The authors ascertain that this paper has been passed through a spelling and grammatical check before submission

Truncation Error-Based Anisotropic p-Adaptation for Unsteady Flows for High-Order Discontinuous Galerkin Methods

Andrés M. Rueda-Ramírez[a,*], Gerasimos Ntoukas[b], Gonzalo Rubio[b,c,*], Eusebio Valero[b,c], Esteban Ferrer[b,c]

[a] *Department of Mathematics and Computer Science, University of Cologne, 50931 Cologne, Germany*
[b] *ETSIAE-UPM (School of Aeronautics), Universidad Politécnica de Madrid, Plaza de Cardenal Cisneros 3, 28040 Madrid, Spain*
[c] *Center for Computational Simulation, Universidad Politécnica de Madrid, Campus de Montegancedo, Boadilla del Monte, 28660 Madrid, Spain*

Abstract

We extend the tau-estimation method to unsteady problems and use it to adapt the polynomial degree for high-order discontinuous Galerkin simulations of unsteady flows. The adaptation is local and anisotropic and allows capturing relevant unsteady flow features while enhancing the accuracy. We first revisit the definition of the truncation error, studying the effect of the treatment of the mass matrix. Secondly, we extend the tau-estimation strategy to unsteady problems. Finally, we present and compare two adaptation strategies for unsteady problems: the dynamic and static methods.

We test the efficiency of the p-adaptation strategies with unsteady two-dimensional simulations using the Euler and Navier-Stokes equations. Since the method relies on the exponential convergence of the scheme, we focus in laminar test cases. The adaptation methods enable reductions in the number of degrees of freedom with respect to uniform refinement, leading to speed-ups of up to x4.5 and x2.2 for Euler and Navier-Stokes.

Keywords: High-order discontinuous Galerkin, Anisotropic p-adaptation, Unsteady p-adaptation.

1. Introduction

High-order DG methods are expected to be the engine of the next generation of CFD codes [1]. Commonly, high-order DG schemes are formulated as multi-domain spectral methods. As a result, besides the increased accuracy of spectral (high-order) methods, they also provide a compact stencil and, therefore, a local character, a feature that makes them highly parallelizable and flexible for complex 3D geometries [2, 3]. Moreover, DG methods can handle non-conforming meshes with hanging nodes and different polynomial degrees efficiently [4–6]. This feature makes them well suited for mesh adaptation strategies.

Among the different DG formulations, the discontinuous Galerkin spectral element method (DGSEM) [7, 8] is a nodal (collocation) version of the DG method. Traditionally, this method employs tensor-product Lagrange basis functions within hexahedra and stores data at the Gauss or Gauss-Lobatto nodes of the quadrature rule. It is important to note that DGSEM is not restricted to tensor product elements; it showcases adaptability to simplex elements as well, as elaborated in [9]. The use of a quadrature rule with the same number of nodes as the approximate solution equips the DGSEM with a diagonal mass matrix and

*Corresponding authors:
Email addresses: aruedara@uni-koeln.de (Andrés M. Rueda-Ramírez), g.rubio@upm.es (Gonzalo Rubio)

very cheap-to-compute operators. This translates into a computational cost that maintains linear scalability with the increasing number of degrees of freedom, even as the polynomial degree is raised, as highlighted in [10, 11]. In fact, the computational cost of the DGSEM has been estimated to be a factor of four smaller than other conventional DG methods [12]. In addition, since the DGSEM uses tensor-product bases, it can handle p-anisotropic discretizations efficiently [13, 14], i.e., discretizations with different polynomial degrees in each coordinate direction. For all those properties, the DGSEM has been used in a wide range of applications, including the discretization of the incompressible Navier-Stokes equations [15], the compressible Navier-Stokes equations [16, 17], the Cahn Hilliard equation [18], magneto-hydrodynamics [19, 20], among others.

In their famous review paper, Wang et al. [3] point out that one of the challenges the high-order community must address to impact the design process and replace traditional low-order codes is the development of efficient mesh adaptation strategies. The idea behind these strategies is to reduce the number of degrees of freedom (DOFs) while maintaining high accuracy, which translates into shorter computational times and reduced storage requirements. Local adaptation can be performed by subdividing or merging elements (h-adaptation), by enriching or reducing the polynomial degree in certain elements (p-adaptation), by relocating the position of the nodes in a mesh (r-adaptation). For all these strategies, it is of paramount importance to identify the flow regions that require refinement or coarsening with a local error estimation.

Adaptation strategies have been classified according to the type of error measure that is employed as feature-based adaptation, adjoint-based adaptation, and local error-based adaptation. A comparison of these three approaches was performed by Fraysse et al. [21] for finite volume approximations and by Kompenhans et al. [22] and Naddei et al. [23] for high-order DG methods. The feature-based adaptation is the classical approach and uses easy-to-compute error measures that depend on the flow features. They rely on the assumption that high errors are expected where the flow is more difficult to resolve. Hence, refinement is predicted where high velocity, density or pressure gradients are identified [24, 25]. For DG discretizations, an easy-to-compute feature-based adaptation criterion is the assessment of jumps across element interfaces [26–28]. The main disadvantage of these methods is that there is no direct relation between the adaptation criterion and the numerical errors, and thus, the accuracy is not easily predictable. Additionally, the only way to solve steady-state problems is to adapt iteratively.

A second and more sophisticated approach is known as adjoint-based adaptation. In this approach, a functional target is defined (e.g., drag or lift in external flow aerodynamics), and the adjoint problem is solved to obtain a spatial distribution of the functional error, which is then used to adapt the mesh. This technique was originally developed for structural analysis using FEM by Babuška and Miller [29, 30], and has been used recently for adaptation strategies in DG methods [31–33]. The main drawback of this approach is the high computational cost to solve the adjoint problem and the storage requirements needed to save the error estimators, especially in unsteady flows. Moreover, only the error of the functional analyzed is guaranteed to be reduced, whereas the error of other functionals may deteriorate.

A computationally more efficient alternative is the local error-based adaptation, which is based on the assessment of any measurable (not feature-based) local error in all the cells of the domain [32]. The local error-based adaptation methods are interesting since, in contrast to feature-based methods, they provide a way to predict and control the overall accuracy and are computationally cheaper than adjoint-based schemes [13, 22]. For those reasons, local error-based adaptation strategies are retained in this work. A large amount of effort has been invested in the development of reliable local error-based adaptation methods. Mavriplis [34, 35] has used Estimations of the local discretization error to develop hp-adaptation techniques for the spectral element method. Residual-based p-adaptation is also a local error-based adaptation method, which uses the residual to measure how accurate the local approximation is. This method was originally developed for Finite Elements (FE) and has been successfully used with DG methods [23, 31]. In the case of modal (hierarchical) DG methods, a possibility is to employ low cost error estimates that take advantage of the modal approximation to drive p-adaptation procedures, such as the Variational Multiscale (VMS) indicator by Kuru and De la Llave Plata [36], or the spectral decay indicator by Persson and Peraire [25].

Other techniques used to dynamically adapt the dissipation properties of high-order methods include troubled cell-indicators, Multidimensional Optimal Order Detection (MOOD) and adaptive dissipation con-

trol. Troubled cell-indicadors were initially introduced in [37] and are commonly employed in conjunction with TVD-TVB limiters [38–40]. MOOD methods involve assessing the maximum achievable accuracy order at each cell that ensures the fulfillment of physical and numerical admissibility conditions [41–46]. Finally, adaptive dissipation control aims at adapting the dissipation properties of the scheme by means of, e.g., modal filters (which are closely related to p-adaptation) [47–50].

In this work, we favor truncation error estimators, another local error-based alternative to drive a mesh adaptation method. The truncation error is related to the discretization error through the Discretization Error Transport Equation [51], where it acts as a local source term. This relation makes it useful as an indicator for mesh adaptation methods [52, 53], since refining the mesh where the truncation error is high reduces the discretization error in all the mesh [54], with an additional advantage: truncation error estimation requires less computational effort than adjoint methods. Finally, it has been shown that controlling the truncation error targets the numerical accuracy of all functionals at once [13, 55], ensuring that adapting a mesh using the truncation error leads necessarily to an error decrease in any other functional (e.g., lift or drag).

The τ-estimation method proposed by Brandt [56], which estimates the local truncation error by injecting a fine grid solution into coarser meshes, has been used to perform local error-based mesh adaptation in low-order schemes [21, 53, 57–60]. Rubio et al. [61] extended the τ-estimation approach to high-order methods using a continuous Chebyshev collocation method. Later, Rubio et al. [54] applied it to DGSEM discretizations. Kompenhans et al. [13] applied the τ-estimation approach to perform steady-state p-adaptation using the Euler and Navier-Stokes equations, and showed that a reduction of the truncation error increases the numerical accuracy of all functionals at once. Furthermore, Kompenhans et al. [22] also showed that truncation error-based adaptation can exhibit better performance than feature-based adaptation. In contrast to most local error-based adaptation methods, where multiple error estimation and adaptation stages are needed in a steady-state solution, the τ-estimation method generates a unique prediction of what polynomial degree is needed for a desired truncation error threshold. Therefore, in steady state the adaptation strategy is to converge a high-order approximation (reference mesh) to a specified global residual and then to perform a single error estimation followed by a corresponding p-adaptation process. Besides, the truncation error is known to decay exponentially in smooth solutions [13, 54]. Therefore, if the estimation is good, it is possible to extrapolate the behavior and predict the polynomial degree needed for a desired error threshold [13, 62].

Being a relatively recent technique, p-adaptation methods that use τ-estimators have only been applied to steady-state solutions with high-order methods [13, 14]. In this work, we propose a methodology to extend this technique to unsteady problems. The rest of this work is organized as follows. First, in Section 2.2, we show that the truncation error can be formulated in several forms, depending on the choice of the continuous and discrete partial differential operators. Herein, a thorough analysis of the formulation that is traditionally used in the DG community [13, 22, 54, 62, 63] is presented, a new formulation for the truncation error is proposed, and a comparative analysis of both techniques is detailed. Second, we provide a strategy to estimate the truncation error in unsteady problems in Section 2.3. Third, we propose two p-adaptation algorithms for unsteady problems in Section 2.4, which use an unsteady τ-estimation method: (i) a dynamic p-adaptation strategy, which performs several stages of estimation and p-adaptation throughout a simulation and second, and (ii) a static p-adaptation strategy, which performs several truncation error estimation stages, but only one p-adaptation stage. Finally, the methods are applied to unsteady problems modeled by the compressible Euler and Navier-Stokes equations in Section 3, and a detailed analysis of their performance is presented. The most important findings of this paper are summarized in Section 4.

2. Numerical Methods

2.1. The Discontinuous Galerkin Spectral Element Method

We consider the approximation of systems of conservation laws,

$$\partial_t \mathbf{q} + F(\mathbf{q}) = \mathbf{0}, \quad \text{in } \Omega, \tag{1}$$

subject to appropriate boundary conditions, where $\mathbf{q}$ is the state vector of conserved variables, and $F(\mathbf{q}) = \vec{\nabla} \cdot \vec{\mathbf{f}}$ is the continuous partial differential operator, where $\vec{\mathbf{f}}$ is a flux block vector, which depends on $\mathbf{q}$.

