Contents lists available at ScienceDirect

Journal of Computational Physics

journal homepage: www.elsevier.com/locate/jcp

A functional oriented truncation error adaptation method

Wojciech Laskowski[a,*], Gonzalo Rubio[a,b], Eusebio Valero[a,b], Esteban Ferrer[a,b]

[a]*ETSIAE-UPM (School of Aeronautics - Universidad Politécnica de Madrid) - Plaza de Cardenal Cisneros 3, 28040 Madrid, Spain*
[b]*Center for Computational Simulation - Universidad Politécnica de Madrid, Campus de Montegancedo, Boadilla del Monte, 28660 Madrid, Spain*

ARTICLE INFO

Article history:

ABSTRACT

1. Introduction

Mesh adaptation allows one to localise the numerical degrees of freedom in mesh regions that require increased accuracy. This is particularly important in Computational Fluid Dynamics (CFD) where millions of degrees of freedom can be required. Mesh adaptation is considered a pillar of the future developments in CFD [1]. Adaptation can be performed by moving the nodes (r–adaptation), subdividing or merging elements (h–adaptation), or by enriching or reducing the polynomial order (p–adaptation). Mesh adaptation algorithms need a sensor to identify the flow regions that require refinement or coarsening. Three classical approaches can be found in the literature: feature–based, local error–based, and adjoint–based [2]. Feature–based adaptation refines regions where large gradients in the numerical solution are found. This approach requires low–cost sensors, which do not have an explicit connection with the error in the solution and thus often results in inefficiently adapted meshes, in which the final accuracy cannot be predicted beforehand [3]. Adjoint–based adaptation requires the selection of a particular functional (e.g. energy, drag or lift) to generate an optimal mesh that reduces the error associated to that functional. This approach is efficient in reducing the selected functional error, as it targets the source of the functional error, but it cannot ensure the reduction of other functional errors. For example, Fidkowski and Roe [4] have shown that the adjoint–based adaptation (targeting lift or drag) does not decrease the error in another functional (in particular, wake rake). Additionally, adjoint–based adaptation has a high cost since an adjoint system (of the size of the original system) needs to be solved (and stored in memory) [5]. Local error–based adaptation includes the truncation error adaptation method, which aims to equi-distribute the local error in the mesh domain. For advection–dominated problems, truncation error–based sensors localise the elements where the discretisation error is generated, such that the degrees of freedom can be efficiently located at the

*Corresponding author: wj.laskowski@upm.es (Wojciech Laskowski)

source, and not misused in elements polluted by the advected error [6]. This method does not require the solution of additional systems and is cheap. Examples can be found for low-order [7, 8, 9] and high-order methods [10, 11, 12]. The advantages of truncation error control have encouraged the development of h-adaptation [13, 14, 15, 16, 17, 18] and p-adaptation [19, 20, 21] techniques. Truncation error–based adaptation has been shown to control all functional errors [19]. Examples can be seen in [14], where truncation error–based h–adaptation is used to decrease the error in lift, drag, and moment coefficients or in [20], where truncation error–based p–adaptation is shown to decrease the error in lift and drag.

However, truncation error adaptation techniques have a substantial drawback: the user–given error threshold that controls the refinement/coarsening of the cells has no engineering or physical meaning, and consequently the user needs to calibrate the truncation error thresholds to improve the mesh and reduce functional errors. In this short note, we explain how to select the truncation error threshold by relating the truncation error to the functional error. In doing so, we derive a truncation error–based adaptation technique with the advantages of adjoints, i.e., we can target a specific functional error. The cost of this approach is higher than traditional truncation error adaptation, but lower than adjoint–based adaptation.

First, we define the essential error bounds in Section 2. Second, we detail the truncation error functional–based approach in Section 3. Third, we present a test case to show the applicability and computational cost of this adaptation methodology in Section 4. Conclusions are included in Section 5.

