nature of Eq. (9) and following a steady-state approach, being interested in adapting for time-integrated statistically-steady quantities. The idea of using a steady-state method to target adaptation of quasi-stationary governing equations was already employed in the literature, see for example Braak *et al.*[27] The adjoint solution in this case is obtained as the solution of

$$\mathbf{R}^{\mathbf{\Psi}}(\widehat{\mathbf{W}}, \mathbf{\Psi}) = \left(\frac{\widehat{\partial \mathbf{R}}}{\partial \mathbf{W}}\right)^T \mathbf{\Psi} + \left(\frac{\widehat{\partial J}}{\partial \mathbf{W}}\right)^T = \mathbf{0}, \tag{12}$$

where the residual Jacobian and the output linearization are evaluated using a pre-computed time-averaged solution. We remark that this approach reduces the computational cost relative to solving Eq. (9), as the steady-state adjoint is obtained through the solution of a single linear system. However, it does not provide exact sensitivities with respect to the output, since the average field $\widehat{\mathbf{W}}$ is not a solution of the space-time problem, $\overline{\mathbf{R}}(\mathbf{W}(t), t) = \mathbf{0}$.

B. Entropy-adjoint sensitivity

The entropy-adjoint approach can be derived from by choosing an output functional such that

$$J = \int_{\partial\Omega} \mathbf{f}_e \mathbf{n} d\sigma + \int_{\Omega} \nabla \mathbf{v}^T \mathbf{F}_v d\Omega - \int_{\partial\Omega} \mathbf{v}^T \mathbf{F}_v \mathbf{n} d\sigma, \tag{13}$$

where $\mathbf{f}_e$ is the entropy flux associated with an entropy function $U(\mathbf{w}_c)$, and $\mathbf{w}_e$ are the entropy variables associated with U. The main idea behind the approach is to choose entropy variables that symmetrize inviscid and viscous terms of the compressible Navier–Stokes equations. This can be achieved by setting

$$U = -\frac{\rho S}{\gamma - 1}, \qquad S = \ln p - \gamma \ln \rho, \tag{14}$$

where S is the physical entropy and γ is the ratio of specific heats. The entropy variables are then

$$\mathbf{w}_e = U^T_{\mathbf{w}_c} = \left[\frac{\gamma - S}{\gamma - 1} - \frac{1}{2}\frac{\rho u_i u_i}{p}, \frac{\rho u_i}{p}, -\frac{\rho}{p}\right]^T \tag{15}$$

while the corresponding entropy flux is $f_{e,i} = u_i U$.

The three integrals appearing in Eq. (13) can be associated to the flow physics by noting that the first represents the convective outflow of entropy across the domain boundary, the second term is the generation of entropy due to the viscous dissipation, and the third is the entropy diffusion across the boundary. Since U is opposite sign of S, the first term measures the net convective inflow of physical entropy across the boundary, while the third term computes the net diffusive inflow of physical entropy into the domain. Since the second integral evaluates the generation of entropy inside the domain, the theoretical value of the functional J for an exact solution should be zero, meaning that the production of entropy is balanced from the entropy flux throughout the boundaries. Since the imbalance is not strictly verified in a discrete sense, the output functional J targets areas of spurious entropy production by the numerical discretization.

The choice of such output functional and entropy variable is particular attractive. First, it is worth pointing out that under, under the assumption of limited entropy generation over the domain $\Delta S/R \ll 1$, where R is the gas constant coefficient, adapting using the objective function defined in Eq. (13) can be connected to a drag-adaptation, where the objective function is calculated integrating entropy, as proposed by Oswatitsch.[28] As demonstrated by Fidkowski *et al.*,[11] adapting using entropy variables can be considered equivalent to a farfield drag output adjoint for inviscid cases. For viscous cases, the equivalence of the error estimate can be recovered through an additional fine space residual evaluation involving the viscous terms, which is however neglected in the current study.

In fact, Fidkowski and Roe[10] demonstrated that the entropy variables serve as solution of the adjoint problem formulated as in Eq. (8). This means that the solution Ψ is obtained from the state variable directly, and hence at extremely low computational cost. The residual's linearization is not required for its computation, meaning that the adjoint solution remains bounded for chaotic flow problems. Finally, the average adjoint value can be obtained directly from the average state, similarly to what is performed in Eq. (12).

C. Error localization

In the results, we compare entropy-adjoint and output-adjoint adaptive indicators for a two-dimensional problem. These indicators take the following form for element K:

$$\eta_K^{\psi} \equiv \left| \delta \widehat{\mathbf{\Psi}}_{K,h}^T \widehat{\mathbf{R}}_h(\mathbf{W}_h^H) \right|, \tag{16}$$

where $\delta \widehat{\mathbf{\Psi}}_{K,h}^T$ is the steady-state, fine-space entropy or output adjoint solution computed from the coarse-space, time-averaged primal solution injected into the fine-space. The δ indicates that the coarse-space projection of this output or entropy adjoint is removed prior to error estimation. $\widehat{\mathbf{R}}_h(\mathbf{W}_h^H)$ is the time-averaged unsteady residual, computed during the unsteady primal solution by injecting the coarse-space solution into the fine-space and calculating the residual. The residual is then averaged in time, and cancellations between residuals at different times are allowed. The purpose of using an averaged, unsteady residual is to obtain an accurate representation of the extent to which the fine-space unsteady equations are not satisfied using the coarse-space solution. A static sensitivity field computed from the steady-state entropy and output adjoint then provides the weight on this residual. The error indicator drives a fixed-fraction adaptive strategy, and a burn-time is used prior to each unsteady simulation.

Eq. (16) shows that a fine space adjoint solution is necessary to compute $\delta \widehat{\mathbf{\Psi}}_{K,h} = \widehat{\mathbf{\Psi}}_{K,h} - \widehat{\mathbf{\Psi}}_{K,H}$. In general, two methods are commonly used to this end:[13, 29]

- the computation of a fine space adjoint $\widehat{\mathbf{\Psi}}_{K,h}$ on a finer space or using higher order polynomials, $\mathcal{V}_h \in \mathcal{V}_H$, and then subtracting $\widehat{\mathbf{\Psi}}_{K,H}$ obtained through standard Galerkin projection operators;
- the interpolation of the finer space solution through interpolation on a wider patch, also known as patch reconstruction, as $\widehat{\mathbf{\Psi}}_{K,h} = \mathbf{I}_H^h \widehat{\mathbf{\Psi}}_{K_p,h}$, where K_p is the union of the elements sharing a face with K.

In this work the auxiliary problem on a finer solution space is avoided by following the second approach: the global vector of degrees of freedom of entropy variables is computed, and then reconstructed element-by-element on a higher polynomial degree ($k_K + 1, \forall\, K \in \mathcal{T}_h$) using information on the patch.

