

p-adaptation techniques for unstructured grids: a finite volume approach

Filipe F. Tenreiro^a, Duarte M. S. Albuquerque^{a,b,*}

^aIDMEC, Instituto Superior Técnico, Technical University of Lisbon, Av. Rovisco Pais 1, 1049-001, Lisbon, Portugal

^bUNIDEMI, NOVA School of Science and Technology, NOVA University Lisbon, 2829-516, Caparica, Portugal

Abstract

With the ever-increasing power of computing resources, the development of faster algorithms to keep up with this growth is of the utmost interest to the research community. However, many computational fluid dynamics software packages still lack efficiency when high-accuracy simulations are required. Through numerical error estimation and solution-based order adaptation (or *p*-adaptation), we strive to dramatically improve their efficiency and robustness.

In order to construct an estimate for the criterion used to drive the (adaptive) process, it is first necessary to examine the sources of the error. Irrespective of the discretisation method employed, the fact that profile assumptions are necessary to derive the discretised equations means that the discretisation process inevitably introduces an error unless the underlying solution has a profile that is exactly represented by the assumed one. Provided that both iterative and round-off errors are small enough to where they do not pollute the solution, the primary source of numerical error should be associated with discretisation.

Even though it is the discretisation error that one wishes to reduce through adaptation, we should instead adapt based on the local contributions to this error, i.e., the truncation error (or perhaps an estimate of it). The two proposed methods for estimating the truncation error use a higher-order flux reconstruction about the face. While the higher-order polynomial regression estimator uses a solution that is of higher order of accuracy to approximate the leading term of the truncation error, the tau-extrapolation estimator simply attempts to extrapolate it from the underlying discrete one.

Finally, the automatic error-controlled (adaptive) algorithm uses the previous estimates to refine elements (or faces) in restricted portions of the domain deemed inaccurate. The process is applied to the steady-state form of a general scalar transport equation by choosing a solution *a priori* and then operating the governing equation onto the chosen solution, thereby generating an additional source term that requires no discretisation. Even though this method proved to be tailor-made to handle smooth problems with rapid solution changes, the presence of strong singularities appeared to significantly reduce its effectiveness.

Keywords: Finite volume method, Unstructured grids, Higher-order methods, Numerical error estimation, *p*-adaptation

*Corresponding author

Email addresses: filipe.tenreiro@tecnico.ulisboa.pt (Filipe F. Tenreiro), duartealbuquerque@tecnico.ulisboa.pt (Duarte M. S. Albuquerque)

Preprint submitted to Computer Methods in Applied Mechanics and Engineering

February 7, 2024

1 **1. Literature review**

2 The predictive capability of computational fluid dynamics (CFD) does not only depend on
3 the validity of the sub-models employed but also on the ability to accurately and reliably estimate
4 and reduce numerical errors. Most verification efforts that are documented for commercial codes
5 seem to be limited to simple benchmark examples that demonstrate “engineering accuracy”,
6 rather than verifying the order of accuracy of the code [1]. Abanto et al. [2] performed order
7 verification studies on three different CFD codes, which were formally said to be second-order
8 accurate. Most tests resulted in either first-order accuracy or non-convergent behaviour. It is
9 our opinion that code users should be aware that commercial software companies are unlikely to
10 perform rigorous code verification studies unless users request it.

11 *1.1. Adaptive mesh refinement*

12 Adaptive mesh refinement (or simply mesh adaptation) refers to the modification of the ex-
13 isting mesh to more accurately capture solution features without excessively increasing the com-
14 putational cost, i.e., it aims to close the gap between high accuracy and low computational load.
15 Central to any mesh adaptation technique are two essential requirements: a means of recognising
16 where further resolution is needed and a mechanism to actually perform adaptation. However, as
17 Hawken et al. [3] rightfully point out, even though a number of monitors are used in the literature,
18 most of them are often chosen “heuristically”, with “little to no justification” for them.

19 *1.1.1. Adaptation drivers*

20 One of the most challenging aspects of mesh adaptation is finding a good criterion to drive
21 the (adaptive) process. The least defensible methods for adaptation are based on either solution
22 features or estimates of the discretisation error (feature-based adaptation). Conversely, the most
23 rigorous ones are based on evaluating or estimating the local truncation error or, equivalently, the
24 residual (truncation error-based adaptation).

25 *1.1.2. Adaptation methods*

26 Adaptive mesh refinement methods are generally classified as one of the following: *h*-, *p*-, or
27 *r*-adaptation.

28 **Order-based adaptation (*p*-adaptation).** Order-based adaptation (or *p*-adaptation) con-
29 sists of modifying (usually increasing) the local formal order of accuracy of the discretisation
30 scheme without changing the geometry of the grid (see Fig. 1b). Even though this technique has
31 proven accurate for smooth problems, particularly if the partial differential equations (PDEs) are
32 elliptic or parabolic, it is typically not as successful when applied in the presence of discontinui-
33 ties and/or singularities, where higher-order methods become less effective [4]. Despite its
34 popularity in the finite element (FE) community, not much work has been reported in the finite
35 difference (FD) and finite volume (FV) frameworks.

36 **Mesh-based adaptation (*h*- and *r*-adaptation).** Local grid coarsening and/or refinement
37 (or *h*-adaptation) alters grid resolution without distortion by introducing and/or removing points
38 hierarchically (see Fig. 1c) [5]. One drawback of using pure *h*-adaptation, though, is that the
39 quality of the grid is adversely affected at the interface between adapted and unadapted regions.
40 The algorithm proposed by Albuquerque et al. [6] uses a grid interface correction step to provide
41 a smoother and higher-quality grid.

42 Grid movement (or r -adaptation) involves stretching the grid without changing the number
 43 of nodes or the connectivity list (see Fig. 1d) [7]. Although this technique comes at little to no
 44 added cost, care must be taken to ensure that problems due to excessive skewness do not arise.
 45 The movement of the nodes can be controlled in various ways [8]; one common technique is
 46 to treat the grid as if it were an elastic solid and solve a system of equations (subject to some
 47 forcing) that deforms the original grid.