2. Foundations

Let us consider a continuous partial differential equation (PDE) and its semi-discretisation in space:

$$\frac{du}{dt} + F(u) = 0, \qquad M_h^N \frac{du_h^N}{dt} + F_h^N(u_h^N) = 0. \tag{1}$$

The subindex h denotes the characteristic mesh element size, while the superindex N relates to the order of convergence of the spatial discretisation scheme. Finite element or high-order discontinuous Galerkin discretisations use a mass matrix in (1), however this definition (and all the derivations that follow) can be extended to finite difference or finite volume discretisations, in which the mass matrix becomes $M_h^N = I_h^N$ (where I_h^N is the discrete identity) or $M_h^N = V_h^N$ (where V_h^N is a diagonal matrix with the mesh element volumes in the diagonal). We denote by u and u_h^N as the steady state solutions to the continuous and discrete equations. We define the discrete spatial partial differential operator using the mass matrix weighted flux: $R_h^N(\cdot) = \left(M_h^N\right)^{-1} F_h^N(\cdot)$. We follow [6] to define the truncation error τ as:

$$R_h^N(u) = \tau_h^N, \tag{2}$$

and although not strictly necessary, it is helpful to introduce also the discretisation error, ϵ_h^N, as the difference between the exact solution of the problem, u, and the approximate solution, u_h^N. Roy [6] shows that both errors are connected through the Discretisation Error Transport Equation (DETE) equation.

To relate the truncation error and the discretisation error to the functional errors (for a functional $J_h^N(\cdot)$), we need to use adjoint theory. This was detailed in our previous work [19], where we showed that the truncation error controls

the discretisation error and all functional errors following the inequality:

$$\|\tau_h^N\| \gtrsim \frac{\|\epsilon_h^N\|}{\left\|\left(\left.\frac{\partial R_h^N}{\partial u_h^N}\right|_{u_h^N}\right)^{-1}\right\|} \gtrsim \frac{\|J_h^N(u) - J_h^N\left(u_h^N\right)\|}{\left\|\left(\left.\frac{\partial R_h^N}{\partial u_h^N}\right|_{u_h^N}\right)^{-1}\right\| \left\|\left.\frac{\partial J_h^N}{\partial u_h^N}\right|_{u_h^N}\right\|}. \tag{3}$$

The derivation in [19] made use of the L_∞ norm, however (3) holds for any matrix induced norm. Here, $\|u\| = \left(\sum_i u_i^2\right)^{1/2}$ represents the Euclidean L_2 norm. In this work, the inequality (3) is exploited to relate the truncation error to the functional error. Note that the method is suitable for sufficiently smooth solutions (asymptotic range), since inequality (3) is based on the asymptotic decay rate of errors.

3. Selection of τ threshold based on a target functional error

Here, we present a 3–step technique to define the steady truncation error threshold in a p–adaptation algorithm.

- Step 1: The user selects the target functional error $\|J_h^N(u) - J_h^N\left(u_h^N\right)\|$.
- Step 2: We scale the target functional error by the norm of the inverse of the Jacobian matrix of the system $\left\|\left(\left.\frac{\partial R_h^N}{\partial u_h^N}\right|_{u_h^N}\right)^{-1}\right\|$ and by the norm of the functional derivative $\left\|\left.\frac{\partial J_h^N}{\partial u_h^N}\right|_{u_h^N}\right\|$. The scaling is performed for every candidate mesh (with polynomial order N and characteristic mesh element size h).
- Step 3: Using the relation between the errors, inequality (3), we compute the maximum allowed value for $\|\tau_h^N\|$, that fulfills the targeted functional error defined in Step 1.

Computation of the norm of the functional derivative in Step 2 is cheap, but computing the inverse of the Jacobian is expensive. In our approach, we do not need to invert the Jacobian, but only require its L_2 norm: $\left\|\left(\left.\frac{\partial R_h^N}{\partial u_h^N}\right|_{u_h^N}\right)^{-1}\right\| = \left(\sigma_{min}\left(\left.\frac{\partial R_h^N}{\partial u_h^N}\right|_{u_h^N}\right)\right)^{-1}$, where $\sigma_{min}\left(\left.\frac{\partial R_h^N}{\partial u_h^N}\right|_{u_h^N}\right)$ represents the minimum singular value of the Jacobian matrix of the system. As discussed in [22], this minimum singular value is almost mesh independent, and therefore can be estimated for a coarse mesh and used for all finer meshes in Step 2. We verify the mesh independence of σ_{min} for a particular test case in Table 1. Additionally, we include the computational cost to obtain σ_{min} in each grid to quantify the computational cost savings of the proposed coarse grid estimation.