In an advection-diffusion conservation law, such as the Navier-Stokes equations, the flux vector can be written as

$$\vec{\mathbf{f}} = \vec{\mathbf{f}}^{\,a}(\mathbf{q}) - \vec{\mathbf{f}}^{\,\nu}(\mathbf{q}, \vec{\nabla}\mathbf{q}), \tag{2}$$

where $\vec{\mathbf{f}}^{\,a}$ is the advective flux and $\vec{\mathbf{f}}^{\,\nu}$ is the diffusive flux. Because of the dependency of the diffusive flux on $\vec{\nabla}\mathbf{q}$, (1) is a second order PDE. Following Arnold et al. [64], (1) can be rewritten as a first-order system,

$$\begin{cases} \partial_t \mathbf{q} + \vec{\nabla} \cdot \left(\vec{\mathbf{f}}^{\,a}(\mathbf{q}) - \vec{\mathbf{f}}^{\,\nu}(\mathbf{q}, \vec{\mathbf{g}}) \right) = \mathbf{0} \ , \ \text{in } \Omega, & \text{(3a)} \\ \vec{\nabla}\mathbf{q} = \vec{\mathbf{g}} \ , \ \text{in } \Omega. & \text{(3b)} \end{cases}$$

To obtain the DGSEM-version of (3), the computational domain is subdivided into non-overlapping hexahedral elements, all variables are approximated by piece-wise Lagrange interpolating polynomials of degree N that are continuous in each element, but allowed to be discontinuous across element interfaces: $\mathbf{q} \leftarrow \mathbf{q}^N$, $\vec{\mathbf{f}} \leftarrow \vec{\mathbf{f}}^{\,N}$ and $\vec{\mathbf{g}} \leftarrow \vec{\mathbf{g}}^{\,N}$. Furthermore, (3a) and (3b) are multiplied by an arbitrary polynomial (test function) of degree N, the derivative terms are integrated by parts, and all integrals are evaluated numerically with a quadrature rule of $N+1$ points, to obtain

$$\begin{cases} J_j w_j \partial_t \mathbf{q}_j^N - \displaystyle\int_{\Omega^e}^N \vec{\mathbf{f}}^{\,N} \cdot \vec{\nabla}\phi_j \mathrm{d}\Omega^e + \int_{\partial\Omega^e}^N \hat{\mathbf{f}}\phi_j \mathrm{d}S^e = \mathbf{0}, & \text{(4a)} \\ \qquad - \displaystyle\int_{\Omega^e}^N \mathbf{q}^N \vec{\nabla}\phi_j \mathrm{d}\Omega^e + \int_{\partial\Omega^e}^N \phi_j \hat{\mathbf{q}}\vec{n} \mathrm{d}S^e = J_j w_j \vec{\mathbf{g}}_j^{\,N} & \text{(4b)} \end{cases}$$

for each degree of freedom of each element. In (4), $\hat{\mathbf{f}}$ and $\hat{\mathbf{q}}$ are the numerical traces of the flux and the solution, respectively, the superindex N on the integrals denotes the evaluation with a quadrature rule of $N+1$ points, the functions ϕ_j are the so-called basis functions, which are tensor product expansions of the Lagrange interpolating polynomials, the J_j are the Jacobians of the geometry transformation with which the mesh is created, and the w_j are the weights of the quadrature rule. The derivation of (4) is given in [8, 65].

The discretization of the system can be compactly written as

$$\underline{\mathbf{M}}^N \frac{\mathrm{d}\mathbf{Q}^N}{\mathrm{d}t} + \mathfrak{F}^N(\mathbf{Q}^N) = \mathbf{0}, \tag{5}$$

where $\underline{\mathbf{M}}^N$ is the mass matrix of the system, $\mathbf{Q}^N$ a vector with all the unknowns, and $\mathfrak{F}^N$ the discrete partial differential operator.

The mass matrix of the DGSEM is diagonal. Therefore, (5) is often rewritten as

$$\frac{\mathrm{d}\mathbf{Q}^N}{\mathrm{d}t} + (\underline{\mathbf{M}}^N)^{-1} \mathfrak{F}^N(\mathbf{Q}^N) = \mathbf{0}. \tag{6}$$

The DGSEM allows the selection of different polynomial degrees (N) for each element. Moreover, it allows distinct polynomial degrees in each spatial dimension (N_x, N_y, N_z), enabling anisotropic p-adaptation. Handling the geometric representation of p-non-conforming faces in general curvilinear hexahedral elements follows the guidelines outlined in [66]. The coupling between the faces of elements with different polynomial degrees follows the mortar method [67]. Determining the polynomial degree within each element can occur either prior to initiating the simulation (concurrently with mesh generation) or automatically during steady or unsteady simulations. The automatic adaptation process relies on a sensor that designates which elements require refinement or coarsening. In this work we will show how to use the truncation error as a sensor for adaptation in unsteady flows.

The simulations showcased in this paper were conducted utilizing the open source code HORSES3D [17]. For a more comprehensive understanding of the formulation, please refer to the aforementioned paper and its associated references.

2.2. Formulation of the Truncation Error

The truncation error is defined as the difference between the discrete partial differential operator and the continuous partial differential operator, both applied to the exact solution of the problem. This is often known as Generalized Truncation Error Expresion (GTEE) [68, 69]. For the problem at hand, we take the difference between the discrete equation (6) applied to the sampled continuous solution, $\mathbf{I}^N\mathbf{q}$, and the sampled continuous equation (1) to obtain:

$$\frac{\mathrm{d}\mathbf{I}^N\mathbf{q}}{\mathrm{d}t} - \mathbf{I}^N\partial_t\mathbf{q} + (\underline{\mathbf{M}}^N)^{-1}\mathfrak{F}^N(\mathbf{I}^N\mathbf{q}) - \mathbf{I}^N F(\mathbf{q}) = \tilde{\boldsymbol{\tau}}^N, \tag{7}$$

where $\tilde{\boldsymbol{\tau}}^N$ is the truncation error and $\mathbf{I}^N$ is a projection operator used to project the solution from one space to another. Here we are simply sampling the continuous solution into our discrete space.

Assuming that the restriction operator commutes with the time derivative ($\mathbf{I}^N\partial_t\mathbf{q} = \frac{\mathrm{d}\mathbf{I}^N\mathbf{q}}{\mathrm{d}t}$), the two first terms in (7) cancel out,

$$\tilde{\boldsymbol{\tau}}^N = (\underline{\mathbf{M}}^N)^{-1}\mathfrak{F}^N(\mathbf{I}^N\mathbf{q}) - \mathbf{I}^N F(\mathbf{q}). \tag{8}$$

The discretization error is defined as the difference between the approximate solution and the exact solution to the problem, i.e., $\boldsymbol{\epsilon}^N = \mathbf{Q}^N - \mathbf{I}^N\mathbf{q}$. Taking the difference between (6) and (1) sampled into the discrete space, we get the following.

$$\frac{\mathrm{d}\mathbf{Q}^N}{\mathrm{d}t} - \mathbf{I}^N\partial_t\mathbf{q} + (\underline{\mathbf{M}}^N)^{-1}\mathfrak{F}^N(\mathbf{Q}^N) - \mathbf{I}^N F(\mathbf{q}) = \mathbf{0}. \tag{9}$$

By the linearity of the time derivative and using the definition of the discretization error,

$$\frac{\mathrm{d}\boldsymbol{\epsilon}^N}{\mathrm{d}t} = -(\underline{\mathbf{M}}^N)^{-1}\mathfrak{F}^N(\boldsymbol{\epsilon}^N + \mathbf{I}^N\mathbf{q}) + \mathbf{I}^N F(\mathbf{q}). \tag{10}$$

Now, for simplicity, we consider that $\mathfrak{F}^N$ is a linear operator (it can be linearized otherwise to achieve a similar result; see, e.g., [70]),

$$\frac{\mathrm{d}\boldsymbol{\epsilon}^N}{\mathrm{d}t} + (\underline{\mathbf{M}}^N)^{-1}\mathfrak{F}^N(\boldsymbol{\epsilon}^N) = -(\underline{\mathbf{M}}^N)^{-1}\mathfrak{F}^N(\mathbf{I}^N\mathbf{q}) + \mathbf{I}^N F(\mathbf{q}) = -\tilde{\boldsymbol{\tau}}^N. \tag{11}$$

Therefore, for linear operators, the discretization error is governed by the same equation as the numerical solution with the addition of the truncation error as a source term. This equation is known as Discrete Error Transport Equation (DETE). As can be seen, the truncation error is interesting for mesh adaptation, as it acts as a source for the generation of discretization error.

The truncation error definition, (8), can be re-scaled with the mass matrix:

$$\boldsymbol{\tau}^N = \underline{\mathbf{M}}^N\tilde{\boldsymbol{\tau}}^N = \mathfrak{F}^N(\mathbf{I}^N\mathbf{q}) - \underline{\mathbf{M}}^N\mathbf{I}^N F(\mathbf{q}), \tag{12}$$

which is equivalent to defining the truncation error as the projection of the difference between discrete and continuous operators on the individual basis functions of the finite element subspace. With this definition, the DETE reads:

$$\underline{\mathbf{M}}^N\frac{\mathrm{d}\boldsymbol{\epsilon}^N}{\mathrm{d}t} + \mathfrak{F}^N(\boldsymbol{\epsilon}^N) = -\mathfrak{F}^N(\mathbf{I}^N\mathbf{q}) + \underline{\mathbf{M}}^N\mathbf{I}^N F(\mathbf{q}) = -\boldsymbol{\tau}^N. \tag{13}$$

The definition (12) has previously been used in [13, 22, 54, 62, 63], and will be called traditional formulation in this work. The definition (8) has been recently used in [55] and will be called, in this work, new formulation. The main advantage of the new formulation is that it is directly related to functional errors, as shown in [55].

Note that previous works in the context of the DGSEM [13, 22, 54, 55, 62, 63] focused primarily on steady-state problems; therefore, the second term in the RHS of (8) and (12) was zero.

We have derived two formulations of the truncation error and showed that each leads to a different version of the DETE: (11) and (13). Although both formulations appear to be similar, some remarks can be made about their properties.

1. The traditional version of the truncation error (12) acts as a source term for the discretization error, after projecting it point-wise on the basis functions ϕ_j, which build the finite element subspace, as shown in (13). On the other hand, the new approximation of the truncation error (8) acts directly as a source term of the pointwise values of the discretization error; see (11).

2. Since the DGSEM is a collocation method and the mass matrix is a diagonal matrix containing the mapping Jacobian and quadrature weights, see, for example, [17], each form of the truncation error can be obtained from the other by scaling it point-wise with $J_j w_j$, see (12). In other words, the main difference between both formulations is the weight that they give to the element size.

3. Due to the strong similarities between the two truncation error approximations, the anisotropic properties and the possibility to estimate the error in a multigrid cycle (see [63]) hold for both error measures.

Due to the similarities of both truncation error formulations, the tilde notation will be dropped in next sections, and the expressions will hold for both, unless the contrary is explicitly stated.

On a final note, in [54] the concept of *isolated* truncation error was introduced. While the standard *non-isolated* truncation error uses all terms appearing in the discrete discontinuous Galerkin variational formulation, (4), the *isolated* truncation error replaces the surface numerical flux functions by simple evaluations of the flux with the inner solution of each element. The *isolated* truncation error has some advantages for mesh adaptation, as shown in [22].

2.3. Truncation Error Estimation in Unsteady Problems

Now that we have defined the truncation error, we need a method to estimate it when the exact solution is not available. Previous works, see [13, 22, 54, 55, 62, 63], consider only steady problems, and, therefore, only the first term of (8) or (12) takes nonzero values. In these works, the exact solution is approximated by a solution obtained on a higher-order mesh ($P > N$),

$$\mathbf{I}^N \mathbf{q} \approx \mathbf{I}^N \mathbf{q}^P, \tag{14}$$

in a process known as τ-estimation [54, 61]. Using this solution, the truncation error is estimated in all coarser meshes $N \in [1, P-1]$. This is useful for adaptation, as the exact polynomial degree required for a given accuracy can be directly read from the estimated truncation errors. If the problem solved has spatial dimension higher than one, the coarser meshes can be generated with anisotropic polynomial degrees (different polynomial degrees for the different spatial dimensions), resulting in the so-called truncation error map. Additionally, the truncation error for polynomial degrees $N > P-1$ can be estimated by τ-extrapolation, and the whole estimation process can be embedded within an anisotropic multigrid cycle. The interested reader is referred to [63] and references therein for details. It is worth noting that the accuracy of the truncation error estimation, as analyzed in [54, 61], relies on the validity of 14. This implies that if the flow is significantly underresolved, the truncation error estimate might be influenced by the discretization error present in the finer mesh. In such scenarios, the truncation error estimation could be coarse, and the insights provided by the methodology would primarily highlight regions of complexity. This situation may arise in turbulent flows or compressible flows featuring shockwaves. However, it is worth mentioning that the τ-estimation method has yielded promising outcomes for mesh adaptation in low-order methods even when shocks are present, as indicated in [58]. This work will not delve into the examination of these more intricate test cases, leaving them for future investigations.

When addressing unsteady problems, our approach involves approximating the second term in (8) and (12). A logical strategy for achieving this approximation is to rely on the high-order solution, denoted as $\mathbf{q}^P$, and approximate the continuous partial differential operator with its high-order counterpart as follows:

$$\mathbf{I}^N F(\mathbf{q}) \approx \mathbf{I}^N (\underline{\mathbf{M}}^P)^{-1} \mathfrak{F}^P(\mathbf{q}^P). \tag{15}$$

This approach introduces minimal overhead, as the term $(\underline{\mathbf{M}}^P)^{-1}\mathfrak{F}^P(\mathbf{q}^P)$ is computed during the calculation of $\mathbf{q}^P$, which is necessary for estimating the truncation error.

2.4. p-Adaptation Strategies

Two adaptation strategies can be identified in unsteady flow simulations: dynamic and static adaptation. These two strategies have already been widely used for unsteady adaptivity. See, for example, [71, 72] for dynamic adaptation methods or [23, 73] for static adaptation methods.