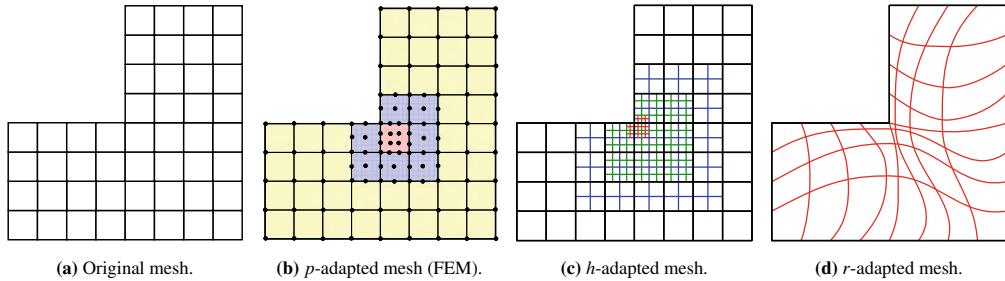


Figure 1. Adaptation methods. Adapted from [9].

48 A number of researchers have found that a combination of different adaptation methods pro-
 49 vides the best approach to reducing the discretisation error. In the finite element method (FEM),
 50 h - and p -adaptation are often combined in what is known as hp -adaptive FEs [10, 11].

51 1.2. Higher-order methods

52 Even though high-order does not have a brick-wall definition, CFD practitioners consent that
 53 a given numerical method is said to be of higher order when its order of accuracy is higher
 54 than two [12]. High-order is not a new concept, and the number of higher-order methods in the
 55 literature is considerable; these are reviewed in [13] and references therein. Although higher-
 56 order methods have been around for quite some time, not much attention was paid to them, as in
 57 the earlier stages of development, researchers were keen to make first- and second-order accurate
 58 methods more efficient. Now, having hit the saturation point in their development, the focus has
 59 turned to higher-order methods.

60 Although it is well known that a second-order accurate method is computationally more
 61 expensive than a first-order one, we can only compare method efficiency on the basis of the com-
 62 putational cost required to achieve a given error threshold. Broadly speaking, method efficiency
 63 has to do with the balance between accuracy and cost:

$$\text{Efficiency} = \frac{\text{Accuracy}}{\text{Cost}} = \underbrace{\left(\frac{\text{Accuracy}}{\text{Unknowns} \times \text{Time steps}} \right)}_{\text{Discretisation}} \underbrace{\left(\frac{\text{Unknowns} \times \text{Time steps}}{\text{Cost}} \right)}_{\text{Implementation}}. \quad (1)$$

64 Even though higher-order methods have superior accuracy per unknown, i.e., per degree of
 65 freedom (DoF), discretisation efficiency is only half of the picture; implementation efficiency is
 66 just as important. The throughput (number of unknowns at a given cost) of higher-order methods
 67 can never be higher than that of lower-order ones [14]. Nevertheless, because they possess a
 68 higher degree of data reuse, higher-order methods tend to have a higher arithmetic intensity than
 69 well-optimised lower-order ones [15], i.e., for every byte of information transferred from main
 70 memory to cache, more floating-point operations are made.

71 **2. Contributions**

72 The main contributions made during this paper are as follows:

- 73 • Proposal of a numerical reconstruction technique whose point-value interpolation is done
74 about (the centroid of) the face.

- 75 • Derivation of two distinct *a posteriori* error estimators.

76 – The consistency (i.e., asymptotic exactness) of these estimates is verified by evaluating
77 the rate of reduction of the truncation error (or discrete residual) on successively
78 finer grids.

79 – To relate the discretisation and truncation errors, we use the discrete form of the
80 (discretisation) error transport equation (28).

- 81 • Proposal of a novel order-based adaptation (or p -adaptation) method.

82 – Since the proposed (adaptive) process does not involve changing the geometry of the
83 grid, the error estimators derived in Sections 5.3.1 and 5.3.2 use a higher-order flux
84 reconstruction technique.

85 – The proposed refinement rules were inspired by those of the same names in Red-
86 Green (h -) refinement [16].

87 **3. The governing equation**

88 The conservation equations that govern the transport of mass, momentum, energy, and other
89 specific quantities have a common form embodied in the general scalar transport equation.
90 This equation simply resembles a balance between accumulation, transport, and generation of
91 a generic scalar property whose transport can be dominated by convection (associated with a
92 macroscopic movement of the medium) or by diffusion (associated with molecular diffusion).
93 The changes in this property are described in terms of ϕ : the unknown of the equation.

94 The standard form of the transport equation reads

$$\frac{\partial \phi}{\partial t} + \nabla \cdot (\mathbf{u}\phi) - \nabla \cdot (\boldsymbol{\Gamma}\nabla\phi) = \varphi(\phi), \quad (2)$$

95 where \mathbf{u} and $\boldsymbol{\Gamma}$ are the velocity and diffusivity fields, respectively, and $\varphi(\phi)$ is the volumetric
96 source (or sink) contribution to the balance of ϕ .

97 By integrating the steady-state form of Eq. (2) over a general fixed control volume (CV), this
98 equation is transformed into

$$\int_V \nabla \cdot (\mathbf{J}^{\phi,C} - \mathbf{J}^{\phi,D}) dV = \int_V \varphi(\phi) dV, \quad (3)$$

99 where $\mathbf{J}^{\phi,C} = \mathbf{u}\phi$ and $\mathbf{J}^{\phi,D} = \boldsymbol{\Gamma}\nabla\phi$ are the convective and diffusive components of the total flux
100 vector \mathbf{J}^ϕ , respectively.

101 Using the divergence theorem to replace the volume integral with a surface integral, Eq. (3)
102 becomes

$$\oint_{\partial V} (\mathbf{J}^{\phi,C} - \mathbf{J}^{\phi,D}) \cdot d\mathbf{S} = \int_V \varphi(\phi) dV, \quad (4)$$

103 where **bold** letters denote vectors, (\cdot) the dot product operator, ∂V a closed surface bounding
 104 volume V , $\oint_{\partial V}$ the surface integral over volume V and \mathbf{S} the outward pointing surface vector
 105 with associated normal \mathbf{n} , such that $\mathbf{n} dS = d\mathbf{S}$.

106 4. Finite volume (FV) discretisation

107 The standard form of the transport equation and the formal CV integration were described in
 108 Section 3. Here, we develop the numerical method based on this integration, the finite volume
 109 method (FVM). By working with the two-dimensional (2D) form of Eq. (4), the approximation
 110 techniques that are needed to obtain the so-called discretised equations are introduced.

111 For the sake of brevity, we will only resort to a one-dimensional (1D) (direct) approach to
 112 derive the general form of the two *a posteriori* error estimators introduced in Section 5.3.