4. Test case

We demonstrate the technique on a three-dimensional test case with a manufactured solution of the compressible Navier-Stokes equations [23] with Reynolds number (with characteristic length one) Re = 1000 and Mach number Ma = 0.1:

$$\rho = p = \frac{e^x + e^y + e^z}{10} + 1, \quad u_x = u_y = u_z = 1, \tag{4}$$

where ρ and p are density and pressure and u_i are the velocity components. With the chosen solution, one then can derive a balancing source term [23]. We discretise the equations using a high-order nodal Discontinuous Galerkin

Spectral Element Method (DGSEM), where the physical domain, $\Omega = [0,1]^3$, is tessellated into non-overlapping hexahedral elements including polynomials of degree N [24].

Two uniform Cartesian grids with a total number of elements of 8 and 64, in combination with four polynomial orders, $N = 2, ..., 5$, are used to discretise the domain. Without loss of generality, we define, as the target functional, the total kinetic energy in our computational domain: $E_k = \frac{1}{2}\int_\Omega \rho(u_x^2 + u_y^2 + u_z^2)\, d\Omega$.

4.1. 3–step τ–based adaptation example

Figure 1 shows the normalised relation between the discretisation error, truncation error, and the functional error against the number of degrees of freedom (*DOF*), for the 8-element grid (Figure 1a) and for the 64-element grid (Figure 1b). Additionally, Figure 1 is used to illustrate the adaptation methodology proposed in Section 3. We can follow the 3–step methodology to obtain the truncation error threshold.

- Step 1: We select the target functional error, for example, 10^{-2} error in kinetic energy: $\|J_h^N(u) - J_h^N\left(u_h^N\right)\| = 10^{-2}$, point **[1]** in Figures 1a and 1b.

- Step 2: We scale the functional error by the norm of the functional derivative and the minimum singular value σ_{min} from the Jacobian: $10^{-2} \times \frac{\sigma_{min}}{\left\|\partial J_h^N/\partial u_h^N\big|_{u_h^N}\right\|}$, where σ_{min} is computed for the coarsest mesh (i.e. 8-element and polynomial $N = 2$ mesh). Then, for every candidate mesh, $N = 2, ..., 5$, we compute the scaling, which shows minor variations due to the norm of the functional derivative, blue lines in Figures 1a and 1b.

- Step 3: Based on (3), we compare $\|\tau_h^N\|$ to the estimation of Step 2 (for every refinement level). The coarsest mesh that fulfills $\|\tau_h^N\| < 10^{-2} \times \frac{\sigma_{min}}{\left\|\partial J_h^N/\partial u_h^N\big|_{u_h^N}\right\|}$ is retained since it fits the imposed threshold. This leads to a polynomial order $N = 5$ in both cases, point **[2]** in Figures 1a and 1b.

The new adapted meshes ($N = 5$) have functional errors below the 10^{-2} error target in kinetic energy. The resulting number of degrees of freedom is $DOF = 8640$ for the 8-element case (see Figure 1a) and $DOF = 69120$ for the 64-element case (see Figure 1b).

We also include a simple, functional error–based, p-uniform mesh adaptation example. For the functional error–based adaptation, we follow the *discrete adjoint method* [6], where the target functional error can be approximated as a product of discrete adjoint sensitivities Ψ and truncation error defined as in (2):