In the following sections, we present a detailed description of how dynamic and static p-adaptation can be implemented for truncation error-based p-adaptation methods. The main difference between the strategies presented here and those encountered in the literature is the way the adaptation algorithms treat the error estimates.

Most of the error estimation strategies available in the literature are designed to mark a number of elements for enrichment or order reduction [23, 71, 72]. As a result, in every adaptation stage, the polynomial degrees are increased or reduced by one. On the contrary, the truncation error estimation provides an exact value for the polynomial degree needed for each coordinate direction of every element after each estimation stage. This property of the τ-estimation method provides several advantages for the p-adaptation of unsteady computations, as will be discussed in the following sections.

2.4.1. Dynamic p-Adaptation

The dynamic p-adaptation is the most straightforward p-adaptation strategy for unsteady flows. It computes an error measure periodically during a simulation and adapts the polynomial degrees of the discretization according to the estimated error, right after every estimation procedure. The adaptation of the polynomial degree in every step follows the procedure introduced in [63]. The dynamic p-adaptation strategy is well suited for transient simulations in which the region of interest changes over time.

Figure 1 illustrates the dynamic p-adaptation process. The interval between adaptation stages, Δt_e, can be specified as a physical time, as a number of iterations (time steps) or can be changed throughout the simulation. At every p-adaptation stage, the underlined process in Figure 1, the storage must be reallocated and the solution projected in the new polynomial spaces. Since this process is done several times during the solution procedure, some overhead is expected. Therefore, the construction of the data structures for the new spatial resolution and the transfer of information are critical steps that must be optimized to enhance the performance. Furthermore, if the τ-estimation method is used, a number of low-order ($N < P$) discretizations are needed to evaluate the truncation error. As a result, an additional overhead is added in the construction of these coarse grids.

Note that a traditional error estimator, which simply marks some elements for refinement or coarsening, may perform poorly with the dynamic p-adaptation strategy of Figure 1. Such an error estimator imposes a one-by-one increase in the polynomial degree. Therefore, if Δt_e is too large, a dynamic p-adaptation strategy may not have enough time to increase the resolution of a zone of the domain before the flow feature of interest goes out of it. In other words, the refinement zones are likely to lag behind the difficult-to-capture flow features. On the contrary, since the truncation error estimator identifies what polynomial degree is needed immediately, the resolution can be increased to the necessary level immediately. As a result, the truncation error estimator may be more suitable to handle larger values of Δt_e than traditional estimators.

It is possible to obtain overshoots in the truncation error estimates if the polynomial degree of the reference mesh, P, is too low, as shown in [63]. Therefore, a truncation error-based dynamic p-adaptation method may suffer unnecessary polynomial degree oscillations that are caused and nurtured by the constant jump between a low and a high P. These polynomial degree oscillations may deteriorate the accuracy and, therefore, should be avoided when possible. A possible way to attenuate this phenomenon is to limit the maximum polynomial degree jump (by element and coordinate direction) after each p-adaptation stage.

Additionally, in parallelized simulations, a dynamic p-adaptation strategy requires dynamic load balancing to maintain an even workload between the processors and avoid deadlocks. Otherwise, the reduction in the number of degrees of freedom achieved with the enhanced spatial discretization may not translate into shorter computation times. The design of efficient dynamic load balancing algorithms is a challenging topic of research that is not treated in this work.

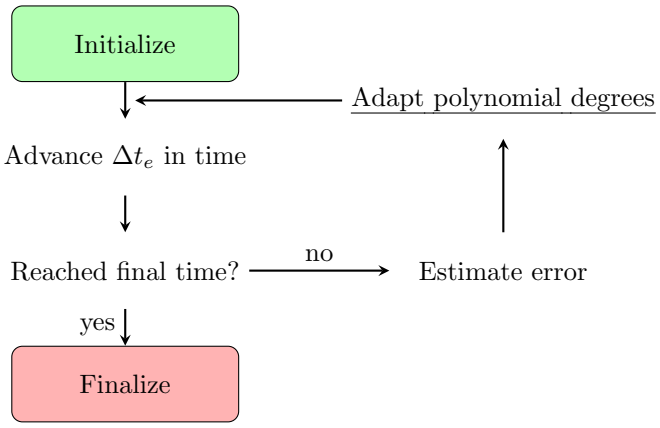


Figure 1: Flowchart of the dynamic p-adaptation.

2.4.2. Static p-Adaptation

The static p-adaptation strategy differs from the dynamic p-adaptation in that only one p-adaptation process is performed after several estimation stages.

Figure 2 presents a flowchart of the static adaptation strategy. First, the solution is advanced in time with a fixed spatial resolution until a final estimation time, T_e, is reached. During this stage, periodic error estimations are performed with an interval of Δt_e but, instead of changing the spatial resolution, the error estimation is stored for future processing. When T_e is reached, a p-adaptation procedure is performed using all the stored error estimates. Subsequently, the simulation is advanced in time with the new fixed spatial resolution until the final time. The static p-adaptation is well suited for simulations in which the features of interest are located in a fixed region of the domain.

Note that a traditional error estimator, which only marks some elements for one-by-one refinement or coarsening, may also perform poorly in the static p-adaptation algorithm of Figure 2. In fact, if such an error estimator is used, the algorithm would have to be slightly modified, so that after the p-adaptation stage, the simulation goes back to the error estimation stage, as in [23]. That extra loop would have to be repeated a specific number of times, or until no element is marked for refinement or coarsening, which represents increased computational cost. Therefore, the ability to predict the exact polynomial that is needed makes the truncation error estimator an attractive indicator for statically p-adapted unsteady simulations.

The static p-adaptation strategy provides several implementation advantages over the dynamic p-adaptation. First, the construction of the data structures and the projection of the solution to the new spatial discretization are no longer critical steps, as the p-adaptation procedure is only done once. Therefore, these operations can even be performed off-line, and their computational cost does not significantly impact the performance of the method. Second, the coarse-grid discretizations that are needed for the truncation error estimation are only constructed once at the beginning of the simulation and used throughout the entire estimation stage. Finally, dynamic load balancing is no longer needed, as the loads must be balanced only once after the p-adaptation step.

The static p-adaptation algorithm has two drawbacks. First, it is only useful for statistically steady flows or where the features that need high spatial resolution are located in a specific region of the domain. Second, the static p-adaptation strategy requires a preliminary simulation to estimate the error.

In the case of aerodynamic simulations of external flow, the flows are usually statistically steady, and the *interesting* flow features are concentrated in a small region of the domain. Furthermore, the estimation time is generally much shorter than the total simulation time.

To process the p-anisotropic truncation error estimation data and feed the p-adaptation algorithm, two main approaches can be identified:

1. At each estimation stage, s, select the polynomial degrees for each of the elements in the mesh, $N_i^{e,s}$,

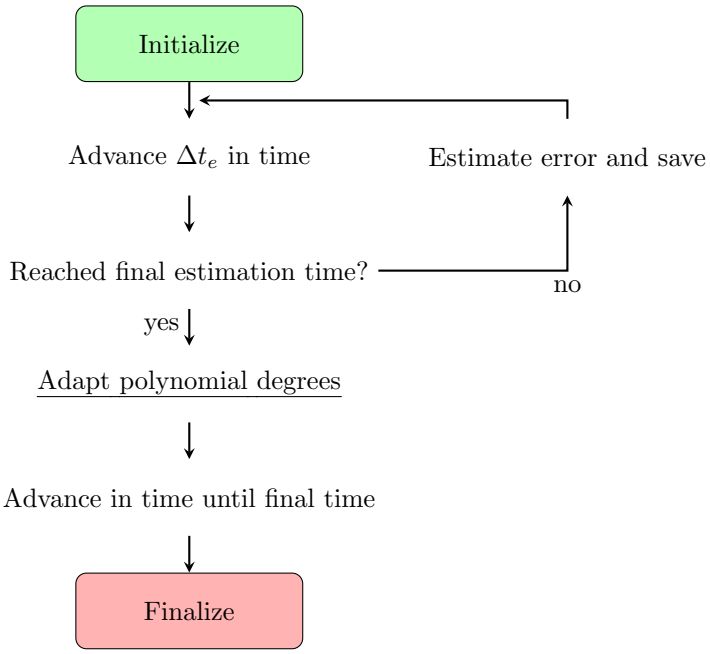


Figure 2: Flowchart of the static p-adaptation.

and then predict a final polynomial degree from the estimates,

$$N_i^e = F(N_i^{e,1}, \dots, N_i^{e,n_e}), \tag{16}$$

where n_e is the number of estimation stages.

2. At each estimation stage, s, generate the truncation error map (see [63] for details) for every element of the mesh, $\|\boldsymbol{\tau}\|_\infty^{e,s}$, compute a total truncation error map for every element based on the estimations,

$$\|\boldsymbol{\tau}\|_\infty^{e} = F\left(\|\boldsymbol{\tau}\|_\infty^{e,1}, \dots, \|\boldsymbol{\tau}\|_\infty^{e,n_e}\right), \tag{17}$$

and select the polynomial degree from the total truncation error map.

The function $F(\cdot)$ can be defined in multiple ways, for example, the average or maximum functions,

$$F_{\text{av}}(N_i^{e,1}, \dots, N_i^{e,s}) = \frac{1}{n_e}\sum_{s=1}^{n_e} N_i^{e,s}, \qquad F_{\max}(N_i^{e,1}, \dots, N_i^{e,s}) = \max_s |N_i^{e,s}|. \tag{18}$$

As a conservative criterion, we use $F_{\max}$ to ensure that the specified truncation error threshold is satisfied throughout the whole simulation.

Approach 1 needs less storage space and can be implemented more easily than *approach 2*. However, it may lead to the over-enrichment of some areas of the domain when combined with p-anisotropic discretizations. To illustrate this, let us consider a specific element in a hypothetical two-stage estimation procedure ($n_e = 2$) of a 2D simulation that uses $F_{\max}$. Let us assume a minimum polynomial degree $N_{\min} = 1$ and a maximum polynomial degree $N_{\max} = 3$ for the p-adaptation. Furthermore, we are interested in selecting the polynomial degree combination that minimizes the number of degrees of freedom (NDOF).

Table 1 shows a possible outcome of the two-stage estimation procedure. There are five polynomial degrees that fulfill the specified error threshold, $\tau_{\max}$, in each estimation stage. Among those, *approach 1* would select the polynomial degrees underlined in red in each estimation stage because they minimize the

instantaneous NDOF. As can be observed, at the end of the estimation, *approach 1* selects the polynomial degrees $N^e = (3,3)$, which correspond to NDOF = 16. This outcome is not optimal because $N^e = (2,2)$, with an associated NDOF = 9, would actually fulfill $\tau_{\max}$ with fewer degrees of freedom.

Table 1: Possible outcome of the *approach 1* to static p-adaptation.

Coordinate direction i	Polynomial degrees with $\left\|\boldsymbol{\tau}^N\right\|_\infty < \tau_{\max}$ Stage $s=1$ $N_i^{e,1}$					Stage $s=2$ $N_i^{e,2}$					Selected degree N_i^e
1	3	2	3	2	1	3	2	3	2	3	3
2	3	3	2	2	3	3	3	2	2	1	3
NDOF	16	12	12	9	8	16	12	12	9	8	16

Approach 2 generates a total truncation error map by applying $F_{\max}$. The use of $F_{\max}$ implies that the polynomial degree combinations that fulfill $\tau_{\max}$ in the total map are those that fulfill $\tau_{\max}$ in all the estimation stages, i.e., the combinations that are not underlined in red. Of that set of combinations, *approach 2* clearly selects $N^e = (2,2)$, with an associated NDOF = 9.

As shown in this simple example, *approach 2* is better than *approach 1*. Therefore, *approach 2* is selected for the static p-adaptation simulations that are shown in this paper.

Although superior, *approach 2* has two drawbacks. First, the extrapolated truncation error map (see, e.g., [63]) must be obtained for each estimation process, so that the total truncation error map can be obtained with (17), which involves more computational resources per estimation stage. Second, if an element is not in the asymptotic range in any of its reference coordinate directions, it may not be possible to extrapolate the values of the inner truncation error map. In such cases, instead of extrapolating the truncation error, we assign a high value to it for $N_i \geq P_i$ as a secure criterion.

3. Numerical Results

In this section, we test the performance of the methods described in this paper to perform p-adaptation of unsteady flow problems using truncation error estimates. The methodology presented in this paper is valid for the *non-isolated* and *isolated* truncation errors. For simplicity, we only use the *isolated* truncation error in this section to drive the p-adaptation procedures.