D. Adaptation algorithm

The adaptation of the elemental polynomial degree is driven by an error estimator that identifies regions of the domain that lack/exceed a required resolution. These regions can be refined/coarsened by increasing/decreasing the degree of the polynomial approximation of the solution. In this work only refinement has been considered. Although performing the adaptation at each time step could be considered as the best choice, this would lead to a significant overhead related to the need of continuously balance the computational load over the processes. For this reason, in this work the error estimators are computed with the time-averaged solution, $\overline{\mathbf{W}}$. Regardless of the adopted error estimator, the pseudo code of the adaptation procedure (including coarsening) is reported in Algorithm 1. Here, $\hat{k}$ is the polynomial degree at the beginning of the

Algorithm 1 Adaptation algorithm

```
1: $l = 0$
2: $k_K = \hat{k}\ \forall K \in \mathcal{T}_h$
3: for $i_{cyc} = 1$ to $N_{cyc}$ do
4:    integrate the governing equation in time
5:    evaluate the time-averaged solution, $\overline{\mathbf{W}}$
6:    if $mod(i_{cyc}, \mathcal{N}) = 0$ and $l \leq n_{adp}$ then
7:       $l \leftarrow l + 1$
8:       compute and normalize the estimators $\eta_K^* \ \forall K \in \mathcal{T}_h$
9:       for $K \in \mathcal{T}_h$ do
10:          if $POS_K \geq (1 - \mathcal{G}_r) card(\mathcal{T}_h)$ then
11:             $k_K \leftarrow \min(k_K + 1, k_{max})$
12:          else if $POS_T \leq (\mathcal{G}_c) card(\mathcal{T}_h)$ then
13:             $k_K \leftarrow \max(k_K - 1, 1)$
14:          end if
15:       end for
16:       balance the load among processors via re-partitioning
17:       $L_2$ projection of the solution on the new space
18:    end if
19: end for
```

computation, N_{cyc} is the total number of time steps of the simulation, k_{max} is the maximum allowable polynomial degree defined by the user, $\mathcal{N}$ is the number of time-steps between two adaptation cycles or between the simulation beginning and the first adaptation cycle, $\mathcal{G}_r$ is the percentage of the total number of elements that will be marked for refinement, $\mathcal{G}_c$ is the percentage of the total number of elements that will be marked for coarsening, n_{adp} is the number of adaptation cycles to be performed, POS_K is the position of the element numbered from zero and sorted in increasing order according to the estimator η_K^{TOT} or η_K^{EA}. We remark that orthonormal and herarchical modal bases greatly simplify the L_2 projection operators. In practice, the DOFs of the restricted solution are equal to the low-order subset of their high-order representations, while the DOFs of the prolongated solution are the same as the low-order solution with zero high-order components.

IV. Load balancing approach

Any adaptation algorithm applied to parallel simulations generally induces a strong imbalance of the load per partition and a drastic reduction of the parallel efficiency. In such cases, an effective repartition of the computational grid is mandatory to achieve good load balancing. The approach used here exploits the ability of the Metis[1] library to optimize, enforcing multiple constraints, the partition of a weighted graph, where graph vertices correspond to mesh elements, while minimizing the number of faces of the partition boundaries. It is worth noting that the multi-constrained mesh partitioning approach has a significant impact on the parallel efficiency of simulations, as the floating-point operation count in different phases of the solution scales differently with the polynomial degree. For instance, the Jacobian matrix evaluation scales approximately as k^{3d}, while the residual evaluation as k^{2d}. Moreover, it turns out that the additional simultaneous balancing of the operation count related to the volume and surface integrals evaluation improves the parallel efficiency. A good balance of the computational costs related to the different solution phases was achieved by using five constraints. Four constraints stem from balancing the computational cost of residual evaluation and Jacobian assembly on different element and face types, which require different numbers of Gauss integration points. The last constraint balances the computational cost of matrix-vector products. Figure 1 shows the results of some numerical experiments aimed at assessing the parallel efficiency of the p-adaptation strategy for a three-dimensional model test case computed on a grid with 768 elements. The tests were run on an AMD Opteron machine with 2 computational nodes, each with two AMD 6276 Opteron CPUs, for a total of 64 cores. The solutions were advanced in time using the backward-Euler scheme, which requires the solution of one nonlinear system per time step. The solution of the linear system required in the Newton-Raphson approach was obtained using a block-Jacobi preconditioned GMRES iterative solver.

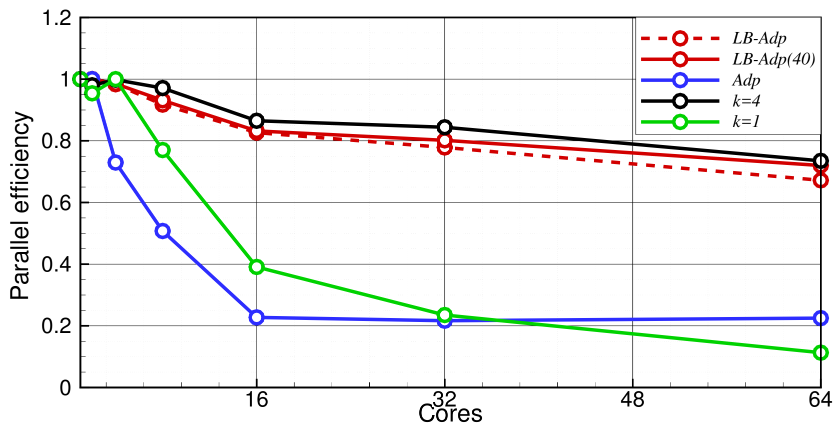


Figure 1. Parallel efficiency of the solver on AMD Opteron processors. $k = 1, 4$ stands for the fixed-p solver, Adp stands for the adaptive solution without employing the load balancing algorithm, $LB - Adp$ refers to the adaptive run with load balancing approach, and $LB - Adp(40)$ refers to scalability obtained using the CPU time required by the final 40 iterations on the adapted discretization.

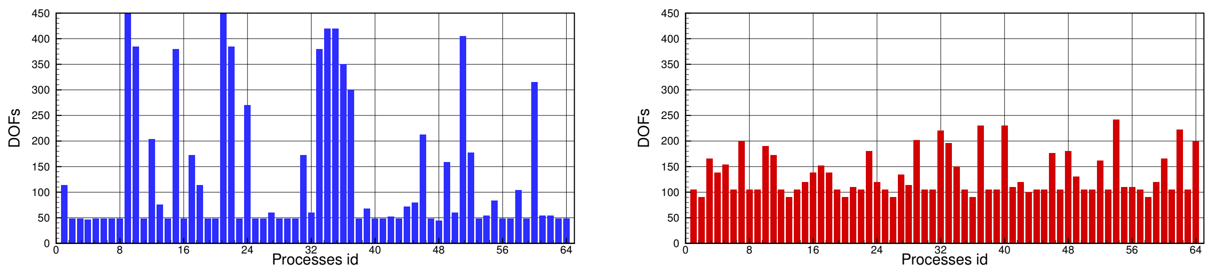
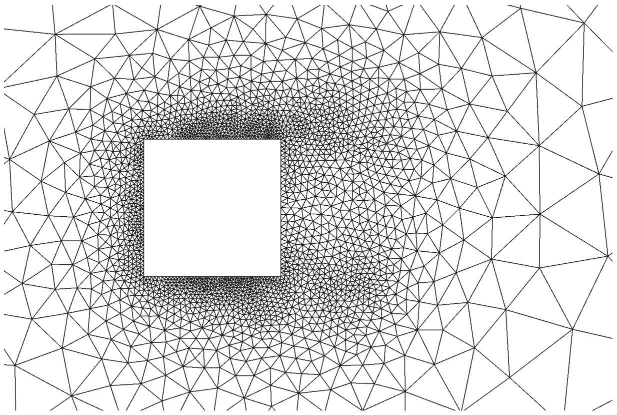


Figure 2. Load of the processors without (left) and with (right) the use of the load balancing approach after six p-refinements for the BTC0 test case.

The curves labeled $k = 1$ and $k = 4$ refer to the computation of solutions for 10 time steps using first- and fourth-degree polynomials, and serve as reference. Considering the rather small number of elements, the lower-order computations scale poorly as proven by the $k = 1$ curve. The p-adaptive solutions were run for 100 time steps by using the backward-Euler scheme, starting from a $k = 1$ approximation and performing 6 adaptation cycles (each time adapting 20% of the elements marked for adaptation), every 10 time steps and setting to $k = 4$ the maximum polynomial degree. In the last 40 time steps no further p-adaptation was performed. The scalability of the p-adaptive solver, running on the unweighted partition of the grid and without load balancing (Adp), is clearly quite poor. Activating the load balancing ($LB - Adp$), the scalability improves and the curve gets closer to the uniform $k = 4$ polynomial degree case computed on the unweighted partitioned gird, even if the final number of Dofs is more than three times smaller for the adaptive computation. The graph in this case was obtained using the multi-constraint partitioning strategy. It is worth noting that the parallel efficiency of the p-adaptive computation, evaluated over the last 40 time-steps, when the solution is already adapted, is even higher, as shown by the $LB - Adp(40)$ curve in Figure 1. Finally, Figure 2 shows the load per processor, after six p-adaptation cycles without (left) and with (right) load balancing.