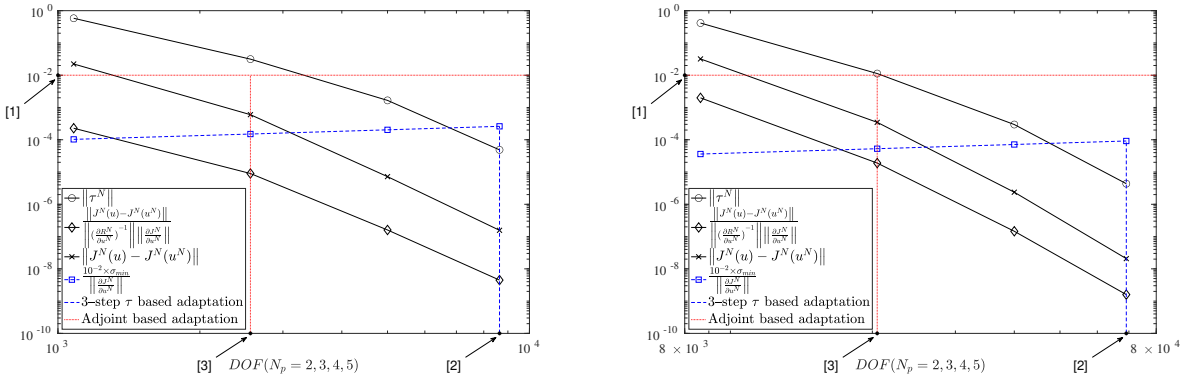
$$\|J_h^N(u) - J_h^N\left(u_h^N\right)\| \approx \|\Psi^T R_h^N(u)\|. \tag{5}$$

Discrete adjoint sensitivities are obtained by solving the following linear system of equations:

$$\left(\frac{\partial R_h^N}{\partial u_h^N}\bigg|_{u_h^N}\right)^T \Psi = \left(\frac{\partial J_h^N}{\partial u_h^N}\bigg|_{u_h^N}\right)^T. \tag{6}$$

As for the 3–step truncation error–based procedure, this system of equations must be solved for every candidate mesh. The coarsest mesh that fulfills $\|\Psi^T R_h^N(u)\| < 10^{-2}$ is retained since it fits the imposed threshold. The computed approximation of the functional error $\|\Psi^T R_h^N(u)\|$ is not represented in Figure 1, as it overlaps the functional error

curve $\|J_h^N(u) - J_h^N\left(u_h^N\right)\|$ (see (5)). This adaptation approach results in $DOF = 2560$ for the 8-element mesh and $DOF = 20480$ for the 64-element mesh (see red lines and point **[3]** in Figure 1a and 1b, respectively). The new truncation error–based procedure results in meshes with more degrees of freedom ($DOF_{\text{3–step}}/DOF_{\text{functional}} = 3.375$ for the two meshes) than when using only one selected functional error (i.e. an adjoint method), but this is the price to pay for not solving an adjoint system (Equation (6)). In addition, the truncation error controls all possible functional errors (and not only the selected one) when adapting.



(a) 8-element grid

(b) 64-element grid

Fig. 1: Numerical results of the relation between truncation error and functional errors (see inequality (3)) against degrees of freedom based on increasing polynomial order. The points (circles, crosses and diamonds) represent meshes with polynomial orders $N = 2, 3, 4, 5$, while continuous lines connecting them are shown for convenience. The number of degrees of freedom is defined as $DOF = N_{eq} \cdot N_{el} \cdot (N_p + 1)^3$, where the variables denote: number of equations, number of elements, and polynomial order (uniform in all spatial directions), respectively. The 3-step τ–based adaptation technique is also visualised in the plot (blue lines). Additionally, an adjoint functional error–based adaptation is shown (red lines). Points [1], [2], [3] denote crucial stages in the adaptation stages, as explained in Section 4.1.

4.2. Computational cost

In this section, we compare the computational cost of the adjoint–based adaptation and the 3–step τ–based adaptation examples of the previous section. For all computations we use MATLAB [25] on an 8-core Intel Core i7-8550U @1.80GHz. We concentrate on the critical steps of each methodology: for the 3–step τ–based adaptation we check the L_2 norm estimation of the inverse of the Jacobian, while for the adjoint–based adaptation we examine the linear system of equations arising from the discrete adjoint problem (6).