The governing equations are the two-dimensional compressible Euler/Navier-Stokes equations in conservative form,

$$\partial_t \mathbf{q} + \vec{\nabla} \cdot \left(\overleftrightarrow{\mathbf{f}}^{\,a} - \overleftrightarrow{\mathbf{f}}^{\,\nu} \right) = \mathbf{0}, \tag{19}$$

where the conserved quantities are the mass, momentum and energy (per unit of volume), $\mathbf{q} = [\rho, \rho\vec{v}, \rho E]^T$, ρ is the fluid's density, $\vec{v}$ is the velocity vector, E is specific the total energy (internal energy plus kinetic energy), and $\overleftrightarrow{\mathbf{f}}^{\,a}$ and $\overleftrightarrow{\mathbf{f}}^{\,\nu}$ are called the advective and diffusive flux tensors, respectively.

The flux tensors can be written in compact form as

$$\overleftrightarrow{\mathbf{f}}^{\,a}(\mathbf{q}) = \begin{bmatrix} \rho\vec{v} \\ \rho\vec{v} \otimes \vec{v} + \underline{\mathbf{I}}p \\ \vec{v}(\rho E + p) \end{bmatrix}, \quad \overleftrightarrow{\mathbf{f}}^{\,\nu}(\mathbf{q}, \vec{\nabla}\mathbf{q}) = \begin{bmatrix} \vec{0} \\ \underline{\tau} \\ \underline{\tau}\vec{v} + \kappa\vec{\nabla}T \end{bmatrix}. \tag{20}$$

where p is the (static) pressure, $\underline{\mathbf{I}}$ is the identity matrix, $\underline{\tau}$ is the stress tensor, $T = p/\rho R$ is the temperature, κ is the thermal conductivity, and R is the specific gas constant.

In this paper, we use the calorically perfect gas approximation to compute the pressure,

$$p = (\gamma - 1)\rho e, \tag{21}$$

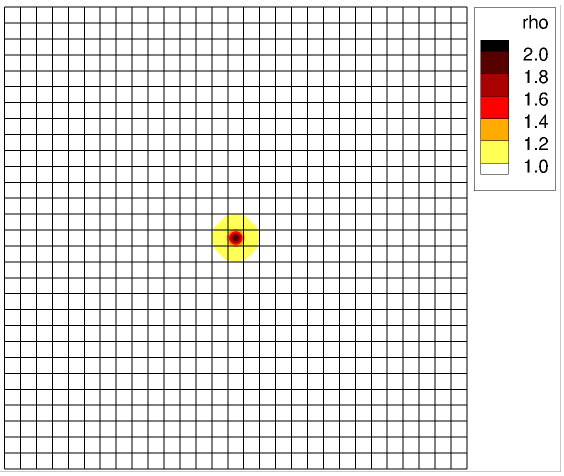


Figure 3: Initial condition of the advected density pulse simulation.

where $e = E - \|\vec{v}\|^2/2$ is the specific internal energy and $\gamma = c_p/c_v$ is the heat capacity ratio. Furthermore, we use Stokes' hypothesis to compute the stress tensor,

$$\underline{\tau} = \mu\left(\left(\vec{\nabla}\vec{v}\right)^T + \vec{\nabla}\vec{v}\right) - \lambda\vec{\nabla}\cdot\vec{v}\underline{\mathbf{I}}, \tag{22}$$

with $\lambda = -\frac{2}{3}\mu$ the bulk viscosity coefficient.

We choose the typical parameters for air: $\mathrm{Pr} = c_p/\mu\kappa = 0.72$, $\gamma = 1.4$, while μ is calculated using Sutherland's law.

The Euler equations of gas dynamics are defined in the same way, but setting the viscosity to zero, $\mu = 0$, such that the viscous flux vanishes.

The p-adaptation procedures are implemented in the open source high-order discontinuous Galerkin framework HORSES3D [17]. All simulations use the Roe solver [74] as the advective numerical flux and BR1 [75] as the diffusive numerical flux. The time-marching scheme in the following examples is Williamson's low-storage third-order Runge-Kutta method [76]. Additionally, the time-step size is dynamically changed using the CFL condition in all simulations (see [17] for details). The main reason is that we want to take time steps that are as large as possible, and the time-step size is a function, among others, of the polynomial degree.

Since the flow features that we analyze have a periodicity in time, the interval between estimation/adaptation stages, Δt_e, is selected as a constant time for each simulation and not as the time that corresponds to a number of time steps.

3.1. Advection of a Density Pulse in a Uniform Flow

In this section, we simulate the advection of a Gaussian pulse in a square domain with periodic boundary conditions with the compressible Euler equations of gas dynamics and $\mathrm{Ma}_\infty = 0.5$. The initial condition is

$$\begin{array}{ll} u = 1 & \rho = e^{5(x^2+y^2)} + 1 \\ v = 0 & p = 1 \end{array}, \tag{23}$$

and the two-dimensional computational domain is tessellated with a structured mesh of 841 quadrilateral elements, as shown in Figure 3. The final time is $t = 29$, when the pulse should be back where it started. All simulations are run in serial with a sixth generation 8-core Intel i7 processor and 32GB of RAM.

In this test case, both forms of the truncation error (traditional and new) perform equivalently. The polynomial degree distributions obtained with the new formulation of the truncation error are almost identical

to those obtained with the traditional formulation, if the specified error threshold, $\tau_{\max}$, is scaled with the (constant) element size. As we discussed in Section 2.2, the main difference between the two formulations is that the traditional truncation error is scaled with the element size. Since the element size is uniform in the whole domain, no significant differences are observed between the traditional and new forms of the truncation error.

We tested static and dynamic p-adaptation algorithms with truncation error thresholds ranging between $10^{-3} \leq \tau_{\max} \leq 1$, and intervals between adaptation/estimation stages ranging between $0.5 \leq \Delta t_e \leq 10$. The polynomial degree is adapted according to the output of the error estimations in the range $1 \leq N_i \leq 8$ for each direction i of every element. Furthermore, the two different polynomial degree jump conditions that were introduced in [63] are considered:

(a) The first one imposes that the polynomial degree after every adaptation stage must fulfill

$$N_i^e \geq \max_{j \epsilon \mathcal{N}(e)} \left\lfloor \frac{2}{3} N_i^j \right\rfloor, \tag{24}$$

where N_i^e is the polynomial degree of element e in the coordinate direction i, $\mathcal{N}(e)$ is the list of the neighbor elements of e, N_i^j is the polynomial degree of the neighbor element j in the matching coordinate direction i, and $\lfloor \cdot \rfloor$ is the integer part floor function.

(b) The second polynomial degree jump condition imposes

$$N_i^e \geq \max_{j \epsilon \mathcal{N}(e)} \left\lfloor N_i^j - 1 \right\rfloor. \tag{25}$$

In the static p-adaptation cases, a preliminary simulation must be run to estimate the error, as seen in Figure 2. Since the pulse always changes position, the preliminary simulation must be run for 29 time units to obtain a significant sample. To have enough points to extrapolate the anisotropic truncation error estimates, the τ-estimation simulation uses a discretization of uniform polynomial degree $P = 3$.

Figures 4 and 5 show the behavior of the dissipation error as a function of the number of degrees of freedom and the computation time for the p-adaptive simulations with the polynomial degree jump conditions of (24) and (25), respectively. The dissipation error is measured as the difference in ρ between the exact solution and the simulation outcome at the centroid of the moving Gaussian. The dispersion error, which can be measured as the absolute value of the position of the Gaussian centroid, is of the order of machine zero (the DGSEM exhibits very low dispersion errors in this case). Note that the computation time needed for the estimation simulation in the static p-adaptation cases has already been added to the simulation time in Figures 4(b) and 5(b).

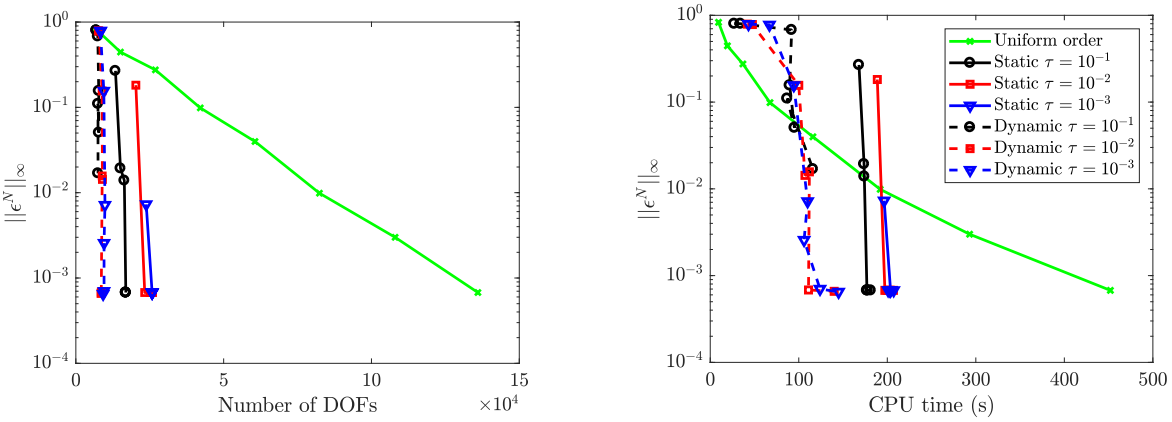
As can be observed, the truncation error-based p-adaptation techniques perform better than uniform refinement when a dissipation error $\left\| \boldsymbol{\epsilon}^N \right\|_\infty < 10^{-2}$ is desired, as they achieve the same errors with fewer degrees of freedom, which results in shorter computation times for a given accuracy.

Figures 4(a) and 5(a) show that the number of degrees of freedom for the statically p-adaptive simulations is $2-4$ times higher than for the dynamically p-adaptive simulation. This makes sense since the static p-adaptation algorithm enriches all the regions through which the pulse passes, whereas the dynamic p-adaptation algorithm effectively follows it. The longer computation times that are observed in Figures 4(b) and 5(b) for the statically p-adaptive simulations are not only the result of this effect, but also of the extra computation time invested in the preliminary τ estimation simulation.

The number of degrees of freedom of the p-adaptive simulations that obey the polynomial degree jump condition (b) (25) is higher than for condition (a) (24). This is expected since many more elements are enriched in the former, as can be observed in Figure 6. The additional enrichment translates to computation times up to 25% higher when using condition (b) (25).

An additional difference between the two polynomial degree jump conditions, which can be inferred from Figure 6, is that condition (a) is more sensitive to the estimation/adaptation interval, Δt_e. On the one hand, in the dynamically p-adaptive simulations and for a given Δt_e, it is more likely that the density pulse

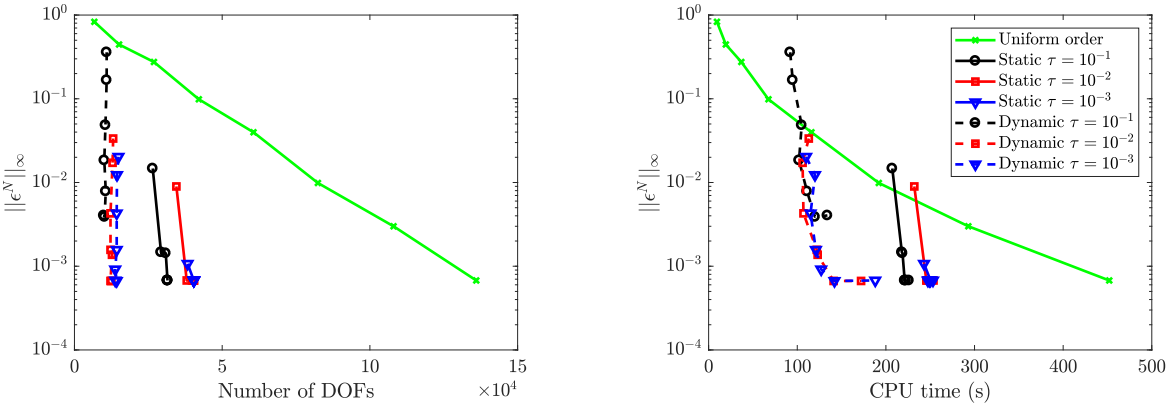
escapes the refined area for condition (a) than for (b), and arrives at an area where no τ-estimation is possible ($P = 1$) or where no extrapolation is possible ($P < 3$). On the other hand, in statically p-adaptive simulations and for a given Δt_e, the refinement areas are more likely to be connected if condition (b) is used instead of (a). This behavior is also illustrated in Figure 7 for the static p-adaptation with the threshold $\tau_{\max} = 10^{-1}$.



(a) Maximum error vs. number of DOFs.

(b) Maximum error vs. CPU-Time.