V. Numerical results

Numerical experiments are devoted to assess the robustness and the accuracy of the proposed methodology in the context of unsteady, chaotic flow problems. First, we assess the methodologies on a two dimensional laminar chaotic flow problem, *i.e.* the two-dimensional flow around a square at $Re = 3\,000$, where the performance of the averaged entropy and output adjoint sensors are compared. Second, we val-


(a) Mesh: 4454 elements

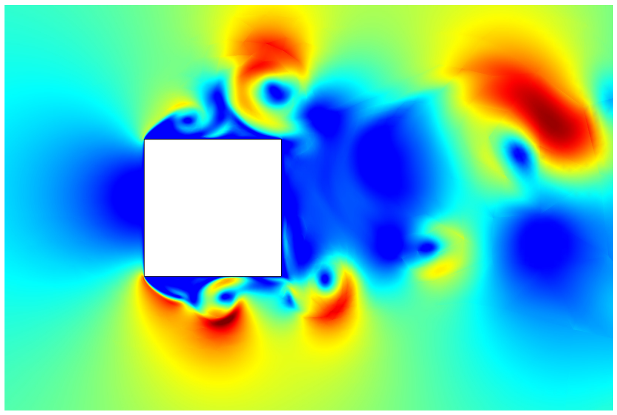
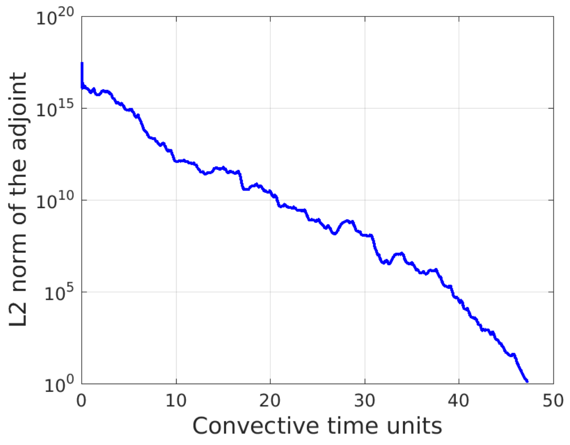

(b) Mach contours (0, 0.2)

Figure 3. Flow around a square. Mesh and Mach contours.



(a) Adjoint norm time history

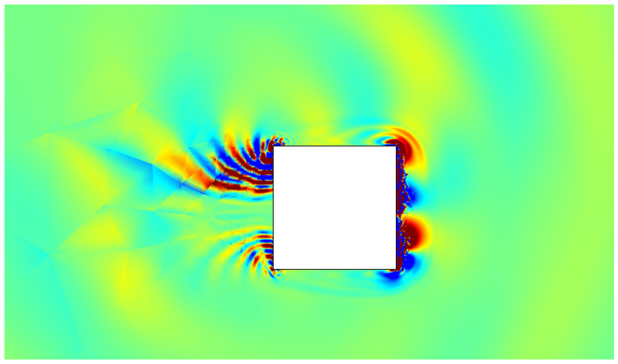

(b) Conservation of mass component of the adjoint at $t = 0$. Scale is $O(10^{17})$.

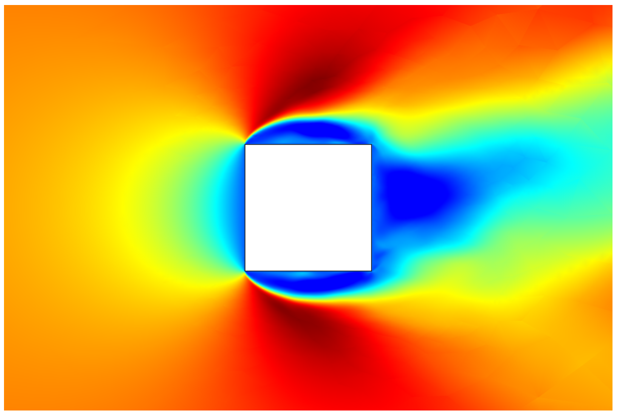
Figure 4. Growth of the adjoint for the time-averaged drag output in reverse time for the two-dimensional square case.

idate the entropy-adjoint adaptive strategy on the implicit LES of the flow around a circular cylinder at $Re = 3\,900$. Third, as proof of concept, we report preliminary results of a higher-Reynolds number test case, the Large Eddy Simulation of the flow around a rectangular cylinder at $Re = 22000$. The results show the effectiveness of the averaged entropy-adjoint based sensor to capture transitional and separated region of the flow, providing an efficient distribution of the degrees of freedom and throughout the domain and a good comparison with experimental data.

A. Two-dimensional square test case

In this section, we compare the performance of the averaged entropy and output adjoint indicators for a two-dimensional simulation. The problem of interest is a unit-square in horizontal flow at Mach number $M = 0.1$ and Reynolds number $Re = 3\,000$. The flow is initialized to free-stream and allowed to develop into a statistically-steady regime, as illustrated in Figure 3, which also shows the mesh used for all of the runs. The output of interest is the time-averaged drag coefficient on the square. We remark that standard unsteady adjoint solutions for this output quickly grows without bound and becomes impossible to use for adaptation, as shown in Figure 4.

Order-adaptive simulations are performed starting from $p = 1$ on every element. At each adaptive iteration, 15% of the elements with the highest error indicator have their order incremented by 1. A burn


(a) Averaged state (x-momentum)

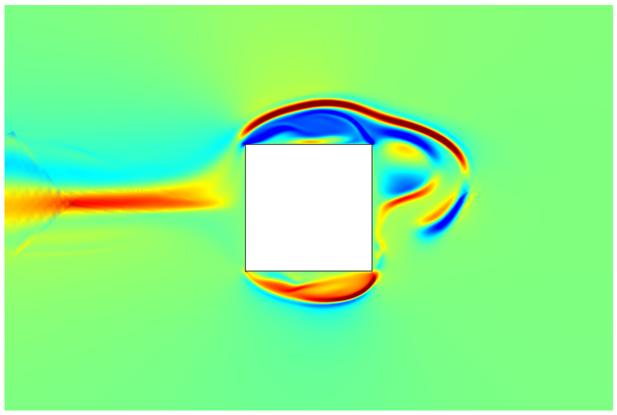
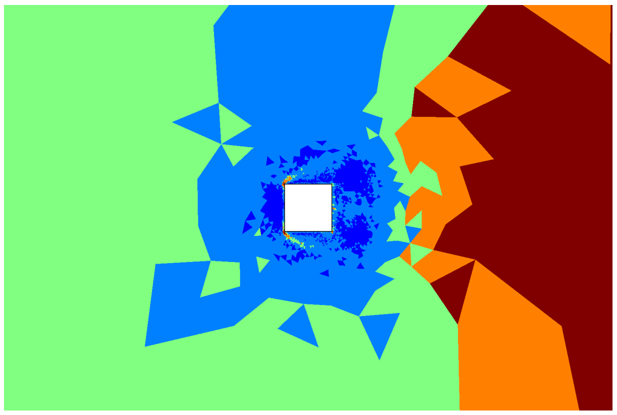
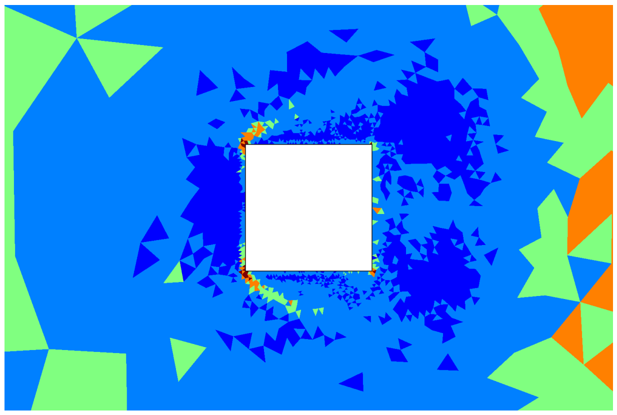
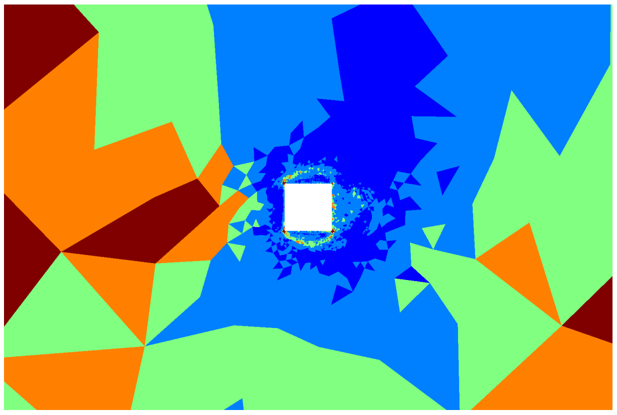