4.2.1. Cost of estimating σ_{min}

As explained in Section 3, we estimate the L_2 norm of the Jacobian inverse by computing the smallest singular value, σ_{min}, which is almost constant for all polynomials and meshes considered, see Table 1. Hence, for the 3–step τ–based adaptation, we only need to compute σ_{min} once in the coarsest, 8-element mesh with polynomial $N = 2$. This value can then be retained for all calculations (all meshes and polynomials) and consequently a very significant saving in computational cost is obtained. Note that several estimations for σ_{min} are available in the literature and include: singular value decomposition, Krylov-subspace based [26] and analytical bounds [27]. Among all available techniques, here we select the LOBPCG for its efficiency and generality (i.e. all types of matrices) [28, 29] to compute σ_{min}. We use element-block additive Schwarz method with one overlapping node [30] as a preconditioner. This is an

an extension of the widely popular element block Jacobi for high-order discretisations [31]. Based on a preliminary convergence study, the tolerance of the iterative method is set to $tol_{solver} = 10^{-2}$. Decreasing the tolerance to lower levels does not have any influence on the accuracy of the adaptation methods considered in this work, as shown in Appendix A.

Table 1: Comparison of smallest singular value of the Jacobian $\frac{\partial R_h^N}{\partial u_h^N}$ for all the grids presented in Section 4 and associated computational cost. The LOBPCG method with a tolerance of 10^{-2} is used for σ_{min} computation. The cost includes preconditioner factorisation and Hermitian operator assembling $\left(\frac{\partial R_h^N}{\partial u_h^N}\right)^T \frac{\partial R_h^N}{\partial u_h^N}$. Bold values highlight the σ_{min} value and cost retained for the 3–step τ–based adaptation method.

	σ_{min}		Computational cost [s] for σ_{min}	
N	$N_{el} = 8$	$N_{el} = 64$	$N_{el} = 8$	$N_{el} = 64$
2	**0.11**	0.14	**0.88**	2.82
3	0.11	0.13	1.13	8.78
4	0.12	0.14	3.26	29.67
5	0.12	0.14	6.91	98.11

4.2.2. Cost comparison with adjoint–based adaptations

The adjoint problem, represented by Equation (6), is solved using the GMRES [32] method. We use the same preconditioner (element-block additive Schwarz method with one overlapping node) and tolerance (10^{-2}) as for the LOBPCG. Additionally, the small size of the problem allows us to include (for completeness) the results obtained by a direct solver (LU factorisation) for solving the discrete adjoint problem. Table 2 presents the total computational cost of the 3–step τ–based adaptation and the adjoint–based adaptation computed on all available grids and polynomial orders. The 3–step τ–based adaptation includes the computation of σ_{min} on the coarsest grid (8-element $N = 2$ grid in the example presented in this work) and the computation of the norm of the functional derivative for every candidate mesh (see Step 2). The adjoint–based adaptation approach requires solving the linear system (6) and evaluating the dot product in (5) for every candidate mesh. We first compare the LU and GMRES approaches for the adjoint–based adaptation. As expected, the LU method beats the GMRES for small problems; however, the scaling of the LU method with the size of the problem is poor, precluding its use in large applications. The 3–step τ–based adaptation presents a computational cost, which remains low and almost constant with the problem size. This is due to the fact that the most expensive part of the methodology (the computation of σ_{min}) is always performed in the coarsest mesh.

Independently of the method selected for adaptation, one needs to estimate the functional error in a set of meshes (from coarse to fine) until the imposed functional error is met. Therefore, if the overall added cost is taken into account, our cheap 3–step τ–estimation method leads to a very significant saving, when compared to the adjoint method. For example, in the test of Figure 1 the adjoint system for $N = 2$ and $N = 3$ must be solved for both, 8-element and 64-element case, while $N = 4$ and $N = 5$ can be skipped to save computational cost, as the prescribed tolerance is already met for $N = 3$. This results in a total added cost ($N = 2$ and $N = 3$ cases) of 1.48 s for GMRES (0.35 s for LU) for the 8-element case and 9.87 s for GMRES (15.39 s for LU) for the 64-element case. When using the 3–step τ–based adaptation, σ_{min} is only computed once in the coarsest mesh, which has a cost of 0.88 s,

Table 2: Computational cost (in seconds) for the 3–step τ–based and adjoint–based adaptation. The cost includes preconditioner factorisation for iterative methods and Hermitian operator assembling (for LOBPCG). A tolerance of 10^{-2} is set for all iterative methods.