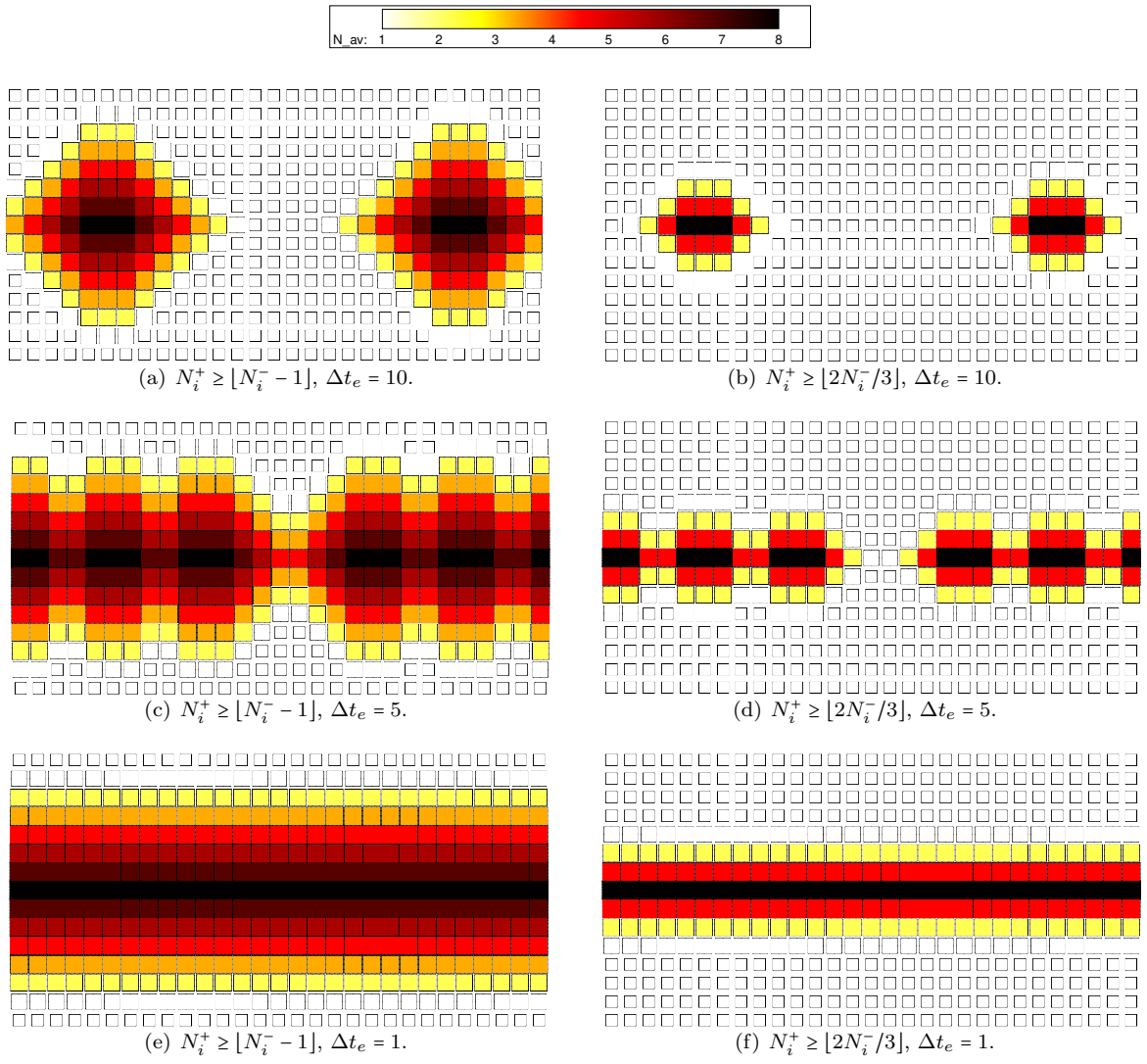
Figure 4: Error performance of static and dynamic p-adaptation procedures for the advection of a density pulse with a polynomial degree jump condition of $N_i^e \geq \max_{j \in \mathcal{N}(j)} \lfloor 2N_i^j/3 \rfloor$ (24). Each point in the plot corresponds to an interval between adaptation/estimation stages, ranging between $0.5 \leq \Delta t_e \leq 10$. Different truncation error thresholds are represented with lines of different colors (black, red and blue). Error performance of the uniform polynomial degree refinement (in green) is also included.



(a) Maximum error vs. number of DOFs.

(b) Maximum error vs. CPU-Time.

Figure 5: Error performance of static and dynamic p-adaptation procedures for the advection of a density pulse with a polynomial degree jump condition of $N_i^e \geq \max_{j \in \mathcal{N}(j)} \lfloor N_i^j - 1 \rfloor$ (25). Each point in the graph corresponds to an interval between adaptation/estimation stages, ranging between $0.5 \leq \Delta t_e \leq 10$. Different truncation error thresholds are represented with lines of different colors (black, red and blue). Error performance of the uniform polynomial degree refinement (in green) is also included.



(a) $N_i^+ \geq \lfloor N_i^- - 1 \rfloor$, $\Delta t_e = 10$.

(b) $N_i^+ \geq \lfloor 2N_i^-/3 \rfloor$, $\Delta t_e = 10$.

(c) $N_i^+ \geq \lfloor N_i^- - 1 \rfloor$, $\Delta t_e = 5$.

(d) $N_i^+ \geq \lfloor 2N_i^-/3 \rfloor$, $\Delta t_e = 5$.

(e) $N_i^+ \geq \lfloor N_i^- - 1 \rfloor$, $\Delta t_e = 1$.

(f) $N_i^+ \geq \lfloor 2N_i^-/3 \rfloor$, $\Delta t_e = 1$.

Figure 6: Average polynomial degree distribution for static p-adaptation with $\tau_{\max} = 10^{-1}$ and different estimation intervals.

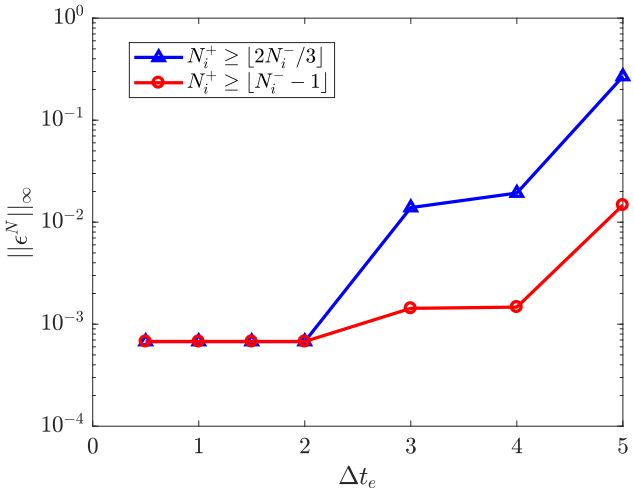
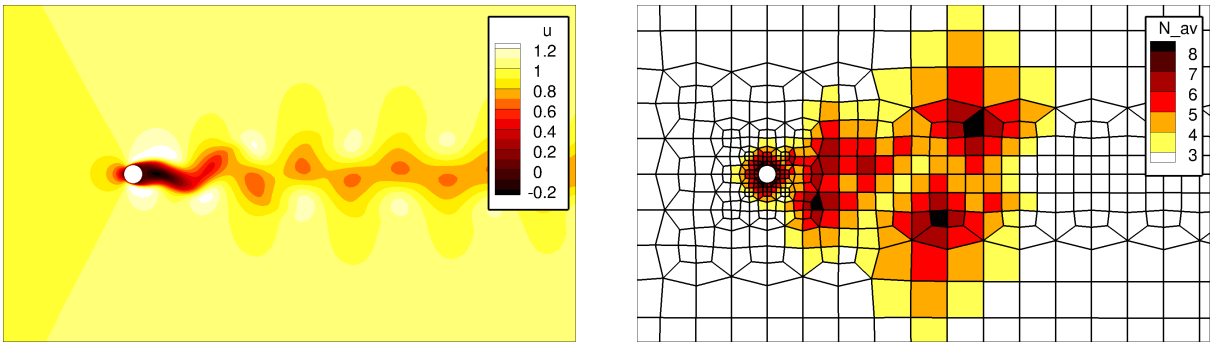


Figure 7: Combined effect of the polynomial degree jump condition and the estimation interval in the static p-adaptation error with threshold $\tau_{\max} = 10^{-1}$.

3.2. Subsonic Flow Past a Cylinder

We simulate the flow around a circular cylinder at a Reynolds number of $\mathrm{Re}_\infty = 100$ and a Mach number of $\mathrm{Ma}_\infty = 0.15$ on a high-order curved ($N_{\mathrm{geo}} = 3$) mesh with 1282 quadrilateral elements and the DGSEM method. We assess the performance of the truncation error-based static and dynamic p-adaptation methods and show that the static p-adaptation algorithm performs well in this example since the solution is statistically steady, as in most external aerodynamic problems.

Figure 8 shows the mesh that was used, the instantaneous horizontal velocity contours, and an instantaneous distribution of polynomial degrees for the dynamic p-adaptation method.



(a) Instantaneous horizontal velocity contours.

(b) Instantaneous average polynomial degrees (dynamic p-adaptation) for $\tilde{\tau}_{\max} = 5$.

Figure 8: Vortex shedding behind a cylinder at $\mathrm{Re}_\infty = 100$.

The results presented in this section were obtained using a 40-core 2.10GHz Intel(R) Xeon(R) Gold 6230 CPU with 170 GB of RAM. Each simulation was run with 10 cores and shared memory parallelization (OpenMP + guided schedule) to compute the spatial terms. Note that this parallel implementation has a near-optimal scalability for p-anisotropic discretizations and the selected OpenMP schedule, as was shown in [63]. We remark that the guided OpenMP scheduling acts directly as a dynamic load balancing technique in the simulations with dynamic p-adaptation.

For the p-adaptive simulations, the new form of the truncation error is retained because it was shown to work more efficiently on the lift and drag predictions than the traditional form. The reason why is easily seen in Figure 9, which shows the contours of the average polynomial degrees for both formulations of the truncation error as the error threshold, $\tau_{\max}$, is reduced in a static p-adaptation method. For a similar number of degrees of freedom, the p-adaptation algorithm that uses the traditional form of τ tends to enrich large elements that are away from the cylinder, whereas the new form tends to enrich only the boundary layer area and the wake. As explained in Section 2.2, the main difference between the two approaches is the weight they assign to the volume of each element.

The truncation error-based static and dynamic p-adaptation algorithms are tested with truncation error thresholds ranging between $10^{-1} \leq \tilde{\tau}_{\max} \leq 10^2$, and four estimation/adaptation intervals $\Delta t_e = 0.5, 1, 3, 6$ in non-dimensional time units, taking into account that the vortex shedding period is expected to be $T = 6$. Furthermore, the polynomial degree jump across faces is limited to $N_i^+ \geq \lfloor N_i^- - 1 \rfloor$ (25), since this condition provides robustness to the simulation and allows larger estimation intervals, as discussed in Section 3.1. Additionally, the maximum polynomial degree was set to $N_{\max} = 8$ and the minimum polynomial degree to $N_{\min} = 3$. This minimum polynomial degree allows the dynamic p-adaptation to always have enough points to perform the directional truncation error extrapolation.

In the dynamic p-adaptation algorithm, the sub-meshes that are used for the truncation error estimation are constructed every Δt_e time units. After that, the error is estimated using the τ-estimation method and the polynomial degrees are changed accordingly. At every adaptation stage, we only allow the polynomial degree to decrease by one in each element to reduce spurious oscillations that may arise because of large

polynomial degree jumps throughout the simulation (a phenomenon discussed in Section 2.4.1). In the static p-adaptation algorithm, the τ-estimation sub-meshes are only constructed once at the beginning of the simulation. Thereafter, an estimation simulation with polynomial degree $P = 4$ is run for a sampling time of $T_e = 12$, i.e., two vortex shedding cycles. The polynomial degrees are then adapted using *strategy 2* (17), and the rest of the simulation runs without further modifications.

The average adaptation cost for the non-isolated and isolated τ-estimation methods is presented in tables 2 and 3. We have collected data for various truncation error thresholds and adaptation intervals Δt_e, utilizing the dynamic p-adaptation method. To enhance clarity, we present results corresponding to different truncation error levels, since the adaptation cost remains consistent across varying Δt_e intervals. These costs were measured serially and normalized against the cost of a single explicit time step with $N = 8$, using identical hardware and a single thread.