(b) Steady adjoint (conservation of x-momentum)

Figure 5. Flow around a square: averaged state and the steady-state adjoint computed about this state.


(a) Adapted on the entropy adjoint


(b) Adapted on the entropy adjoint (zoom)


(c) Adapted on the drag adjoint

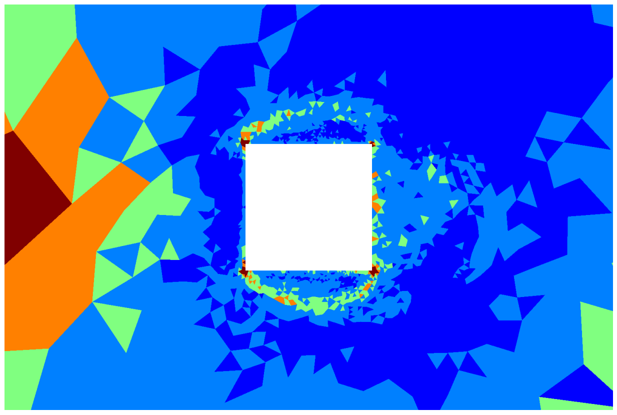
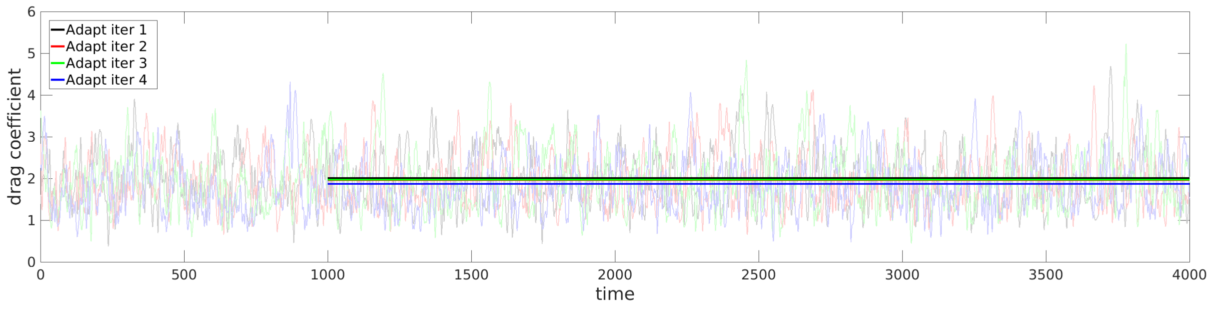
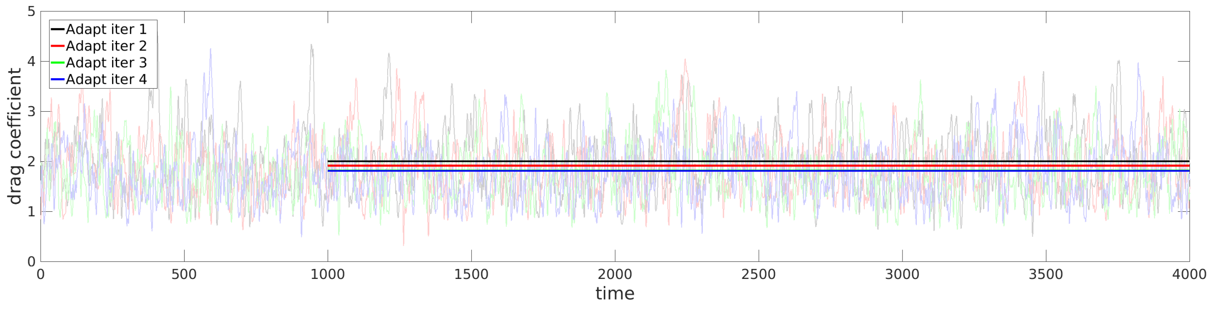

(d) Adapted on the drag adjoint (zoom)

Figure 6. Flow around a square: order distribution (1–5) on the final adapted meshes.

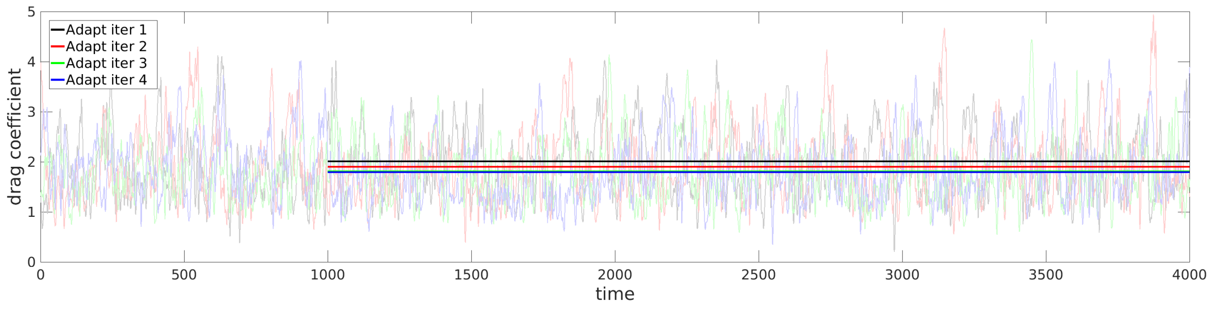
time of 10 convective time units (here CTU, where 1 CTU is the time taken by flow at free-stream speed to traverse the length of the square) is used prior to the averaging for each unsteady run. The averaging time is also 10 CTU. A sufficiently small time-step and high-order time-stepping scheme are used to minimize temporal errors, so that the dominant source of error is spatial. The temporal discretization is therefore not adapted.



(a) Adapted on the residual



(b) Adapted on the entropy adjoint



(c) Adapted on the drag adjoint

Figure 7. Flow around a square: instantaneous drag coefficient time histories and average values (thick lines) on adapted meshes.

Figure 5 shows the time-averaged state and corresponding steady-state adjoint solution from the last adjoint-based adaptive simulation. Note the lack of symmetry, particularly in the adjoint solution, which can be addressed to some extent by increasing the averaging time.