	Computational cost [s]		
	3–step τ–based	Adjoint–based (LU)	Adjoint–based (GMRES)
N	$N_{el} = 8$		
2	0.89	0.11	0.57
3	0.89	0.24	0.91
4	0.90	1.25	3.25
5	0.90	6.36	8.63
N	$N_{el} = 64$		
2	0.91	1.80	1.86
3	0.94	13.59	8.01
4	0.96	86.81	25.12
5	1.01	361.27	82.16

see Table 1. In the test of Figure 1, the 3–step τ–based adaptation procedure is performed for $N = 2, 3, 4, 5$ where the prescribed tolerance is met. This results in a total added cost of 0.94 s for the 8-element case and 1.18 s for the 64-element case. For the test cases considered, the 3–step τ–based adaptation is always faster than the GMRES adjoint–based adaptation. Additionally, Tables 1 and 2 shows that the scalability of the computation of σ_{min} using LOBPCG is similar to the one of solving the adjoint system using GMRES. Therefore, the 3–step τ–based adaptation should still be faster than the adjoint–based adaptation for problems larger than the one considered here. Finally, for the 8-element case, the adjoint–based adaptation is faster than the 3–step τ–based adaptation if the LU method is used. This result only holds for very small problems, where direct methods can be more efficient than iterative methods.

5. Conclusions

This note provides closure to an open issue in the literature of truncation error–based adaptation, which is how to select an appropriate threshold for the truncation error. Relating the truncation threshold to functional errors, we are able to provide physical meaning (functional error) to the truncation error threshold. To do so, we have used the inequality derived in [19], which shows that the truncation error controls all functional errors. The proposed algorithm converts a target functional error threshold (e.g., energy, drag, or lift) into a truncation error threshold that can be used for adaptation. This approach enables a cheap, goal-oriented 3-step truncation error adaptation technique.

We have demonstrated the efficiency of the algorithm in the context of a p-adaptation method, for a three-dimensional compressible Navier-Stokes test case, discretised using a discontinuous Galerkin method. Let us note that the strategy presented is general and can be applied to any numerical technique (e.g., finite volumes using classical h-adaptation).

Acknowledgements

Wojciech Laskowski and Esteban Ferrer would like to thank the European Union's Horizon 2020 Research and Innovation Program under the Marie Skłodowska-Curie grant agreement No 813605 for the ASIMIA ITN-EID project.

Gonzalo Rubio and Eusebio Valero acknowledge the funding received by the grant SIMOPAIR (Project No. REF: RTI2018-097075-B-I00) funded by MCIN/AEI/ 10.13039/501100011033 and by ERDF A way of making Europe. Finally, the authors gratefully acknowledge the Universidad Politécnica de Madrid (www.upm.es) for providing computing resources on Magerit Supercomputer.