113 4.1. Flux integration over element faces

114 Replacing the surface integral by a summation of fluxes over a generic cell C , the left-hand
 115 side (LHS) of Eq. (4) becomes

$$\oint_{\partial C} \mathbf{J}^\phi \cdot d\mathbf{S} = \sum_{f \sim \text{faces}(C)} \left(\int_f \mathbf{J}^\phi \cdot d\mathbf{S} \right). \quad (5)$$

116 No approximations have been made so far, and although Eq. (5) is still formally equivalent
 117 to the set of integral equations over all the CVs, the surface fluxes are evaluated at the faces of
 118 the cell rather than integrated within it. Using a Gaussian quadrature, the integral about a generic
 119 face f becomes

$$\int_f \mathbf{J}^\phi \cdot d\mathbf{S} \approx \sum_{g \sim \mathcal{G}(f)} [\omega_g \mathbf{J}^\phi(\mathbf{x}_g)] \cdot \mathbf{S}_f, \quad (6)$$

120 where g refers to an integration point and $\mathcal{G}(f)$ to the subset of integration points along face f .

121 4.2. Source term volume integration

122 Volume integration is used for the source term. Since the integration domain is a subregion
 123 of \mathbb{R}^2 , the right-hand side (RHS) of Eq. (4) becomes

$$\begin{aligned} \iint_C \varphi dS &= \iint_C \left(\frac{\partial \varphi_x}{\partial x} + \frac{\partial \varphi_y}{\partial y} \right) dS \\ &= \oint_{\partial C} (-\varphi_y dx + \varphi_x dy) \\ &= \sum_{f \sim \text{faces}(C)} \left(\int_f \varphi \cdot d\mathbf{S} \right) \\ &= \sum_{f \sim \text{faces}(C)} \left(\int_f \varphi(\mathbf{r}(t)) \cdot \mathbf{r}'(t) dt \right), \end{aligned} \quad (7)$$

where the vector field and the parametrisation of a generic line segment on the contour of C are respectively given by

$$\begin{aligned}\varphi(x, y) &= -\varphi_y(x, y)\mathbf{i} + \varphi_x(x, y)\mathbf{j} \\ &= -\left(v\Phi - \Gamma_y \frac{\partial\Phi}{\partial y}\right)\mathbf{i} + \left(u\Phi - \Gamma_x \frac{\partial\Phi}{\partial x}\right)\mathbf{j}, \text{ with } \Phi \equiv \Phi(x, y)\end{aligned}\quad (8a)$$

$$\mathbf{r}(t) = x(t)\mathbf{i} + y(t)\mathbf{j}. \quad (8b)$$

124

125 4.3. Numerical solution reconstruction

126 The approximations to the integral in Eq. (6) require the values of ϕ and its gradient at locations other than the computational nodes (CV centroids). Here, since the point-value interpolation 127 is done about the centroid of the face, there is no need to interpolate the cell/vertex-centred 128 reconstruction to obtain the surface fluxes.

129 The local approximation of the underlying solution about the centroid of f can be modelled 130 as a n th-degree polynomial, yielding the general polynomial regression model 131

$$\begin{aligned}\phi_f(\mathbf{x}) &= \mathbf{c}_f^T \mathbf{p}_n(\mathbf{x} - \mathbf{x}_f) + \mathbf{r}_f \\ &= \mathbf{c}_f^T \sum_{i=0}^n \sum_{j=0}^{n-i} (x - x_f)^i (y - y_f)^j + \mathbf{r}_f,\end{aligned}\quad (9)$$

132 where $\mathbf{p}_n(\mathbf{x})$ is a 2D polynomial basis function vector of degree n and \mathbf{r}_f is a residual vector.

133 The polynomial regression model

$$\phi_f(x_i, y_i) = c_0 + c_1(x_i - x_f) + c_2(y_i - y_f) + \dots + r_{fi}, \text{ with } i = 1, 2, \dots \quad (10)$$

134 can be expressed in matrix form (for each face) in terms of a design matrix $\mathbf{D}_f \in \mathbb{R}^{m \times n}$, a parameter 135 vector $\mathbf{c}_f \in \mathbb{R}^n$, a (fitted) response vector $\phi_s \in \mathbb{R}^m$, and a residual vector $\mathbf{r}_f \in \mathbb{R}^m$ as

$$\underbrace{\begin{bmatrix} 1 & (x_1 - x_f) & (y_1 - y_f) & \dots \\ \vdots & \vdots & \vdots & \ddots \\ 1 & (x_b - x_f) & (y_b - y_f) & \dots \\ 0 & n_{b_x} & n_{b_y} & \dots \\ n_{p_x} + n_{b_y} & [(x_b - x_f) + d_{b_x}] \cdot n_{b_x} & [(y_b - y_f) + d_{b_y}] \cdot n_{b_y} & \dots \end{bmatrix}}_{\mathbf{D}_f} \underbrace{\begin{bmatrix} c_1 \\ \vdots \\ c_b \\ c_b \\ c_p \end{bmatrix}}_{\mathbf{c}_f} = \underbrace{\begin{bmatrix} \phi_1 \\ \vdots \\ \phi_b \\ \nabla \phi_b \cdot \mathbf{n}_b \\ [\phi_b + \mathbf{d}_b \nabla \phi_b] \cdot \mathbf{n}_b \end{bmatrix}}_{\phi_s} - \underbrace{\begin{bmatrix} r_1 \\ \vdots \\ r_b \\ r_b \\ r_b \end{bmatrix}}_{\mathbf{r}_f}, \quad (11)$$

136 which when using pure matrix notation can be written as

$$\mathbf{D}_f \mathbf{c}_f = \hat{\phi}_s, \quad (12)$$

137 with $\hat{\phi}_s = \phi_s - \mathbf{r}_f$ and

$$\mathbf{d}_b \equiv \mathbf{d}(\mathbf{x}_b) = \frac{\boldsymbol{\Gamma}(\mathbf{x}_b)}{\mathbf{u}(\mathbf{x}_b)}. \quad (13)$$

138 The last three rows of this system correspond to boundary condition equations. The one 139 below the red line corresponds to a Dirichlet condition, whereas the ones below the green and 140 blue lines correspond to Neumann and Robin conditions, respectively. If the discretisation 141 stencil contains a boundary face for which either of these conditions is prescribed, the respective 142 equation should be written as in Eq. (11).

143 4.3.1. Unconstrained polynomial reconstruction

144 Consider the previous “overdetermined” (linear) system of equations, where \mathbf{D}_f is assumed
 145 to have full (column) rank. If there is no solution to this system because $\boldsymbol{\phi}_s$ does not lie in the
 146 range of \mathbf{D}_f , then the best approximation is one that minimises the “sum of squares” of the
 147 residuals.