Five adaptive iterations were run using three indicators: the entropy-adjoint weighted residual, the output-adjoint weighted residual, and an unweighted residual. The latter was computed by using the L_1 norm of the time-averaged residual on each element as the indicator, without any adjoint weight. The other adaptive indicators also use the time-averaged residual, but with entropy or output adjoint weights. Figure 6 shows the order distributions on the final adapted meshes for each indicator. We see that a common area to refine includes the corners of the square, particularly in the front and extending along the dominant flow direction towards the rear of the square. The entropy-adjoint indicator additionally targets elements behind the square, whereas the output-adjoint indicator targets elements ahead of the square. This difference is caused by the opposite orientation of the primal versus adjoint wake – the adjoint wake extends ahead of the square.

Once the adapted order fields were obtained, they were tested in long-time simulations. In each such simulation, a burn time of about 100 CTU was first run starting with the final solution from the adaptation sequence. Then, an averaging time of about 300 CTU was simulated to compute the average drag coefficient.

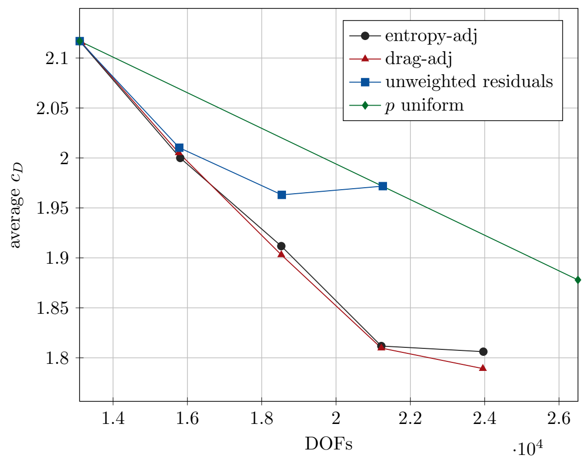


Figure 8. Flow around a square: convergence of the average drag coefficient output on adapted and uniformly-refined order fields.

Figure 7 shows the instantaneous drag coefficient time histories computed using the various adapted order fields. The solid horizontal lines indicate the average drag coefficients, and we observe the smallest difference between the last two adaptive iterations.

Finally, Figure 8 compiles the average drag coefficient results for all of the adaptive simulations and plots them versus the spatial degrees of freedom of the order field. Included in this plot is the convergence of uniform order refinement. Note that this plot shows the drag coefficient itself, not the error. We see that both the entropy and output adjoint adaptations yield order fields that quickly converge to an average drag coefficient in the vicinity of what appears to be the true value (which is lower than the starting point). The unweighted residual, on the other hand, does not converge well, and this is due to the lack of a weight that cuts off adaptation far away from the square: many of those areas still have high residuals due to large elements and passing transient/acoustic waves, even though their order has little impact on the output. These results suggest that for this class of simulations involving bluff bodies and separated flow regions, there exist a similarity between entropy and output adjoint indicators for statistically-steady, yet chaotic simulations. We note however that although the outputs are similar, the order fields differ, suggesting that the optimum distribution of orders may not be overly narrow.

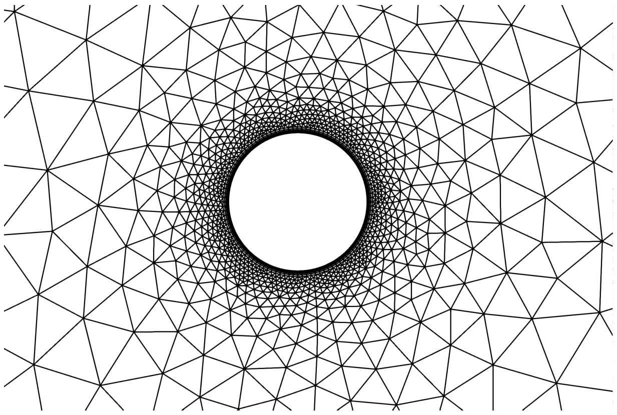
B. Three dimensional test cases

The robustness and efficiency of the proposed approach are evaluated by computing the implicit LES of two problems: i) three-dimensional flow past a circular cylinder at $Re = 3\,900$ and $M = 0.1$; ii) turbulent flow over a square cylinder at $\alpha = 0°$ and Reynolds number $\mathrm{Re} = 22\,000$ and $M = 0.1$. Span-wise periodicity is assumed for both test cases. The meshes used in this work have been generated with a 2D high-order version of a fully-automated hybrid mesh generator based on the advancing-Delaunay strategy[30] and extruded in the span-wise direction. We remark that we didn't specify any refinement in the wake region, as we aim at adapting the wake through the use of higher-order polynomials.

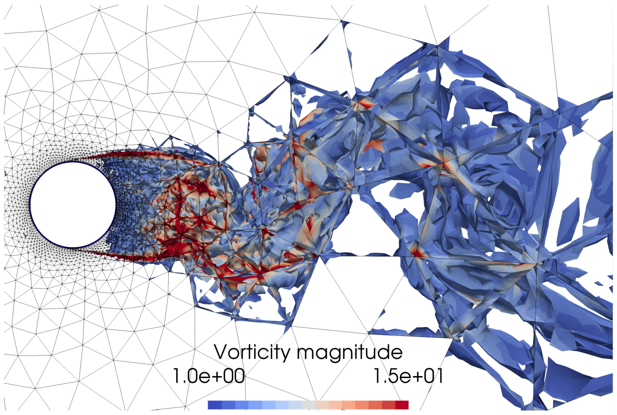
In both cases the simulations have been initialized by computing, in sequence, uniform $\mathbb{P}^0$ and $\mathbb{P}^1$ solutions. After initialization, the adaptation process is activated and driven the entropy adjoint approach. In all the computations, coarsening was disabled, *i.e.*, $\mathcal{G}_c = 0$, and $\mathcal{G}_r$ was set to 0.2 to refine the 20% of elements with the highest estimated error. Both the simulations were performed using 15 KNL nodes, with 68 CPUs each, on the MARCONI A2 HPC system provided by CINECA, the Italian Supercomputing Center.

1. *Turbulent flow around a circular cylinder*

The transitional turbulent flow around a circular cylinder at $Re = 3\,900$ has been solved on grid made of 41865 mesh elements, with quadratic edges in the vicinity of the walls. The grid was obtained by extruding a two-dimensional using 12 elements in the span-wise direction. A circular farfield boundary has been placed at $30R$, with R the radius of the cylinder. Figure 9 shows a snapshot of the computational domain, as well as the instantaneous Mach contour of a fully-developed solution. The problem was analyzed in several previous