The adaptation cost is divided into three categories: estimation, interpolation, and total. The estimation cost encompasses all operations necessary for selecting new polynomial degrees for each element in the mesh. The interpolation cost accounts for projecting the solution from the original to the newly adapted mesh. The total cost includes all operations within the adaptation function. Notably, the total cost significantly exceeds the combined estimation and interpolation costs, largely due to the allocation and deallocation of memory, which is not optimized in our implementation.

A comparison of Tables 2 and 3 reveals that the isolated τ-estimation method incurs a lower cost, attributed to the decoupling of elements and the reduced number of elements marked for refinement. The adaptation cost escalates as the truncation error threshold diminishes. The main reason for that behavior is that a low truncation error threshold results in meshes with higher polynomial degrees, thereby increasing both estimation and interpolation costs due to the necessity of constructing more sub-meshes and requiring higher-order interpolation.

In static adaptation scenarios, the interpolation cost is lower, as it involves only a single interpolation step (as shown in "Adapt polynomial degrees" in Figure 2) after the estimation stage. Moreover, the cost of allocation and deallocation is almost negligible when doing static p-adaptation because the estimation sub-meshes are only constructed only once at the beginning of the estimation stage.

For the specific test case and setups evaluated, we found that the overhead associated with estimation and adaptation remains below 10% of the total computational cost. This overhead is more pronounced in dynamic adaptation, with multiple adaptation routine calls, compared to static adaptation, where adaptation occurs only once after the sampling process concludes.

Table 2: Cost estimation of the non-isolated truncation error for a given error threshold for the cylinder case. We provide the average cost for the estimation of the truncation error, the interpolation to the new polynomial order and the total cost normalised by the time taken to compute one explicit time step with N=8.

TE error thres.	$\frac{\text{Estimation cost}}{\text{N=8 timestep cost}}$	$\frac{\text{Interpolation cost}}{\text{N=8 timestep cost}}$	$\frac{\text{Total cost}}{\text{N=8 timestep cost}}$
10	0.001	0.12	8.86
1	0.001	0.22	11.73
0.1	0.002	0.31	17.11

Table 3: Cost estimation of the isolated truncation error for a given error threshold for the cylinder case. We provide the average cost for the estimation of the truncation error, the interpolation to the new polynomial order and the total cost normalised by the time taken to compute one explicit time step with N=8.

TE error thres.	$\frac{\text{Estimation cost}}{\text{N=8 timestep cost}}$	$\frac{\text{Interpolation cost}}{\text{N=8 timestep cost}}$	$\frac{\text{Total cost}}{\text{N=8 timestep cost}}$
10	0.001	0.13	7.26
1	0.001	0.21	9.84
0.1	0.002	0.33	11.87

Figure 10 shows the performance of the uniform p-refinement, dynamic and static truncation error-based p-adaptation algorithms. The mean absolute lift and the mean drag error (the latter with respect to a

solution of order $N = 9$) are plotted as a function of the number of degrees of freedom (NDOF) and the computation time for each of the methods. Lift and drag are monitored for 100 time units and their average values are computed. The reported computation time is the sum of the CPU time that is needed to advance 100 time units and the CPU time that is needed for the estimation.

Because of the adaptive time-stepping, the monitored variables had to be re-sampled at a uniform time-step sequence to calculate the averages. A brief description of the process to obtain the re-sampled data is provided in Appendix A. In addition, the number of degrees of freedom that is shown for the dynamic p-adaptation simulations corresponds to a weighted average,

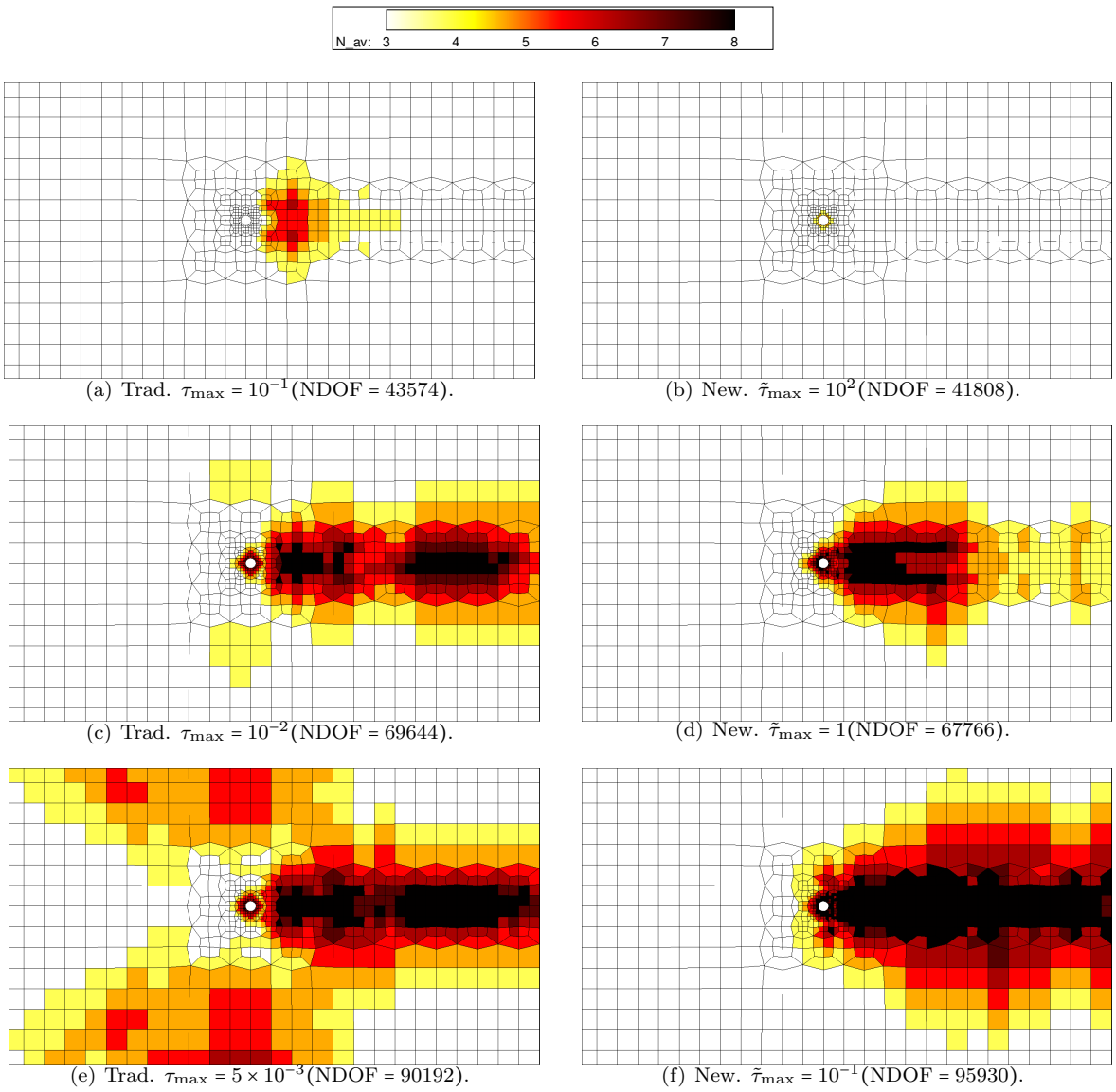
$$\mathrm{NDOF}_{\mathrm{dyn}} = \frac{1}{S}\sum_{i=1}^{S}\mathrm{NDOF}_i, \tag{26}$$

where S is the number of simulation time steps and NDOF_i corresponds to the number of degrees of freedom of the discretization in the iteration i.

It can be observed in Figure 10 that the truncation error-based locally adaptive simulations need fewer degrees of freedom than the simulations with uniform order. Furthermore, in contrast to the advected pulse example, the static p-adaptation method needs fewer degrees of freedom than the dynamic p-adaptation method for the same accuracy. The main reason for this behavior is that the dynamic p-adaptation algorithm might overestimate the polynomial degree needed when the polynomial degree of the reference mesh (used for the estimation), P, is low (a behavior discussed in Section 2.4.1). In fact, the dynamic p-adaptation algorithm is more likely to over-predict the required polynomial degree than the static algorithm since the minimum specified polynomial degree acts sometimes as the estimation polynomial degree in dynamically p-adaptive simulations, $P = N_{\mathrm{min}} = 3$, which is lower than the estimation polynomial degree of the static p-adaptation algorithm, $P = 4$.

The behavior of the error with respect to the computation times is highly dependent on the implementation, the hardware used, and the problem. The results obtained with the current implementation in HORSES3D [17] are reported as a reference. As can be observed, the performance is different for each variable analyzed, but in general a speed-up of about 2.2 can be observed for the static p-adaptation algorithm at the highest level of accuracy that is reached. The dynamic p-adaptation algorithm has the same performance as the static p-adaptation algorithm in some cases, and in some others it exhibits a worse performance. The main reason for that is that the dynamic p-adaptation algorithm is more sensitive to the estimation interval, Δt_e, and that it may also suffer from non-physical oscillations in the solution and its gradients due to the frequent jumps in the polynomial degree.

The truncation error-based p-adaptation methods show the best CPU-time performance when measuring the mean absolute lift, where speed-ups can be observed in virtually all the error range considered for small enough Δt_e. When measuring the mean drag error, the truncation error-based p-adaptation performs relatively similar to the uniform p-refinement with respect to to CPU-time (if Δt_e is small enough) down to an error of $|\bar{C}_d - \bar{C}_d^{N=9}| \approx 5 \times 10^{-5}$. Below that error, the truncation error-based p-adaptation algorithms outperform the uniform refinement technique.



(a) Trad. $\tau_{\max} = 10^{-1}$(NDOF = 43574).

(b) New. $\tilde{\tau}_{\max} = 10^{2}$(NDOF = 41808).

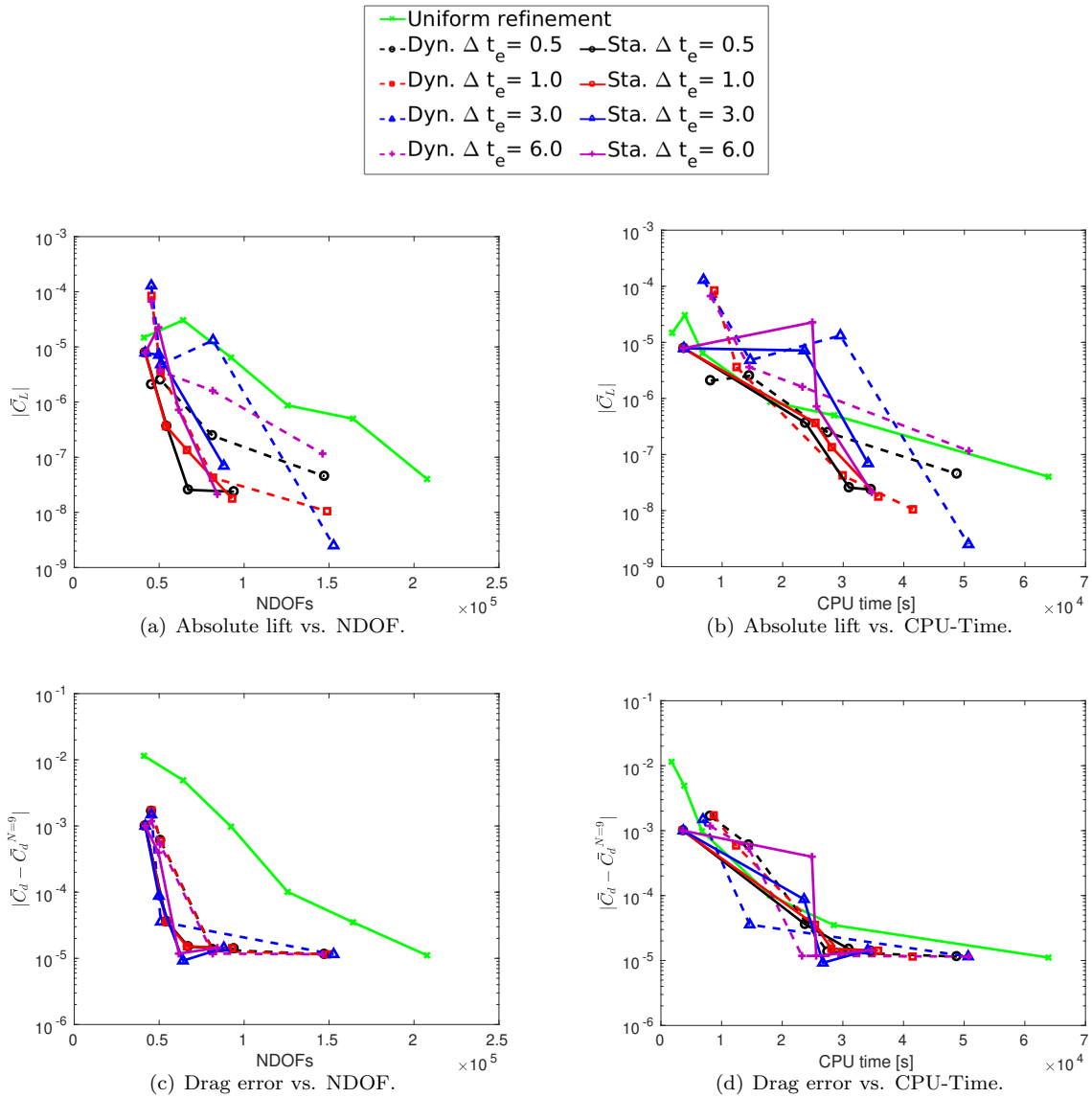
(c) Trad. $\tau_{\max} = 10^{-2}$(NDOF = 69644).