148 In most situations, however, it is desirable to minimise the *weighted* square residual instead:

$$\min_{\mathbf{c}_f} \left\| \sqrt{\mathbf{W}}_f \mathbf{r}_f \right\|_2^2 = \min_{\mathbf{c}_f} \left\| \sqrt{\mathbf{W}}_f (\boldsymbol{\phi}_s - \mathbf{D}_f \mathbf{c}_f) \right\|_2^2, \quad (14)$$

149 where \mathbf{W}_f is a diagonal matrix with positive weights w_i ¹.

150 As detailed in [Appendix A.1.1](#), practical solution methods do not *explicitly* form the product
 151 $\mathbf{D}_f^\top \mathbf{W}_f \mathbf{D}_f$ to solve the (weighted) unconstrained least-squares (ULS) problem in Eq. (14) for \mathbf{c}_f
 152 (recall that the behaviour of the normal equations is governed by κ^2 , not κ). Instead, we use
 153 the QR decomposition of the (weighted) design matrix $\sqrt{\mathbf{W}}_f \mathbf{D}_f$ ² to solve the triangular system
 154 $\tilde{\mathbf{R}}_f \mathbf{c}_f^* = \tilde{\mathbf{Q}}_f^\top \sqrt{\mathbf{W}}_f \boldsymbol{\phi}_s$ for a unique solution $\mathbf{c}_f^* = \mathbf{P}_f \boldsymbol{\phi}_s$, where $\mathbf{P}_f = \tilde{\mathbf{R}}_f^{-1} \tilde{\mathbf{Q}}_f^\top \sqrt{\mathbf{W}}_f$ (see [Appendix A.1.2](#)).

155 Using these expressions, the general discretised relations for the convective and diffusive
 terms of the integral about face f in Eq. (6) can now be written as

$$\int_f \mathbf{J}^{\phi,C} \cdot d\mathbf{S} \approx \left[\begin{bmatrix} u(\mathbf{x}_{g_1}) \omega_{g_1} & u(\mathbf{x}_{g_2}) \omega_{g_2} & \dots \\ v(\mathbf{x}_{g_1}) \omega_{g_1} & v(\mathbf{x}_{g_2}) \omega_{g_2} & \dots \end{bmatrix} \mathbf{d}_f^C(\mathbf{x}_g) \right] [\mathbf{P}_f \boldsymbol{\phi}_s] \cdot \mathbf{S}_f \quad (16a)$$

$$= [\mathbf{t}_f^C \boldsymbol{\phi}_s] \cdot \mathbf{S}_f$$

$$\int_f \mathbf{J}^{\phi,D} \cdot d\mathbf{S} \approx \left[\begin{bmatrix} \Gamma_x(\mathbf{x}_{g_1}) \omega_{g_1} & \Gamma_x(\mathbf{x}_{g_2}) \omega_{g_2} & \dots \\ \Gamma_y(\mathbf{x}_{g_1}) \omega_{g_1} & \Gamma_y(\mathbf{x}_{g_2}) \omega_{g_2} & \dots \end{bmatrix} \mathbf{d}_f^{D_x}(\mathbf{x}_g) \right] [\mathbf{P}_f \boldsymbol{\phi}_s] \cdot \mathbf{S}_f \quad (16b)$$

$$= [\mathbf{t}_f^D \boldsymbol{\phi}_s] \cdot \mathbf{S}_f,$$

where

$$\mathbf{d}_f^C(\mathbf{x}) = \left[\begin{array}{cccccc} 1 & (x - x_f) & (y - y_f) & (x - x_f)^2 & (x - x_f)(y - y_f) & (y - y_f)^2 & \dots \end{array} \right]^\top \quad (17a)$$

$$\mathbf{d}_f^{D_x}(\mathbf{x}) = \left[\begin{array}{cccccc} 0 & 1 & 0 & 2(x - x_f) & (y - y_f) & 0 & \dots \end{array} \right]^\top \quad (17b)$$

$$\mathbf{d}_f^{D_y}(\mathbf{x}) = \left[\begin{array}{cccccc} 0 & 0 & 1 & 0 & (x - x_f) & 2(y - y_f) & \dots \end{array} \right]^\top \quad (17c)$$

156 and \mathbf{t}_f is the (discrete) operator vector.

¹The following inverse-distance weighting function is widely used in FV methods [17]:

$$w_i = \frac{1}{d_i^n}, \quad (15)$$

where $n = 0, 1, 2, 3$, etc., and d_i is the distance between the centroid of face f and the reference location of the i th element of the discretisation stencil. Here, we simply set n to 2.

²Once assembled, the (weighted) design matrix should be normalised using the procedure described in [Appendix B](#).

157 4.3.2. Constrained polynomial reconstruction

158 For the method described here to satisfy the imposed boundary condition exactly, the recon-
 159 struction of boundary faces must be constrained. Rather than enforcing the boundary condition
 160 all along the edge of the respective boundary CV, we enforce one constraint per integration (or
 161 Gauss) point.

162 The (linearly) constrained least-squares (CLS) problem refers to the problem of finding a
 163 least-squares solution that exactly satisfies q additional scalar constraints of the form:

$$\underbrace{\begin{bmatrix} 1 & (x_g - x_f) & (y_g - y_f) & \dots \\ \vdots & \vdots & \vdots & \ddots \\ 0 & n_{g_x} & n_{g_y} & \dots \\ \vdots & \vdots & \vdots & \ddots \\ n_{g_x} + n_{g_y} & [(x_g - x_f) + d_{g_x}] \cdot n_{g_x} & [(y_g - y_f) + d_{g_y}] \cdot n_{g_y} & \dots \\ \vdots & \vdots & \vdots & \ddots \end{bmatrix}}_{\mathbf{H}_f} = \underbrace{\begin{bmatrix} c_1 \\ \vdots \\ c_1 \\ \vdots \\ c_1 \\ \vdots \end{bmatrix}}_{\mathbf{c}_f} = \underbrace{\begin{bmatrix} \phi_g \\ \vdots \\ \nabla \phi_g \cdot \mathbf{n}_g \\ \vdots \\ [\phi_g + \mathbf{d}_g \nabla \phi_g] \cdot \mathbf{n}_g \\ \vdots \end{bmatrix}}_{\boldsymbol{\phi}_g}, \quad (18)$$

164 which when using pure matrix notation can be written as

$$\mathbf{H}_f \mathbf{c}_f = \boldsymbol{\phi}_g, \quad (19)$$

165 where $\mathbf{H}_f \in \mathbb{R}^{q \times n}$ and $\boldsymbol{\phi}_g \in \mathbb{R}^q$, and

$$\mathbf{d}_g \equiv \mathbf{d}(\mathbf{x}_g) = \frac{\Gamma(\mathbf{x}_g)}{\mathbf{u}(\mathbf{x}_g)}. \quad (20)$$

166 The three blocks of equations in Eq. (18) correspond to boundary condition constraints.
 167 While the first block denotes a Dirichlet constraint, the second (below the green line) and third
 168 (below the blue line) ones denote Neumann and Robin constraints, respectively.