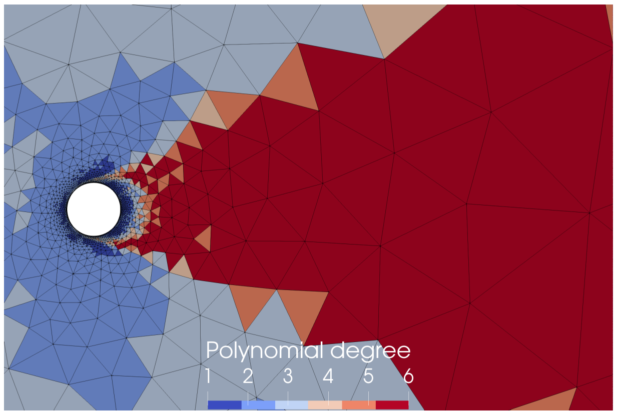


(a) Mesh: 41 865 elements

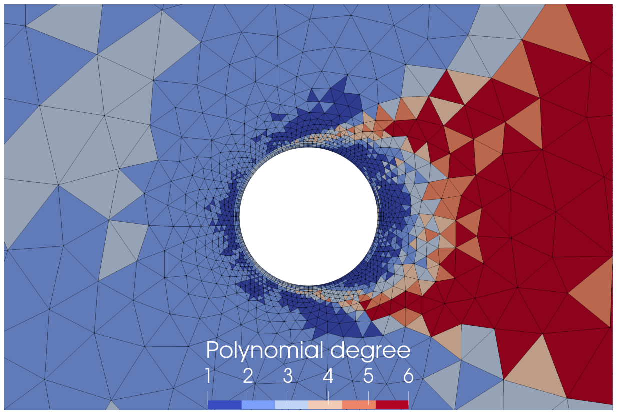


(b) Q-criterion iso-contours coloured by the vorticity magnitude

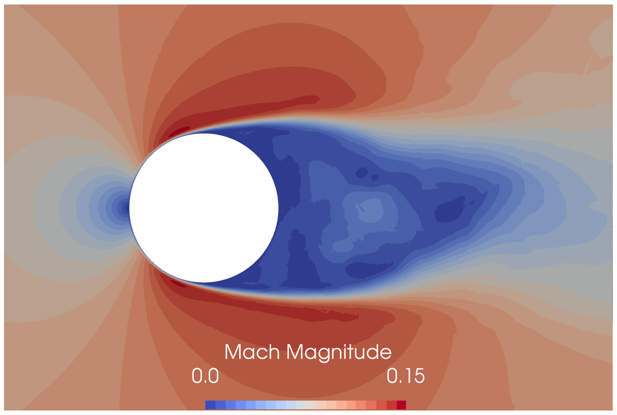
Figure 9. LES solution of the flow around a circular cylinder. Computational mesh, 41 865 elements with quadratic edges (left). Istantaneous field (Q-criterion) coloured by the vorticity magnitude (right).



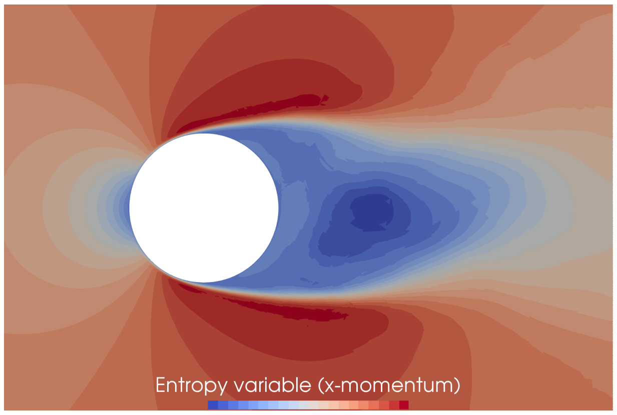
(a) Polynomial order distribution



(b) Polynomial order distribution (zoom)



(c) Average Mach contour



(d) Average entropy variable (x-momentum)

Figure 10. LES solution of the flow around a circular cylinder: polynomial order distribution, Mach number and x-momentum entropy variable contours.

papers and it is part of the test case suite of the International Workshop on high-order methods.[31] The simulation was adapted on 7 consecutive cycles, where the 20% of the elements was targeted for polynomial order increase at each cycle. The final adapted setting was used to compute an average solution over

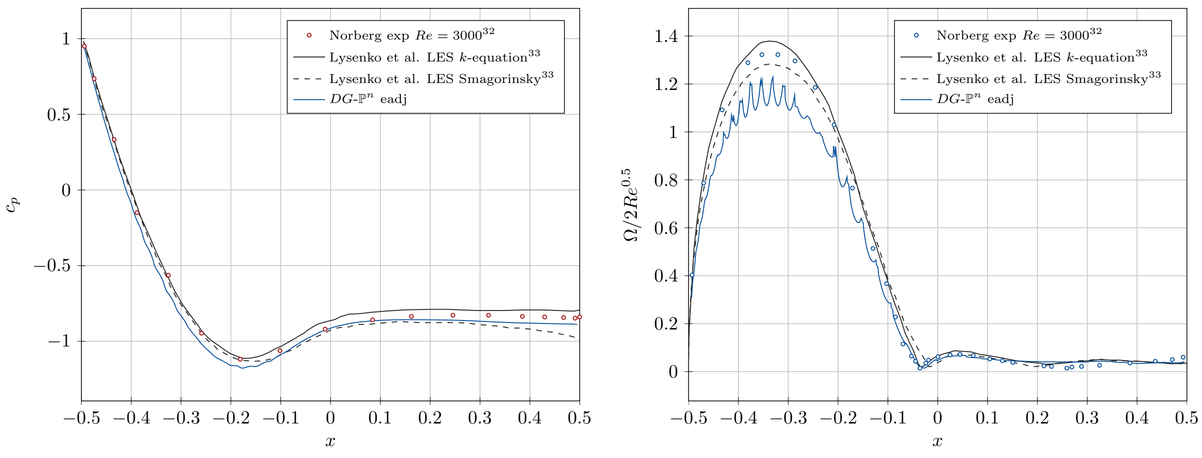


Figure 11. Circular cylinder $Re = 3900$. Span-wise and time-averaged pressure coefficients c_p (left) and non-dimensional wall vorticity $\Omega/2Re^{0.5}$ (right) along the cylinder.

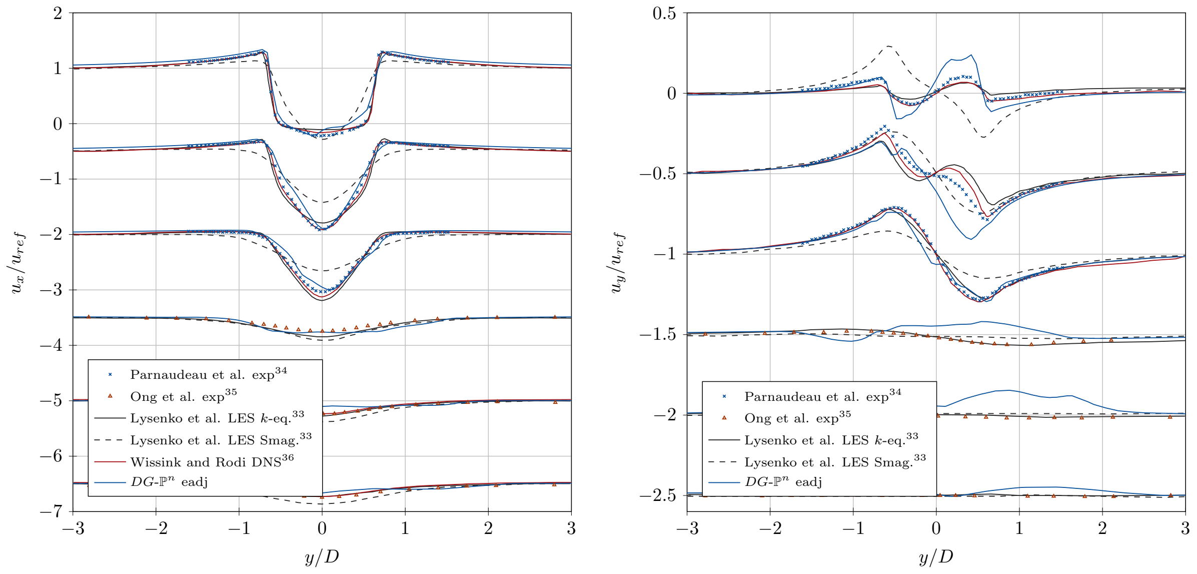


Figure 12. Circular cylinder $Re = 3900$. Mean stream-wise u_x and cross-wise u_y velocity at different locations in the wake of the cylinder.

$T = 11.36$ CTU, here defined as $R/(V_\infty)$. The final number of DoF obtained from the adaptation process and used to compute the average is $6.95 \cdot 10^5$.