(d) New. $\tilde{\tau}_{\max} = 1$(NDOF = 67766).

(e) Trad. $\tau_{\max} = 5 \times 10^{-3}$(NDOF = 90192).

(f) New. $\tilde{\tau}_{\max} = 10^{-1}$(NDOF = 95930).

Figure 9: Comparison of the two possible formulations of the truncation error: traditional (left) and new (right). Average polynomial degree distribution for static p-adaptation for different error thresholds.



(a) Absolute lift vs. NDOF.

(b) Absolute lift vs. CPU-Time.

(c) Drag error vs. NDOF.

(d) Drag error vs. CPU-Time.

Figure 10: Performance of the static and dynamic p-adaptation procedures for the flow past a cylinder at $\mathrm{Re}_\infty = 100$, $\mathrm{Ma}_\infty = 0.15$. Each point in the graph corresponds to a truncation error threshold, ranging between $10^{-1} \leq \tilde{\tau}_{\max} \leq 10^{2}$. Different intervals between adaptation/estimation stages are represented with lines of different colors (black, red, blue and purple). Performance of the uniform polynomial degree refinement (in green) is also included.

4. Conclusions

In this paper, we have extended the truncation error-based p-adaptation method to unsteady problems. First, we presented a new form of the truncation error, which holds close similarities to the formulation traditionally used in the literature for variational methods [13, 14, 62, 63]. The new form of the truncation error performs well and similarly to the traditional form for a test case with uniform mesh size. However, when considering a nonuniform mesh size, the new form outperforms the traditional formulation. Second, we extended the τ-estimation method to estimate the truncation error of unsteady flow problems with the DGSEM. The method developed here retains the anisotropic properties and the ability to be estimated in a multigrid cycle, as proposed by the authors in [63]. Third, we proposed two truncation error-based p-adaptation strategies: the dynamic and static adaptation methods. We analyzed both strategies and used them successfully to enhance the performance of DGSEM in the open-source framework HORSES3D [17]. We conclude that the static p-adaptation method performs better than the dynamic one in statistically steady problems where the flow features are concentrated in a small part of the domain. Similarly, the dynamic p-adaptation method outperforms the static one when the flow features move through a large portion of the domain. As the τ-estimation method relies on the exponential convergence of the numerical scheme, we have focused in laminar test cases without shocks (problems with smooth solutions). For these test cases, significant speed-ups (up to $\times 4$) are reported. Future work includes extending the methodology incorporate compressible test cases featuring shocks and turbulence.

Appendix A. A Note on Post-Processing

In this section, we provide a short description of the post-processing method used to acquire the results presented in Section 3.2. Since we carried out the calculations with a constant CFL (instead of a constant time-step size) and stored the lift and drag at every time step, we re-sample the lift and drag signals to obtain equispaced data in time. The process consists of three steps:

1. We take the last part of the signal to avoid the effect of any transients from the restart. For all simulations considered, the last 20 time units (of a total of 100) showed to have a periodic behavior.

2. We cut the signal from left and right to ensure that we are averaging over entire periods of the signal. First, we compute the mean value of the signal (lift or drag), locate the first position where this mean value appears in the time series, and remove data left from that point. Then, we locate the last position where this mean value appears in the time series with a slope of the same sign and remove data right from that point. The resulting signal has n points.

3. We feed the time series obtained in step 2 into the MATLAB function *resample* to get a new signal with $4 \times n$ equidistant points.

Disclosure Statement

The authors report there are no competing interests to declare.

Acknowledgments

GR, EV and EF acknowledge the funding received by the Grant DeepCFD (Project No. PID2022-137899OB-I00) funded by MCIN/AEI/10.13039/501100011033/ and by ERDF A way of making Europe. AR acknowledges funding through the Klaus-Tschira Stiftung via the project "HiFiLab". GR, EV and EF acknowledge the funding received by the Grant NextSim / AEI /10.13039/501100011033 and H2020, GA-956104. EF would like to thank the support of Agencia Estatal de Investigación (for the grant "Europa Excelencia 2022" Proyecto EUR2022-134041/AEI/10.13039/501100011033) y del Mecanismo de Recuperación y Resiliencia de la Unión Europea, and the Comunidad de Madrid and Universidad Politécnica

de Madrid for the Young Investigators award: APOYO-JOVENES-21-53NYUB-19-RRX1A0. Finally, all authors gratefully acknowledge Universidad Politécnica de Madrid (www.upm.es) for providing computing resources on Magerit Supercomputer.

References

[1] M. Wagner, Performance and scalability of the cfd solver coda (2022).
[2] B. Cockburn, G. E. Karniadakis, C.-W. Shu, The Development of Discontinuous Galerkin Methods, Discontinuous Galerkin Methods 11 (0) (2000) 3–50. doi:10.1007/978-3-642-59721-3_1.
URL http://dx.doi.org/10.1007/978-3-642-59721-3{_}1
[3] Z. Wang, K. Fidkowski, R. Abgrall, F. Bassi, D. Caraeni, A. Cary, H. Deconinck, R. Hartmann, K. Hillewaert, H. Huynh, N. Kroll, G. May, P.-O. Persson, B. van Leer, M. Visbal, High-order CFD methods: current status and perspective, International Journal for Numerical Methods in Fluids 72 (8) (2013) 811–845. doi:10.1002/fld.3767.
[4] E. Ferrer, R. H. Willden, A high order Discontinuous Galerkin - Fourier incompressible 3D Navier-Stokes solver with rotating sliding meshes, Journal of Computational Physics 231 (21) (2012) 7037–7056. doi:10.1016/j.jcp.2012.04.039.
URL http://dx.doi.org/10.1016/j.jcp.2012.04.039
[5] E. Ferrer, An interior penalty stabilised incompressible discontinuous Galerkin–Fourier solver for implicit large eddy simulations, Journal of Computational Physics 348 (2017) 754–775. doi:https://doi.org/10.1016/j.jcp.2017.07.049.
URL https://www.sciencedirect.com/science/article/pii/S0021999117305570
[6] E. Ferrer, A high order discontinuous Galerkin-Fourier incompressible 3d Navier-Stokes solver with rotating sliding meshes for simulating cross-flow turbines, Ph.D. thesis, Oxford University, UK (2012).
[7] K. Black, A conservative spectral element method for the approximation of compressible fluid flow, Kybernetika 35 (1) (1999) 133–146.
[8] D. A. Kopriva, Implementing spectral methods for partial differential equations: Algorithms for scientists and engineers, Springer Science & Business Media, 2009.
[9] J. S. Hesthaven, T. Warburton, Nodal discontinuous Galerkin methods: algorithms, analysis, and applications, Springer Science & Business Media, 2007.
[10] F. Hindenlang, G. J. Gassner, C. Altmann, A. Beck, M. Staudenmaier, C. D. Munz, Explicit discontinuous Galerkin methods for unsteady problems, Computers and Fluids 61 (2012) 86–93. doi:10.1016/j.compfluid.2012.03.006.
URL http://dx.doi.org/10.1016/j.compfluid.2012.03.006
[11] H. Ranocha, M. Schlottke-Lakemper, J. Chan, A. M. Rueda-Ramírez, A. R. Winters, F. Hindenlang, G. J. Gassner, Efficient implementation of modern entropy stable and kinetic energy preserving discontinuous Galerkin methods for conservation laws, arXiv preprint arXiv:2112.10517 (2021).
[12] A. D. Beck, D. G. Flad, C. Tonhäuser, G. Gassner, C. D. Munz, On the Influence of Polynomial De-aliasing on Subgrid Scale Models, Flow, Turbulence and Combustion 97 (2) (2016) 475–511. doi:10.1007/s10494-016-9704-y.
[13] M. Kompenhans, G. Rubio, E. Ferrer, E. Valero, Adaptation strategies for high order discontinuous Galerkin methods based on Tau-estimation, Journal of Computational Physics 306 (2016) 216–236. doi:10.1016/j.jcp.2015.11.032.
URL http://dx.doi.org/10.1016/j.jcp.2015.11.032
[14] A. M. Rueda-Ramírez, G. Rubio, E. Ferrer, E. Valero, An Anisotropic p-Adaptation Multigrid Scheme for the Discontinuous Galerkin Spectral Element Method, Lecture Notes in Computational Science and Engineering. Proceedings of the International Conference on Spectral and High-Order Methods (ICOSAHOM 2018). London, UK. (2019).
[15] J. Manzanero, G. Rubio, D. A. Kopriva, E. Ferrer, E. Valero, Entropy-stable discontinuous Galerkin approximation with summation-by-parts property for the incompressible Navier-Stokes equations with variable density and artificial compressibility (2019). arXiv:1902.08089.
URL http://arxiv.org/abs/1902.08089
[16] G. J. Gassner, A. R. Winters, F. J. Hindenlang, D. A. Kopriva, The BR1 Scheme is Stable for the Compressible Navier – Stokes Equations, Vol. 77, Springer US, 2018. doi:10.1007/s10915-018-0702-1.
URL https://doi.org/10.1007/s10915-018-0702-1
[17] E. Ferrer, G. Rubio, G. Ntoukas, W. Laskowski, O. Mariño, S. Colombo, A. Mateo-Gabín, H. Marbona, F. M. de Lara, D. Huergo, et al., HORSES3D: A high-order discontinuous Galerkin solver for flow simulations and multi-physics applications, Computer Physics Communications 287 (2023) 108700.
[18] J. Manzanero, G. Rubio, D. A. Kopriva, E. Ferrer, E. Valero, A free-energy stable nodal discontinuous Galerkin approximation with summation-by-parts property for the Cahn-Hilliard equation., arXiv Numerical Analysis (2019). arXiv:arXiv:1902.08089v1, doi:arXiv:1902.08089v1.
URL http://arxiv.org/abs/1902.08089
[19] M. Bohm, A. R. Winters, G. J. Gassner, D. Derigs, F. Hindenlang, J. Saur, An entropy stable nodal discontinuous Galerkin method for the resistive MHD equations. Part I: Theory and numerical verification, Journal of Computational Physics 1 (2018) 1–35. doi:10.1016/j.jcp.2018.06.027.
URL https://doi.org/10.1016/j.jcp.2018.06.027
[20] A. M. Rueda-Ramírez, F. J. Hindenlang, J. Chan, G. J. Gassner, Entropy-stable gauss collocation methods for ideal magneto-hydrodynamics, Journal of Computational Physics 475 (2023) 111851.
[21] F. Fraysse, E. Valero, J. Ponsín, Comparison of Mesh Adaptation Using the Adjoint Methodology and Truncation Error Estimates, AIAA Journal 50 (9) (2012) 1920–1932. doi:10.2514/1.J051450.