169 Following the procedure detailed in [Appendix A.2](#), we can now write the optimality conditions
 170 for the (weighted) CLS problem as one set of $n + q$ (linear) equations as

$$\left[\begin{array}{cc} 2\mathbf{D}_f^\top \mathbf{W}_f \mathbf{D}_f & \mathbf{H}_f^\top \\ \mathbf{H}_f & \mathbf{0} \end{array} \right] \left[\begin{array}{c} \mathbf{c}_f \\ \boldsymbol{\mu} \end{array} \right] = \left[\begin{array}{c} 2\mathbf{D}_f \sqrt{\mathbf{W}_f} \boldsymbol{\phi}_s \\ \boldsymbol{\phi}_g \end{array} \right], \quad (21)$$

171 where $\boldsymbol{\mu} \in \mathbb{R}^q$ is a set of Lagrange multipliers.

172 These equations are called the Karush-Kuhn-Tucker (KKT) equations for the CLS problem
 173 (KKT are the initials of the last names of William Karush, Harold Kuhn, and Albert Tucker,
 174 the three researchers who derived the optimality conditions for a more general form of the con-
 175 strained minimisation problem).

176 Assuming that \mathbf{H}_f has linearly independent rows and the stacked matrix $\left[\begin{array}{c} \sqrt{\mathbf{W}_f} \mathbf{D}_f \\ \mathbf{H}_f \end{array} \right]$
 177 has linearly independent columns, we can solve this set of equations for a unique solution
 178 $\mathbf{c}_f^* = \mathbf{P}_f \boldsymbol{\phi}_s + \mathbf{k}_f$, where $\mathbf{P}_f = \tilde{\mathbf{R}}_f^{-1} \{ [\tilde{\mathbf{Q}}_{1f}^\top - \tilde{\mathbf{Q}}_{2f}^\top (\tilde{\mathbf{U}}_f^{-1} \tilde{\mathbf{T}}_f^\top) \tilde{\mathbf{Q}}_{1f}^\top] \sqrt{\mathbf{W}_f} \}$ and $\mathbf{k}_f = \tilde{\mathbf{R}}_f^{-1} \{ (\tilde{\mathbf{U}}_f^{-1} \tilde{\mathbf{T}}_f^\top)^\top \boldsymbol{\phi}_g \}$
 179 (see [Appendix A.2.1](#)).

Similarly to the unconstrained formulation, the general discretised relations for the convective and diffusive terms of the integral about face f in Eq. (6) can now be written as

$$\int_f \mathbf{J}^{\phi,C} \cdot d\mathbf{S} \approx \left[\begin{array}{ccc} u(\mathbf{x}_{g_1})\omega_{g_1} & u(\mathbf{x}_{g_2})\omega_{g_2} & \dots \\ v(\mathbf{x}_{g_1})\omega_{g_1} & v(\mathbf{x}_{g_2})\omega_{g_2} & \dots \end{array} \right] \mathbf{d}_f^C(\mathbf{x}_g) \left[\begin{array}{c} \mathbf{P}_f \boldsymbol{\phi}_s + \mathbf{k}_f \end{array} \right] \cdot \mathbf{S}_f \quad (22a)$$

$$= [\mathbf{t}_f^C \boldsymbol{\phi}_s + \mathbf{w}_f^C] \cdot \mathbf{S}_f$$

$$\int_f \mathbf{J}^{\phi,D} \cdot d\mathbf{S} \approx \left[\begin{array}{ccc} \Gamma_x(\mathbf{x}_{g_1})\omega_{g_1} & \Gamma_x(\mathbf{x}_{g_2})\omega_{g_2} & \dots \\ \Gamma_y(\mathbf{x}_{g_1})\omega_{g_1} & \Gamma_y(\mathbf{x}_{g_2})\omega_{g_2} & \dots \end{array} \right] \mathbf{d}_f^{D_x}(\mathbf{x}_g) \left[\begin{array}{c} \mathbf{P}_f \boldsymbol{\phi}_s + \mathbf{k}_f \end{array} \right] \cdot \mathbf{S}_f \quad (22b)$$

$$= [\mathbf{t}_f^D \boldsymbol{\phi}_s + \mathbf{w}_f^D] \cdot \mathbf{S}_f,$$

where \mathbf{t}_f and \mathbf{w}_f are the (discrete) operator and constraint vectors, respectively.

4.3.3. Discretisation stencil

Due to the predominantly elliptic nature of the problem at hand, we will employ a central-biased stencil for faces on the bulk of the domain and a non-central (e.g., upwind/downwind) one for the ones near/on the boundary of the domain, where we cannot use central discretisation.

The number of elements that make up the discretisation stencil (m) is inherently determined by the number of monomial terms in the 2D polynomial basis function (n) and by the number of constraints (q) applied to the respective polynomial fit. The more elements (or equations) used, the more computationally expensive the scheme becomes. It is, therefore, crucial to reduce computational cost³ without sacrificing the robustness of the scheme.

The most common approaches adopted in [18–25] and references therein are based on the recursive addition of elements (one by one [20, 21] or layer by layer [23]) until a (minimum) number of elements determined beforehand is reached or a (maximum) condition number is attained. However, one must ensure that some stopping criteria are introduced in case the target metric cannot be reached [25].

Here, we simply add elements (one by one) based on the distance to the centroid of the face about which the polynomial is reconstructed (or fitted) until the problem becomes “sufficiently” well-conditioned (see Fig. 2). Since, in this work, we only deal with regular grids, it suffices to guarantee that the problem is both locally and globally non-singular. If irregular (or very skewed) grids were to be used, then an upper bound should be placed on some ℓ_k -norm of the condition number of matrices \mathbf{D}_f (ULS formulation) or $\left[\frac{\mathbf{D}_f}{\mathbf{H}_f} \right]$ (CLS formulation), so as to not compromise the robustness of the scheme.