Figures 10(a) and 10(b) show the polynomial distribution obtained on the temporal average T. The time and span-wise average flow field over the period T is reported in Figures 10(c) and 10(d). The results show that the entropy adjoint approach based on the averaged entropy variables and averaged residual vector is able to increase the resolution on the shear layer and wake regions of the domain, similarly to what happens in the two-dimensional case. Figures 11 shows the pressure coefficient and the wall vorticity profile on the wall, which compare favourably with experimental data from Norberg[32] and previous numerical simulations.[33] Figures 12 and 13 extend the comparison to the average velocity, velocity fluctuation and pressure on different stations along the wake region.

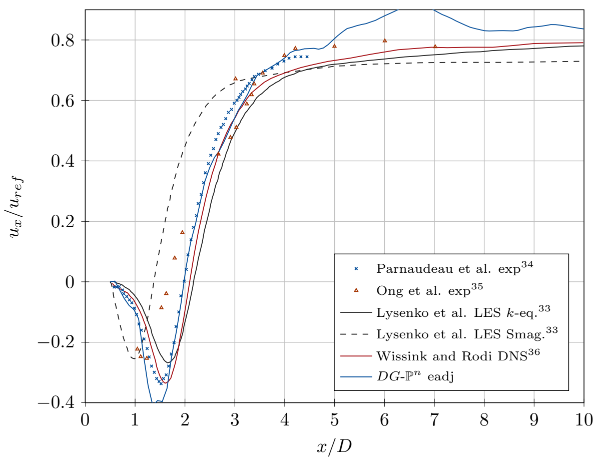
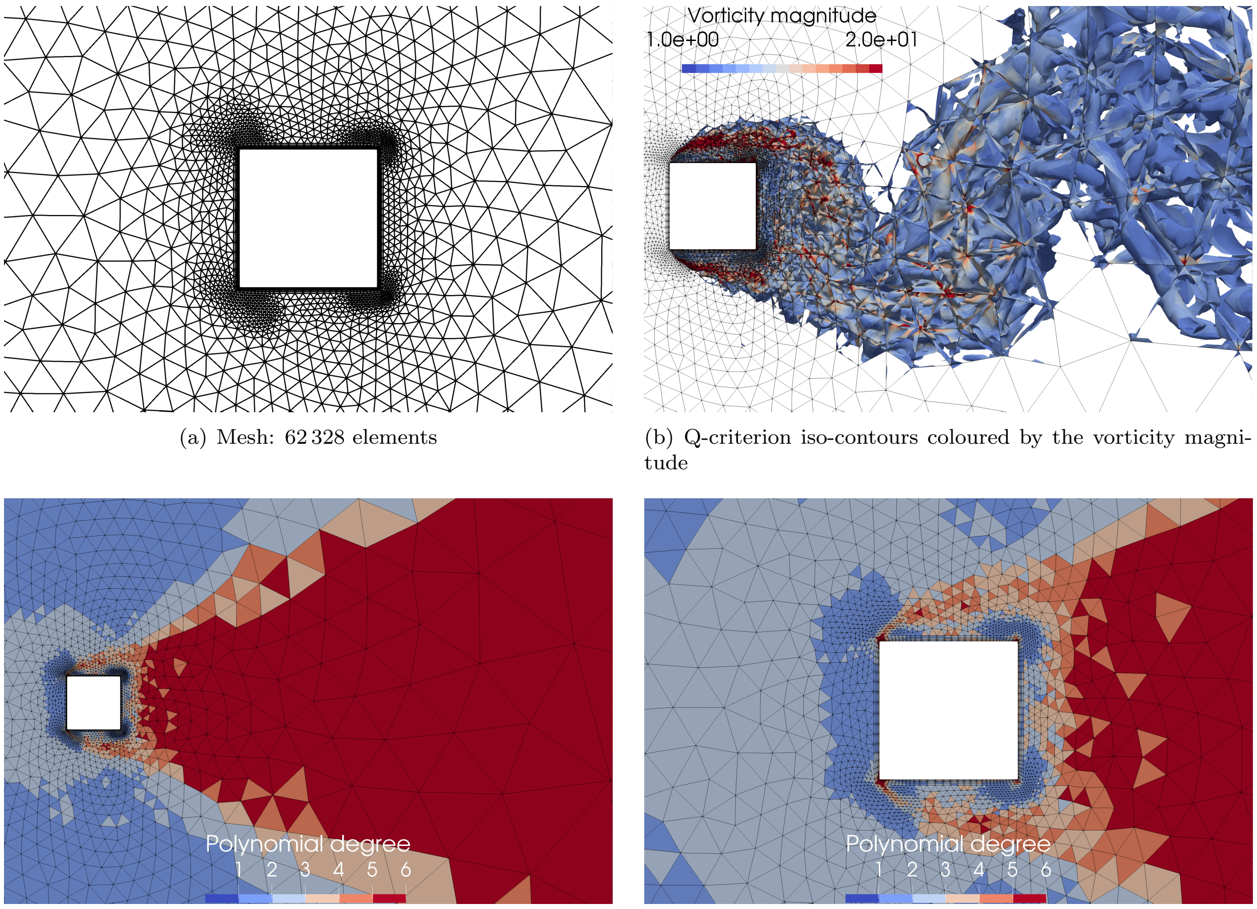


Figure 13. Circular cylinder $Re = 3900$. Mean stream-wise velocity u_x/u_{ref} along the centerline in the wake of the cylinder.



(a) Mesh: 62 328 elements

(b) Q-criterion iso-contours coloured by the vorticity magnitude

(c) Polynomial order distribution

(d) Polynomial order distribution (zoom)

Figure 14. LES solution of the flow around a square cylinder. Detail of the computational mesh with linear edges (top-left); Istantaneous field (Q-criterion) coloured by the vorticity magnitude (top-right); polynomial order distribution (bottom-left); detail of the polynomial order distribution in the vicinity of the wall (bottom-right).

2. *Turbulent flow around a square cylinder*

As proof of concept, the performance of the adaptive indicator is tested on the flow over a square cylinder. The case is computed for a Reynolds number of $\mathrm{Re} = 22\,000$, with an angle of attack $\alpha = 0°$. The hybrid 3D mesh consists of 62 328 elements, with linear edges. The farfield is rectangular of extension $[-10.5D, 20D] \times [-21D, 21D]$. The mesh was extruded in the span-wise direction using 15 elements while imposing a periodicity of $4D$. A detail of the computational mesh and the preliminary results of the computation are depicted in Fig. 14, which shows Mach number contours after six adaptation cycles after an initial burn-time of 2.75 CTU. In particular, Figure 14(c) shows how high order polynomials are clustered in the shear layer and in the wake, which is shifted upstream with respect to the cylinder test case. This is consistent with the instantaneous flow field, where the breakdown to turbulence is shifted upstream due to the higher Reynolds number. The computation of the long-time average results using the adapted polynomial distribution is left for future work.

VI. Conclusions

In this work, a strategy based on an adjoint p-adaptation approach is presented to increase the computational efficiency of a high-order DG solver in the context of unsteady compressible flows simulations. In particular, the entropy-adjoint approach[10] is applied to the run-time average field is adopted to drive the adaptation. The fine space adjoint solution is computed using a patch reconstruction approach, which avoids the need of assembling a finer space problem and maintains the computational costs low. A dynamic load balancing strategy after each adaptation iteration is proposed, based on a multi-constraint domain decomposition algorithm. The application to two-dimensional flow around a square cylinder shows that the approach performs similarly to a drag-based averaged adjoint adaptation method at a cheaper computational cost for simulations involving separated flows, demonstrating the benefits of using the proposed adaptation method. Three dimensional simulations involving the implicit LES of compressible turbulent flows show the suitability of the solver to reduce the computational costs of scale-resolving simulations involving bluff bodies and separated flows using massively parallel platforms. Comparisons of output quantities of engineering interest are also be considered, highlighting the accuracy of the adapted discretization. Future work will be devoted to the extension of the current methodologies for the simulation of incompressible flows, the use of matrix-free iterative solution strategies[37] and preconditioners[38] to further increase the computational efficiency.