[22] M. Kompenhans, G. Rubio, E. Ferrer, E. Valero, Comparisons of p-adaptation strategies based on truncation- and discretisation-errors for high order discontinuous Galerkin methods, Computers and Fluids 139 (2016) 36–46. doi:10.1016/j.compfluid.2016.03.026.
URL http://dx.doi.org/10.1016/j.compfluid.2016.03.026
[23] F. Naddei, M. De La LLave Plata, V. Couaillier, F. Coquel, A comparison of refinement indicators for the p-adaptive simulation of steady and unsteady flows with discontinuous Galerkin methods, Journal of Computational Physics 376 (1 January 2019) (2018) 508–533. doi:https://doi.org/10.1016/j.jcp.2018.09.045.
[24] M. J. Aftosmis, Upwind method for simulation of viscous flow on adaptively refined meshes, AIAA Journal 32 (2) (1994) 268–277. doi:10.2514/3.11981.
URL http://arc.aiaa.org/doi/10.2514/3.11981
[25] P.-O. Persson, J. Peraire, Sub-Cell Shock Capturing for Discontinuous Galerkin Methods, 44th AIAA Aerospace Sciences Meeting and Exhibit (2006) 1–13doi:10.2514/6.2006-112.
URL http://arc.aiaa.org/doi/10.2514/6.2006-112
[26] L. Krivodonova, J. E. Flaherty, Error estimation for discontinuous Galerkin solutions of multidimensional hyperbolic problems, Advances in Computational Mathematics 19 (2003) 57–71. doi:10.1023/A:1022894504834.
[27] L. Krivodonova, J. Xin, J.-F. Remacle, N. Chevaugeon, J. E. Flaherty, Shock detection and limiting with discontinuous Galerkin methods for hyperbolic conservation laws, Applied Numerical Mathematics 48 (3-4) (2004) 323–338. doi:10.1016/j.apnum.2003.11.002.
URL http://linkinghub.elsevier.com/retrieve/pii/S0168927403001831
[28] J.-F. Remacle, J. E. Flaherty, M. S. Shephard, An Adaptive Discontinuous Galerkin Technique with an Orthogonal Basis Applied to Compressible Flow Problems, Society for Industrial and Applied Mathematics. SIAM Review 45 (1) (2003) 53–72.
[29] I. Babuška, A. Miller, The post-processing approach in the finite element method—part 1: Calculation of displacements, stresses and other higher derivatives of the displacements, International Journal for numerical methods in engineering 20 (6) (1984) 1085–1109.
[30] I. Babuška, A. Miller, The post-processing approach in the finite element method—Part 2: The calculation of stress intensity factors, International Journal for numerical methods in Engineering 20 (6) (1984) 1111–1129.
[31] R. Hartmann, Error estimation and adjoint-based adaptation in aerodynamics, European Conference on Computational Fluid Dynamics (2006) 1–14.
[32] R. Hartmann, P. Houston, Adaptive Discontinuous Galerkin Finite Element Methods for the Compressible Euler Equations, Journal of Computational Physics 183 (2) (2002) 508–532. doi:10.1006/jcph.2002.7206.
URL http://linkinghub.elsevier.com/retrieve/pii/S0021999102972062
[33] L. Wang, D. Mavriplis, Adjoint-based h-p Adaptive Discontinuous Galerkin Methods for the Compressible Euler Equations, Journal of Computational Physics 228 (20) (2009) 7643—-7661.
[34] C. Mavriplis, Nonconforming Discretizations and a Posteriori Error Estimators for Adaptive Spectral Element Techniques, Ph.D. thesis, Massachusetts Institute of Technology (1989).
URL http://hdl.handle.net/1721.1/14526
[35] C. Mavriplis, Adaptive mesh strategies for the spectral element method, Computer methods in applied mechanics and engineering 116 (1-4) (1994) 77–86.
URL http://www.sciencedirect.com/science/article/pii/S0045782594800103
[36] G. Kuru, M. De la Llave Plata, An adaptive variational multiscale discontinuous Galerkin method for large eddy simulation, 54th AIAA Aerospace Sciences Meeting (2016) p. 0584.
[37] B. Cockburn, C.-W. Shu, Tvb runge-kutta local projection discontinuous galerkin finite element method for conservation laws. ii. general framework, Mathematics of computation 52 (186) (1989) 411–435.
[38] J. Zhu, J. Qiu, Hermite weno schemes and their application as limiters for runge-kutta discontinuous galerkin method, iii: Unstructured meshes, Journal of Scientific Computing 39 (2) (2009) 293–321.
[39] L. Krivodonova, J. Xin, J.-F. Remacle, N. Chevaugeon, J. E. Flaherty, Shock detection and limiting with discontinuous galerkin methods for hyperbolic conservation laws, Applied Numerical Mathematics 48 (3-4) (2004) 323–338.
[40] G. Fu, C.-W. Shu, A new troubled-cell indicator for discontinuous galerkin methods for hyperbolic conservation laws, Journal of Computational Physics 347 (2017) 305–327.
[41] S. Diot, R. Loubère, S. Clain, The multidimensional optimal order detection method in the three-dimensional case: very high-order finite volume method for hyperbolic systems, International Journal for Numerical Methods in Fluids 73 (4) (2013) 362–392.
[42] W. Boscheri, G. Dimarco, High order modal discontinuous galerkin implicit–explicit runge kutta and linear multistep schemes for the boltzmann model on general polygonal meshes, Computers & Fluids 233 (2022) 105224.
[43] M. Dumbser, O. Zanotti, R. Loubère, S. Diot, A posteriori subcell limiting of the discontinuous galerkin finite element method for hyperbolic conservation laws, Journal of Computational Physics 278 (2014) 47–75.
[44] F. Vilar, A posteriori correction of high-order discontinuous galerkin scheme through subcell finite volume formulation and flux reconstruction, Journal of Computational Physics 387 (2019) 245–279.
[45] V. Maltsev, D. Yuan, K. W. Jenkins, M. Skote, P. Tsoutsanis, Hybrid discontinuous galerkin-finite volume techniques for compressible flows on unstructured meshes, Journal of Computational Physics 473 (2023) 111755.
[46] G. J. Gassner, A. R. Winters, A novel robust strategy for discontinuous galerkin methods in computational fluid mechanics: Why? when? what? where?, Frontiers in Physics 8 (2021) 500690.
[47] D. Flad, A. Beck, C.-D. Munz, Simulation of underresolved turbulent flows by adaptive filtering using the high order discontinuous galerkin spectral element method, Journal of Computational Physics 313 (2016) 1–12.

[48] M. Hamedi, B. C. Vermeire, Optimized filters for stabilizing high-order large eddy simulation, Computers & Fluids 237 (2022) 105301.
[49] A. R. Winters, R. C. Moura, G. Mengaldo, G. J. Gassner, S. Walch, J. Peiro, S. J. Sherwin, A comparative study on polynomial dealiasing and split form discontinuous galerkin schemes for under-resolved turbulence computations, Journal of Computational Physics 372 (2018) 1–21.
[50] T. Dzanic, W. Trojak, F. D. Witherden, On the anti-aliasing properties of entropy filtering for under-resolved turbulent flows, arXiv preprint arXiv:2302.13359 (2023).
[51] C. Roy, Review of Discretization Error Estimators in Scientific Computing, in: 48th AIAA Aerospace Sciences Meeting Including the New Horizons Forum and Aerospace Exposition, American Institute of Aeronautics and Astronautics, Reston, Virigina, 2010. doi:10.2514/6.2010-126.
URL http://arc.aiaa.org/doi/10.2514/6.2010-126
[52] A. Choudhary, C. J. Roy, Structured Mesh r-Refinement using Truncation Error Equidistribution for 1D and 2D Euler Problems, in: 21st AIAA Computational Fluid Dynamics Conference., American Institute of Aeronautics and Astronautics, Reston, Virginia, 2013, p. 2444. doi:10.2514/6.2013-2444.
[53] A. Syrakos, G. Efthimiou, J. G. Bartzis, A. Goulas, Numerical experiments on the efficiency of local grid refinement based on truncation error estimates, Journal of Computational Physics 231 (20) (2012) 6725–6753. arXiv:1508.02345, doi:10.1016/j.jcp.2012.06.023.
[54] G. Rubio, F. Fraysse, D. A. Kopriva, E. Valero, Quasi-a priori truncation error estimation in the DGSEM, Journal of Scientific Computing 64 (2) (2015) 425–455. doi:10.1007/s10915-014-9938-6.
[55] W. Laskowski, G. Rubio, E. Valero, E. Ferrer, A functional oriented truncation error adaptation method., J. Comput. Phys. 451 (2022) 110883.
[56] A. Brandt, O. E. Livne, Multigrid Techniques: 1984 Guide with Applications to Fluid Dynamics, Revised Edition, SIAM, 2011. doi:10.1137/1.9781611970753.
URL http://epubs.siam.org/doi/book/10.1137/1.9781611970753
[57] M. J. Berger, Adaptive finite difference methods in fluid dynamics, in: In Von Karman Inst. for Fluid Dynamics, Computational Fluid Dynamics 50 p (SEE N88-15951 08-34), 1987, pp. 08–34.
[58] F. Fraysse, G. Rubio, J. De Vicente, E. Valero, Quasi-a priori mesh adaptation and extrapolation to higher order using τ-estimation, Aerospace Science and Technology 38 (2014) 76–87. doi:10.1016/j.ast.2014.07.017.
[59] F. Fraysse, E. Valero, G. Rubio, Quasi-a priori truncation error estimation and higher order extrapolation for non-linear partial differential equations, Journal of Computational Physics 253 (2013) 389–404. doi:10.1016/j.jcp.2013.07.018.
[60] A. Syrakos, A. Goulas, Finite volume adaptive solutions using SIMPLE as smoother, International Journal for Numerical Methods in Fluids 52 (11) (2006) 1215–1245. doi:10.1002/fld.1228.
[61] G. Rubio, F. Fraysse, J. De Vicente, E. Valero, The estimation of truncation error by τ-estimation for Chebyshev spectral collocation method, Journal of Scientific Computing 57 (1) (2013) 146–173. doi:10.1007/s10915-013-9698-8.
[62] A. M. Rueda-Ramírez, G. Rubio, E. Ferrer, E. Valero, Truncation Error Estimation in the p-Anisotropic Discontinuous Galerkin Spectral Element Method, Journal of Scientific Computing 78 (1) (2019) 433–466. doi:10.1007/s10915-018-0772-0.
[63] A. M. Rueda-Ramírez, J. Manzanero, E. Ferrer, G. Rubio, E. Valero, A p-multigrid strategy with anisotropic p-adaptation based on truncation errors for high-order discontinuous Galerkin methods, Journal of Computational Physics 378 (2019) 209–233. doi:10.1016/j.jcp.2018.11.009.
[64] D. N. Arnold, F. Brezzi, B. Cockburn, D. Marini, Unified analysis of discontinuous Galerkin methods for elliptic problems, SIAM J. Numer. Anal. 39 (5) (2002) 1749–1779.
[65] G. J. Gassner, Discontinuous Galerkin methods for the unsteady compressible Navier-Stokes equations, Ph.D. thesis, University of Stuttgart (2009). doi:10.18419/opus-3788.
[66] D. A. Kopriva, F. J. Hindenlang, T. Bolemann, G. J. Gassner, Free-stream preservation for curved geometrically non-conforming discontinuous galerkin spectral elements, Journal of Scientific Computing 79 (2019) 1389–1408.
[67] D. A. Kopriva, S. L. Woodruff, M. Y. Hussaini, Computation of electromagnetic scattering with a non-conforming discontinuous spectral element method, International Journal for Numerical Methods in Engineering 53 (1) (2002) 105–122. doi:10.1002/nme.394.
[68] C. Roy, Strategies for driving mesh adaptation in cfd, in: 47th AIAA aerospace sciences meeting including the new horizons forum and aerospace exposition, 2009, p. 1302.
[69] W. L. Oberkampf, C. J. Roy, Verification and validation in scientific computing, Cambridge University Press, 2010.
[70] W. C. Tyson, G. K. Yan, C. J. Roy, C. F. Ollivier-Gooch, Relinearization of the error transport equations for arbitrarily high-order error estimates, Journal of Computational Physics 397 (2019) 108867.
[71] S. Blaise, A. St-Cyr, A Dynamic hp -Adaptive Discontinuous Galerkin Method for Shallow-Water Flows on the Sphere with Application to a Global Tsunami Simulation , Monthly Weather Review 140 (3) (2012) 978–996. doi:10.1175/mwr-d-11-00038.1.
[72] J. S. Cagnone, S. K. Nadarajah, A stable interface element scheme for the p-adaptive lifting collocation penalty formulation, Journal of Computational Physics 231 (4) (2012) 1615–1634. doi:10.1016/j.jcp.2011.10.018.
URL http://dx.doi.org/10.1016/j.jcp.2011.10.018
[73] K. J. Fidkowski, Output error estimation strategies for discontinuous Galerkin discretizations of unsteady convection-dominated flow, International Journal for Numerical Methods in Engineering 88 (2011) 1297–1322.
[74] P. L. Roe, Approximate Riemann solvers, parameter vectors, and difference schemes, Journal of Computational Physics 43 (2) (1981) 357–372. arXiv:arXiv:1011.1669v3, doi:10.1016/0021-9991(81)90128-5.
[75] F. Bassi, F. Rebay, A high-order accurate discontinuous finite element method for the numerical solution of the compressible

Navier-Stokes equations, Journal of Computational Physics 131 (1997) 267–279. doi:http://dx.doi.org/10.1006/jcph.1996.5572.
URL http://isn-csm.mit.edu/literature/1997-jcp-bassi.pdf
[76] J. H. Williamson, Low-storage Runge-Kutta schemes, Journal of Computational Physics 35 (1) (1980) 48–56.