4.3.4. General approach

The result of the discretisation process is a (linear) system of equations of the form

$$\mathbf{A}\boldsymbol{\phi} = \mathbf{b}, \quad (23)$$

where $\boldsymbol{\phi}$ is the discrete solution vector. In this system, the coefficients of the unknown variables that constitute matrix \mathbf{A} are a result of the linearisation procedure and the grid geometry, while vector \mathbf{b} contains all sources, constants, and boundary conditions.

³Here, we use the density (or the number of non-zero entries) of the global coefficient matrix (\mathbf{A}) in Eq. (23) as an alias for the computational cost (or the number of DoFs).

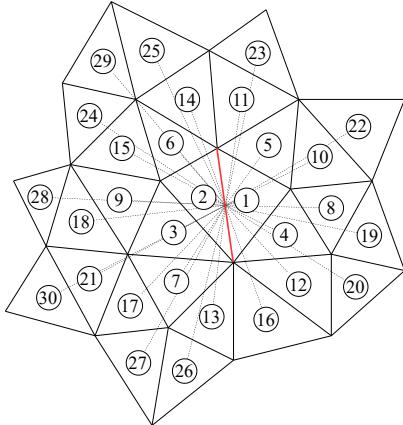


Figure 2. Central-biased stencil about the face marked in red (elements are numbered in the order that they are added).

Denoting the cell- and face-dependent entries of the operator (and constraint) vectors by s_c and s_f , respectively, \mathbf{A} and \mathbf{b} can be assembled in the following way:

$$\mathbf{A}_{ij} = \sum_{f \sim \text{faces}(i)} [\mathbf{t}_f^C(s_c) - \mathbf{t}_f^D(s_c)] \cdot \mathbf{S}_f \quad (24a)$$

$$\mathbf{b}_i = \iint_i \varphi dS - \sum_{f \sim \text{faces}(i)} \left\{ [\mathbf{t}_f^C(s_f) \boldsymbol{\phi}_s(s_f) - \mathbf{t}_f^D(s_f) \boldsymbol{\phi}_s(s_f)] + \underbrace{(\mathbf{w}_f^C - \mathbf{w}_f^D)}_{\text{CLS formulation (only)}} \right\} \cdot \mathbf{S}_f. \quad (24b)$$

207

208 5. Numerical verification

209 The validation of CFD codes for relevant applications is an extremely complex process, as
210 it is hardly possible to generate experimental data at the same level of detail as that provided by
211 the simulation. In addition, since both experimental and numerical data are constantly subjected
212 to many error and uncertainty sources [26], an *absolute* validation of those results remains an
213 unreachable objective. These uncertainties make the verification phase even more essential to
214 ensure that, at least, the numerical discretisation is foolproof. Consequently, we will restrict our
215 analysis to this last phase.

216 5.1. Exact solutions

217 For rigorous code verification, we must have access to the exact solution of the underlying
218 governing equations. However, whenever exact solutions are found for complex PDEs, they
219 often rely on significant simplifications in dimensionality, geometry, physics, etc. (e.g., Couette
220 flow). For this reason, we will employ the method of manufactured solutions (MMS).

221 5.2. Error sources

Suppose that the (possibly non-linear) differential equation is given by

$$\mathcal{L}(\Phi) = 0 \quad (25a)$$

10

and the difference approximation by

$$\mathcal{L}_{(p)}(\phi) = 0, \quad (25b)$$

where $\mathcal{L}_{(p)}(\cdot)$ is the p th-order accurate discrete form of the symbolic differential operator $\mathcal{L}(\cdot)$.

In general, the (sufficiently smooth) exact solution to the governing equations will not satisfy the difference approximation in Eq. (25b). The difference (or remainder),

$$\bar{\tau}^\Phi \equiv \bar{\tau}(\Phi) = \mathcal{L}_{(p)}(\Phi) - \mathcal{L}_{(p)}(\phi) \quad (26)$$

is called *truncation error*.

The solution to the discretised equations differs from the exact solution to the governing equations by the *discretisation error*, i.e.,

$$e^\Phi \equiv e(\Phi) = \phi - \Phi. \quad (27)$$

If the symbolic operator $\mathcal{L}_{(p)}(\cdot)$ is linear, then $\mathcal{L}_{(p)}(\Phi) - \mathcal{L}_{(p)}(\phi) = \mathcal{L}_{(p)}(\Phi - \phi)$, and thus Eq. (26) simplifies to

$$\bar{\tau}^\Phi = -\mathcal{L}_{(p)}(e^\Phi). \quad (28a)$$

Otherwise, for small enough e^Φ , it becomes

$$\bar{\tau}^\Phi \approx -\mathcal{L}'_{(p)}(e^\Phi), \quad (28b)$$

where $\mathcal{L}'_{(p)}$ is the Jacobian matrix of $\mathcal{L}_{(p)}$. This equation governs the transport of the discretisation error and is called the *discretisation error transport equation* since it employs the discrete symbolic differential operator [27].

5.2.1. General approach

To derive the analytical expression for the local truncation error about a generic face e in the bulk of the domain, consider the uniformly spaced 1D grid in Fig. 3.

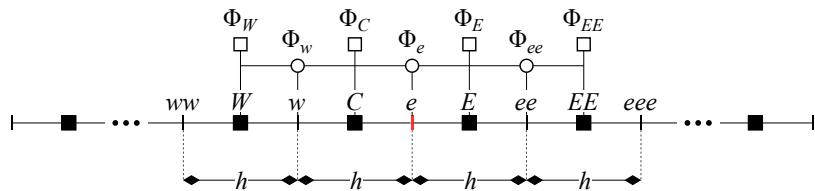


Figure 3. Uniformly spaced 1D grid (spacing h).