References

[1] J. Slotnick, A. Khodadoust, J. Alonso, D. Darmofal, W. Gropp, E. Lurie, D. Mavriplis, CFD vision 2030 study: a path to revolutionary computational aerosciences (2014).
[2] C. Roy, Strategies for driving mesh adaptation in CFD (invited), in: 47th AIAA aerospace sciences meeting including the new horizons forum and aerospace exposition, 2009, p. 1302.
[3] R. P. Dwight, Heuristic a posteriori estimation of error due to dissipation in finite volume schemes and application to mesh adaptation, Journal of Computational Physics 227 (2008) 2845–2863.
[4] K. Fidkowski, P. Roe, Entropy-based mesh refinement, I: the entropy adjoint approach, in: 19th AIAA Computational Fluid Dynamics, 2009, p. 3790.
[5] D. A. Venditti, D. L. Darmofal, Adjoint error estimation and grid adaptation for functional outputs: Application to quasi-one-dimensional flow, Journal of Computational Physics 164 (2000) 204–227.
[6] C. Roy, Review of discretization error estimators in scientific computing, in: 48th AIAA Aerospace Sciences Meeting Including the New Horizons Forum and Aerospace Exposition, 2010. doi:10.2514/6.2010-126.
[7] A. Syrakos, A. Goulas, Estimate of the truncation error of finite volume discretization of the Navier–Stokes equations on colocated grids, International journal for numerical methods in fluids 50 (2006) 103–130.
[8] F. Fraysse, J. de Vicente, E. Valero, The estimation of truncation error by τ-estimation revisited, Journal of Computational Physics 231 (2012) 3457–3482.
[9] 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.
[10] 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 (2013) 146–173.
[11] G. Rubio, F. Fraysse, D. A. Kopriva, E. Valero, Quasi-a priori truncation error estimation in the DGSEM, Journal of Scientific Computing (2014) 1–31.
[12] 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 (2019) 433–466.
[13] 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 (2012) 6725–6753.
[14] F. Fraysse, E. Valero, J. Ponsín, Comparison of mesh adaptation using the adjoint methodology and truncation error estimates, AIAA journal 50 (2012) 1920–1932.
[15] 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.
[16] J. Ponsin, F. Fraysse, M. Gómez, M. Cordero-Gracia, An adjoint-truncation error based approach for goal-oriented mesh adaptation, Aerospace Science and Technology 41 (2015) 229–240.
[17] C. W. Jackson, C. J. Roy, Performance of r-adaptation using truncation error-based equidistribution, Journal of Verification, Validation and Uncertainty Quantification 4 (2019).
[18] C. W. Jackson, C. J. Roy, C. R. Schrock, Truncation error based mesh optimization, Journal of Verification, Validation and Uncertainty Quantification 5 (2020).
[19] M. Kompenhans, G. Rubio, E. Ferrer, and E. Valero, Adaptation strategies for high order discontinuous Galerkin methods based on tau-estimation, Journal of Computational Physics 306 (2016) 216 – 236.
[20] 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.
[21] 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.
[22] K. Rao, P. Malan, J. B. Perot, A stopping criterion for the iterative solution of partial differential equations, Journal of Computational Physics 352 (2018) 265–284.
[23] W. Laskowski, A. M. Rueda-Ramírez, G. Rubio, E. Valero, E. Ferrer, Advantages of static condensation in implicit compressible Navier–Stokes DGSEM solvers, Computers & Fluids 209 (2020) 104646.
[24] D. A. Kopriva, Implementing spectral methods for partial differential equations: algorithms for scientists and engineers, 1st ed., Springer, 2009.
[25] MATLAB version 9.7.0.1471314 (R2019b), The Mathworks, Inc., Natick, Massachusetts, 2017.
[26] H. Avron, A. Druinsky, S. Toledo, Spectral condition-number estimation of large sparse matrices, Numerical Linear Algebra with Applications 26 (2019) e2235.
[27] N. Morača, Bounds for norms of the matrix inverse and the smallest singular value, Linear algebra and its applications 429 (2008) 2589–2601.
[28] A. V. Knyazev, Toward the optimal preconditioned eigensolver: Locally optimal block preconditioned conjugate gradient method, SIAM Journal on Scientific Computing 23 (2001) 517–541.
[29] Y. Saad, Numerical methods for large eigenvalue problems: revised edition, SIAM, 2011.
[30] J. Lottes, P. Fischer, Hybrid Multigrid/Schwarz algorithms for the spectral element method, J. Sci. Comput. 24 (2005) 45–78.

[31] K. J. Fidkowski, T. A. Oliver, J. Lu, D. L. Darmofal, p-Multigrid solution of high-order discontinuous Galerkin discretizations of the compressible Navier-Stokes equations, Journal of Computational Physics 207 (2005) 92–113.
[32] Y. Saad, M. H. Schultz, Gmres: A generalized minimal residual algorithm for solving nonsymmetric linear systems, SIAM Journal on scientific and statistical computing 7 (1986) 856–869.