VII. Acknowledgement

We acknowledge the CINECA award, under the ISCRA initiative (grant number HP10BEE6C4), for the availability of high performance computing resources and support. K. Fidkowski acknowledges support from the U.S. Department of Energy under grant DE-FG02-13ER26146/DE-SC0010341.

References

[1]G. Karypis and V. Kumar. METIS: Unstructured graph partitioning and sparse matrix ordering system, version 5.0. Technical report, 2009.

[2]M. Tugnoli, A. Abbà, L. Bonaventura, and M. Restelli. A locally p-adaptive approach for large eddy simulation of compressible flows in a DG framework. *Journal of Computational Physics*, 349:33–58, 2017.

[3]F. Naddei, M. de la Llave Plata, V. Couaillier, and F. Coquel. A comparison of refinement indicators for p-adaptive simulations of steady and unsteady flows using discontinuous Galerkin methods. *Journal of Computational Physics*, 376:508–533, 2019.

[4]R. Hartmann and P. Houston. Adaptive discontinuous galerkin finite element methods for the compressible euler equations. *Journal of Computational Physics*, 183(2):508–532, 2002.

[5]K. J. Fidkowski and Y. Luo. Output-based space–time mesh adaptation for the compressible Navier–Stokes equations. *Journal of Computational Physics*, 230(14):5753–5773, 2011.

[6]M. Ceze and K. J. Fidkowski. Drag prediction using adaptive discontinuous finite elements. *Journal of Aircraft*, 51(4):1284–1294, 2014.

[7]K. J. Fidkowski. Output-based space–time mesh optimization for unsteady flows using continuous-in-time adjoints. *Journal of Computational Physics*, 341:258–277, 2017.

[8]F. Bassi, L. Botti, A. Colombo, A. Crivellini, A. Ghidoni, A. Nigro, and S. Rebay. Time integration in the discontinuous Galerkin code MIGALE - unsteady problems. In *IDIHOM: Industrialization of High-Order Methods-A Top-Down Approach*, pages 205–230. Springer, 2015.

[9]A. Colombo, G. Manzinali, A. Ghidoni, G. Noventa, M. Franciolini, A. Crivellini, and F. Bassi. A p-adaptive implicit discontinous Galerkin method for the under-resolved simulation of compressible turbulent flows. In *7nd European Conference on Computational Fluid Dynamics*, 2018.

[10]K. J. Fidkowski and P. L. Roe. An entropy adjoint approach to mesh refinement. *SIAM Journal on Scientific Computing*, 32(3):1261–1287, 2010.

[11]K. J. Fidkowski, M. Ceze, and P. L. Roe. Entropy-based drag-error estimation and mesh adaptation in two dimensions. *Journal of Aircraft*, 49(5):1485–1496, 2012.

[12]K. T. Doetsch and K. J. Fidkowski. A combined entropy and output-based adjoint approach for mesh refinement and error estimation. AIAA Paper 2018–0918, 2018.

[13]K. J. Fidkowski and D. L. Darmofal. Review of output-based error estimation and mesh adaptation in computational fluid dynamics. *AIAA journal*, 49(4):673–694, 2011.

[14]M. Ceze, L. Diosady, and S.M. Murman. Development of a high-order space-time matrix-free adjoint solver. In *54th AIAA Aerospace Sciences Meeting*, page 0833, 2016.

[15]G. Zenoni, T. Leicht, A. Colombo, and L. Botti. An agglomeration-based adaptive discontinuous Galerkin method for compressible flows. *International Journal for Numerical Methods in Fluids*, 85(8):465–483, 2017.

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

[17]J. J. Gottlieb and C. P. T. Groth. Assessment of Riemann solvers for unsteady one-dimensional inviscid flows of perfect gases. *Journal of Computational Physics*, 78(2):437–458, 1988.

[18]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, *Proceedings of the 2nd European Conference on Turbomachinery Fluid Dynamics and Thermodynamics*, pages 99–108, Antwerpen, Belgium, March 5–7 1997. Technologisch Instituut.

[19]P. L. Roe. Approximate riemann solvers, parameter vectors, and difference schemes. *Journal of computational physics*, 43(2):357–372, 1981.

[20]J. Lang and J. Verwer. ROS3P—An accurate third-order Rosenbrock solver designed for parabolic problems. *BIT*, 41(4):731–738, 2001.

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

[22]S. Balay, M. F. Adams, J. Brown, P. Brune, K. Buschelman, V. Eijkhout, W. D. Gropp, Kaushik D., M. G. Knepley, L. C. McInnes, F. Barry, K. R. Smith, and H. Zhang. PETSc Web page. `http://www.mcs.anl.gov/petsc`, 2014.

[23]E. J. Nielsen and B. Diskin. Discrete adjoint-based design for unsteady turbulent flows on dynamic overset unstructured grids. *AIAA journal*, 51(6):1355–1373, 2013.

[24]Y. Shimizu and K. J. Fidkowski. Output error estimation for chaotic flows. In *46th AIAA Fluid Dynamics Conference*, page 3806, 2016.

[25]Y. Shimizu and K. J. Fidkowski. Output-based error estimation for chaotic flows using reduced-order modeling. In *2018 AIAA Aerospace Sciences Meeting*, page 0826, 2018.

[26]P. J. Blonigan, P. Fernandez, S. M. Murman, Q. Wang, G. Rigas, and L. Magri. Toward a chaotic adjoint for LES. *arXiv preprint arXiv:1702.06809*, 2017.

[27]M. Braack, E. Burman, and N. Taschenberger. Duality based a posteriori error estimation for quasi-periodic solutions using time averages. *SIAM Journal on Scientific Computing*, 33(5):2199–2216, 2011.

[28]D. Destarac. Far-field/near-field drag balance and applications of drag extraction in CFD. *VKI Lecture Series*, 2:3–7, 2003.

[29]R. Rannacher. Adaptive Galerkin finite element methods for partial differential equations. *Journal of Computational and Applied Mathematics*, 128(1-2):205–233, 2001.

[30]A. Ghidoni, E. Pelizzari, S. Rebay, and V. Selmin. 3D anisotropic unstructured grid generation. *International journal for numerical methods in fluids*, 51(9-10):1097–1115, 2006.

[31]4th international workshop on high order methods. `https://how4.cenaero.be`.

[32]C. Norberg. Flow around a circular cylinder: Aspects of fluctuating lift. *Journal of Fluids and Structures*, 15(3):459 – 469, 2001.

[33]D. A. Lysenko, I. S. Ertesvåg, and K. E. Rian. Large-eddy simulation of the flow over a circular cylinder at Reynolds number 3900 using the OpenFOAM toolbox. *Flow, Turbulence and Combustion*, 89(4):491–518, Dec 2012.

[34]P. Parnaudeau, J. Carlier, D. Heitz, and E. Lamballais. Experimental and numerical studies of the flow over a circular cylinder at reynolds number 3900. *Physics of Fluids*, 20, 08 2008.

[35]L. Ong and J. Wallace. The velocity field of the turbulent very near wake of a circular cylinder. *Experiments in Fluids*, 20(6):441–453, Apr 1996.

[36]J. Wissink and W. Rodi. Numerical study of the near wake of a circular cylinder. *International Journal of Heat and Fluid Flow*, 29:1060–1070, 08 2008.

[37]M. Franciolini, A. Crivellini, and 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*, 159:276–294, 2017.

[38]M. Franciolini, L. Botti, A. Colombo, and A. Crivellini. p-Multigrid matrix-free discontinuous Galerkin solution strategies for the under-resolved simulation of incompressible turbulent flows. *arXiv preprint arXiv:1809.00866*, 2018.