Firstly, we start by writing the facial values of $\phi(x)$ and $\phi(x)'$ about e at $x = x_e$ in terms of the parameter vector of the polynomial regression model as

$$\phi_e(x_e) = \frac{1}{2}\phi_C + \frac{1}{2}\phi_E \quad (29a)$$

$$\phi'_e(x_e) = -\frac{1}{h}\phi_C + \frac{1}{h}\phi_E. \quad (29b)$$

²³⁴ Then, using the general expression in Eq. (26), we may write the local truncation error about
²³⁵ face e at $x = x_e$ as

$$\begin{aligned}\bar{\tau}_e^\Phi(x_e) &= \bar{\tau}_e^{\Phi,C}(x_e) - \bar{\tau}_e^{\Phi,D}(x_e) \\ &= \underbrace{u_e \left(\Phi_e + \frac{h^2}{8} \Phi_e'' + \dots \right)}_{\Phi_e^{(p)}(x_e)} - \Gamma_e \underbrace{\left(\Phi'_e + \frac{h^2}{24} \Phi_e''' + \dots \right)}_{\Phi_e'^{(p)}(x_e)} - \underbrace{\left[u_e \underbrace{\left(\frac{\phi_C + \phi_E}{2} \right)}_{\phi_e^{(p)}(x_e)} - \Gamma_e \underbrace{\left(\frac{\phi_E - \phi_C}{h} \right)}_{\phi_e'^{(p)}(x_e)} \right]}_{\mathbf{t}_e \boldsymbol{\Phi}_s} \\ &= \underbrace{u_e \left[\Phi_e - \left(\frac{\phi_C + \phi_E}{2} \right) \right]}_{\bar{\tau}_0^\Phi|_e(x_e)} - \Gamma_e \underbrace{\left[\Phi'_e - \left(\frac{\phi_E - \phi_C}{h} \right) \right]}_{\bar{\tau}_1^\Phi|_e(x_e)} + u_e \underbrace{\left(\frac{h^2}{8} \Phi_e'' + \dots \right)}_{\bar{\tau}_1^\Phi|_e(x_e) + \dots} - \Gamma_e \underbrace{\left(\frac{h^2}{24} \Phi_e''' + \dots \right)}_{= a_1^\Phi|_e(x_e) h^2 + \dots},\end{aligned}\quad (30)$$

where

$$\begin{aligned}\bar{\tau}_e^{\Phi,C}(x_e) &= \mathbf{t}_e^C \boldsymbol{\Phi}_s - \mathbf{t}_e^C \boldsymbol{\phi}_s \\ &= u_e \left[\Phi_e^{(p)}(x_e) - \phi_e(x_e) \right] \\ &= u_e \left[\left(\frac{\Phi_C + \Phi_E}{2} \right) - \left(\frac{\phi_C + \phi_E}{2} \right) \right] \\ &\stackrel{4}{=} u_e \left[\left(\Phi_e + \frac{h^2}{8} \Phi_e'' + \dots \right) - \left(\frac{\phi_C + \phi_E}{2} \right) \right]\end{aligned}\quad (31a)$$

and

$$\begin{aligned}\bar{\tau}_e^{\Phi,D}(x_e) &= \mathbf{t}_e^D \boldsymbol{\Phi}_s - \mathbf{t}_e^D \boldsymbol{\phi}_s \\ &= \Gamma_e \left[\Phi_e'^{(p)}(x_e) - \phi_e'(x_e) \right] \\ &= \Gamma_e \left[\left(\frac{\Phi_E - \Phi_C}{h} \right) - \left(\frac{\phi_E - \phi_C}{h} \right) \right] \\ &\stackrel{4}{=} \Gamma_e \left[\left(\Phi'_e + \frac{h^2}{24} \Phi_e''' + \dots \right) - \left(\frac{\phi_E - \phi_C}{h} \right) \right]\end{aligned}\quad (31b)$$

²³⁶ are its (weighted) convective and diffusive components at $x = x_e$, respectively.

²³⁷ Rewriting the face truncation (or tau) error in Eq. (30) in a more compact form yields

$$\begin{aligned}\bar{\tau}_e^\Phi(x_e) &= \underbrace{\bar{\tau}_0^\Phi|_e(x_e)}_{\text{Tau}} + \underbrace{\bar{\tau}_{\Sigma k}^\Phi|_e(x_e)}_{\text{Tau-k}} \\ &\equiv \bar{\tau}_0^\Phi|_e(x_e) + \sum_{k=m}^{\infty} \bar{\tau}_k^\Phi|_e(x_e),\end{aligned}\quad (33)$$

²³⁸ where m refers to the leading term of its tau-k component.

⁴ The 1D Taylor series expansions of Φ_C and Φ_E about face e are given by

$$\Phi_C = \Phi_e - \frac{h}{2} \Phi'_e + \frac{h^2}{8} \Phi_e'' - \frac{h^3}{48} \Phi_e''' + \frac{h^4}{384} \Phi_e^{iv} - \frac{h^5}{3840} \Phi_e^v + \dots \quad (32a)$$

$$\Phi_E = \Phi_e + \frac{h}{2} \Phi'_e + \frac{h^2}{8} \Phi_e'' + \frac{h^3}{48} \Phi_e''' + \frac{h^4}{384} \Phi_e^{iv} + \frac{h^5}{3840} \Phi_e^v + \dots, \quad (32b)$$

where Φ'_e , Φ_e'' , Φ_e''' , Φ_e^{iv} and Φ_e^v denote the first, second, third, fourth, and fifth derivatives of $\Phi(x)$ at $x = x_e$.

²³⁹ 5.2.2. *Closure*

Using the 1D expressions derived in Section 5.2.1, we can now write the 2D form of the two components of the truncation error of a generic cell C in a more general form as

$$\begin{aligned}\bar{\tau}_0^\Phi|_C &= \iint_C \varphi dS - \sum_{f \sim \text{faces}(C)} \left(\int_f \mathbf{J}^\phi \cdot d\mathbf{S} \right) \\ &\approx \sum_{f \sim \text{faces}(C)} \left(\int_f \varphi \cdot d\mathbf{s} - \sum_{g \sim \mathcal{G}(f)} [\omega_g \mathbf{J}^\phi(\mathbf{x}_g)] \cdot \mathbf{S}_f \right) \\ &= \sum_{f \sim \text{faces}(C)} \left(\int_f \varphi \cdot d\mathbf{s} - [\mathbf{t}_f \boldsymbol{\phi}_s + \mathbf{w}_f] \cdot \mathbf{S}_f \right) \\ &\stackrel{5}{=} 0\end{aligned}\tag{34a}$$

$$\begin{aligned}\sum_{k=m}^{\infty} \bar{\tau}_k^\Phi|_C &= \sum_{f \sim \text{faces}(C)} \left(\int_f \mathbf{J}^\Phi \cdot d\mathbf{S} \right) - \iint_C \varphi dS \\ &= \sum_{f \sim \text{faces}(C)} \left([\mathbf{t}_f \boldsymbol{\Phi}_s + \mathbf{w}_f] \cdot \mathbf{S}_f - \int_f \varphi \cdot d\mathbf{s} \right) \\ &= \sum_{f \sim \text{faces}(C)} \left(\sum_{k=m}^{\infty} \bar{\tau}_k^\Phi|_f(x_f) \cdot \mathbf{S}_f \right),\end{aligned}\tag{34b}$$

²⁴⁰ where m refers to the leading term of its tau-k component.