Appendix A. Effect of the tolerance in the iterative solver

In this section, we investigate the impact of decreasing the tolerance (stopping criterion) in the iterative solvers and its effect on the final results of the adaptation process. As mentioned in Section 4.2.1, the tolerance used for both LOBPCG and GMRES is set to $tol_{solver} = 10^{-2}$. Here, we decrease the tolerance to $tol_{solver} = 10^{-8}$ and examine the main quantities of interest: the accuracy of the adaptation process and the computational cost. Only the smallest 8–element case is considered.

Both adaptation strategies in the manuscript (3–step τ–based and adjoint–based) require checking if a newly adapted mesh fulfils the target functional error threshold selected by the user. In the example shown in the paper, the functional error threshold is set to $\|J_h^N(u) - J_h^N\left(u_h^N\right)\| = 10^{-2}$. Following the details of the adaptation methods in Section 4.1, we can simplify the stopping criteria of both methods into a single expression. For the adjoint–based adaptation, we have:

$$\|\Psi^T R_h^N(u)\| < 10^{-2}, \tag{A.1}$$

and for the 3–step τ–based adaptation, we use:

$$\frac{\|\tau_h^N\| \times \left\| \partial J_h^N / \partial u_h^N \big|_{u_h^N} \right\|}{\sigma_{min}} < 10^{-2}. \tag{A.2}$$

Inequalities (A.1) or (A.2) are correctly evaluated using only two significant digits, which is a direct consequence of the number of significant digits used to define the functional error threshold (in this example 10^{-2}). In Table A.3 we compute the left hand side of (A.1) and (A.2) with two tolerances: a high tolerance $tol_{solver} = 10^{-2}$ (used throughout this work) and a low tolerance $tol_{solver} = 10^{-8}$. The table shows that both tolerances lead to the same first two digits (in scientific notation) of (A.1) and (A.2). As a consequence, both tolerances result in the same adapted meshes, i.e., $N = 3, DOF = 2560$ for the adjoint–based adaptation and $N = 5, DOF = 8640$ for the 3–step τ–based adaptation.

Regarding the computational cost, decreasing the tolerance for both Krylov solvers (GMRES and LOBPCG) results in higher computational cost, since the number of iterations required to achieve convergence increases. New computational costs of both methods with $tol_{solver} = 10^{-8}$ are included in Table A.3 and are compared with the ones obtained with $tol_{solver} = 10^{-2}$. Decreasing the tolerance to $tol_{solver} = 10^{-8}$ results in a total added cost for the adjoint–based adaptation ($N = 2$ and $N = 3$ cases) of 3.0 s, whereas the final cost for the 3–step τ–based is 2.36 s. This example illustrates that lowering the tolerance increases the computational cost but that does not improve the accuracy of the adaptation strategy. Therefore, $tol_{solver} = 10^{-2}$ is retained in this work.

Table A.3: Effect of the tolerance (stopping criterion) in the iterative solver on the adaptation results and computational cost. Two tolerances are included: $tol_{solver} = 10^{-2}$ and $tol_{solver} = 10^{-8}$. The computational cost includes preconditioner factorisation and Hermitian operator assembling (for LOBPCG).

	3–step τ–based (LOBPCG)		Adjoint–based (GMRES)	
	$\|\tau_h^N\| \times \left\| \partial J_h^N / \partial u_h^N \big\|_{u_h^N} \right\| / \sigma_{min}$		$\|\Psi^T R_h^N(u)\|$	
N	$tol_{solver} = 10^{-2}$	$tol_{solver} = 10^{-8}$	$tol_{solver} = 10^{-2}$	$tol_{solver} = 10^{-8}$
2	5.6918×10^{1}	5.6921×10^{1}	2.2472×10^{-2}	2.2395×10^{-2}
3	2.1099	2.1010	6.0211×10^{-4}	6.0448×10^{-4}
4	8.2219×10^{-2}	8.2222×10^{-2}	-	-
5	1.8710×10^{-3}	1.8770×10^{-3}	-	-
N	Computational cost [s]			
2	0.89	2.31	0.57	1.01
3	0.89	2.31	0.91	1.99
4	0.90	2.31	-	
5	0.90	2.31	-	