²⁴¹ 5.3. *Error estimation*

²⁴² In verification, the exact solution of the mathematical model is generally not known, and
²⁴³ thus errors must be estimated rather than simply evaluated. Current efforts in estimating the
²⁴⁴ discretisation error are focused on *a posteriori* methods. These methods use the solution of
²⁴⁵ the discretised equations to estimate the discretisation error relative to the exact solution of the
²⁴⁶ mathematical model.

²⁴⁷ In general, the level of reliability of *a posteriori* methods is strongly problem-dependent.
²⁴⁸ As detailed in Section 4.3.3, all the methods developed here are predominantly suited for ellip-
²⁴⁹ tic problems, and as a result, they are not as well-developed for mathematical models that are
²⁵⁰ parabolic or hyperbolic in nature. The level of complexity of the problem is also an important
²⁵¹ issue. These error estimators work well for smooth, linear problems with simple physics and ge-
²⁵²ometries; however, strong non-linearities, discontinuities, singularities, and physical complexity
²⁵³ can significantly reduce their applicability.

²⁵⁴ **Example 1.** Consider the 1D steady-state form of Eq. (2) discretised with a second-order accu-
²⁵⁵ rate scheme over the domain $[-1, 1]$, where $u = 1 \text{ [m/s]}$ and $\Gamma = 1 \text{ [m}^2/\text{s]}$. Dirichlet conditions
²⁵⁶ were prescribed for both boundaries, and the (manufactured) solution to this equation was as-
²⁵⁷ sumed to be:

$$\Phi(x) = x^5.\tag{35}$$

⁵ $\bar{\tau}_0^\Phi|_C$ corresponds to the difference between the integral of the source term and the (discrete) surface flux integral. Since the source term is computed exactly, then $\bar{\tau}_0^\Phi|_C = 0$. If a Gaussian quadrature integration was instead used to discretise the RHS of Eq. (4), this would not be the case.

258 The 1D distribution of the analytical scalar property and respective volumetric source (or
259 sink) term is shown in Figs. 4a and 4b, respectively.

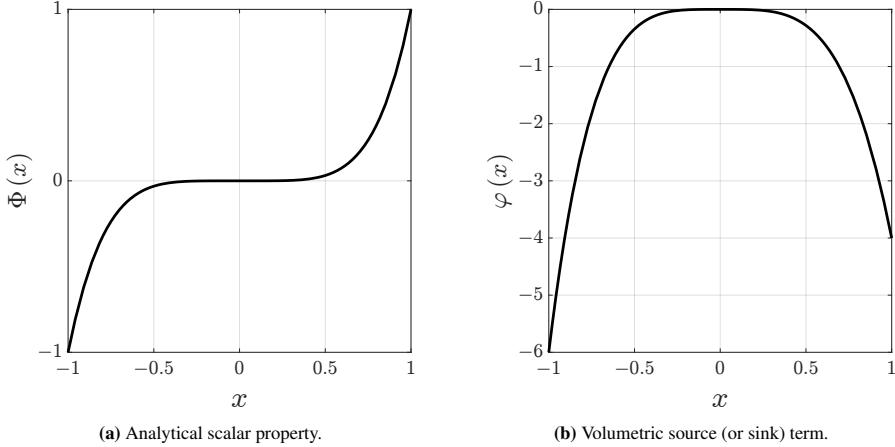


Figure 4. 1D distribution of the analytical scalar property and respective volumetric source (or sink) term.

260 In the next two Sections (5.3.1 and 5.3.2), we present two distinct forms of estimating the
261 numerical error, both of which explicitly require the formulation of a higher-order problem in the
262 original grid.

263 5.3.1. Higher-order polynomial regression

264 The estimate presented here seeks to find an approximation of the exact solution that is of
265 higher order of accuracy than the underlying discrete one. Examples of this approach can be
266 found in the works of Hay and Visonneau [28] and Hay et al. [29].

267 Replacing the analytical response vector with a discrete one of higher order (e.g., fourth-order
268 accuracy), Eq. (30) becomes

$$\begin{aligned} \bar{\tau}_e^{\phi(h)}(x_e) &= \underbrace{u_e \left(\phi_e + \frac{h^2}{8} \phi_e'' \right) - \Gamma_e \left(\phi_e' + \frac{h^2}{24} \phi_e''' \right)}_{\mathbf{t}_{e(l)} \boldsymbol{\phi}_{S(h)}} - \underbrace{\left[u_e \left(\frac{\phi_C + \phi_E}{2} \right) - \Gamma_e \left(\frac{\phi_E - \phi_C}{h} \right) \right]}_{\mathbf{t}_{e(l)} \boldsymbol{\phi}_{S(l)}} \\ &= \underbrace{u_e \left[\phi_e - \left(\frac{\phi_C + \phi_E}{2} \right) \right] - \Gamma_e \left[\phi_e' - \left(\frac{\phi_E - \phi_C}{h} \right) \right]}_{\bar{\tau}_0^{\phi(h)}|_e(x_e)} + \underbrace{u_e \left(\frac{h^2}{8} \phi_e'' \right) - \Gamma_e \left(\frac{h^2}{24} \phi_e''' \right)}_{\bar{\tau}_1^{\phi(h)}|_e(x_e)} , \end{aligned} \quad (36)$$

269 where the subscripts (*l*) and (*h*) refer to the lower- and higher-order accurate discretisation
270 schemes, respectively (note that the facial values ϕ_e , ϕ'_e , ϕ''_e , and ϕ'''_e are approximated by a
271 cubic variation of $\phi(x)$ about the face).

272 If the difference between the analytical expression in Eq. (30) and the finite estimate in
273 Eq. (36) is “small” in comparison to the latter, then this estimate is a “good” approximation
274 of the local face truncation error (see Fig. 5).

275 As shown in Fig. 6, we can see that (in the asymptotic range) this difference is indeed negli-
276 gible, i.e.,

$$\mathcal{L}_{(2)}(\Phi - \phi_{(4)}) \approx O(h^4). \quad (37)$$

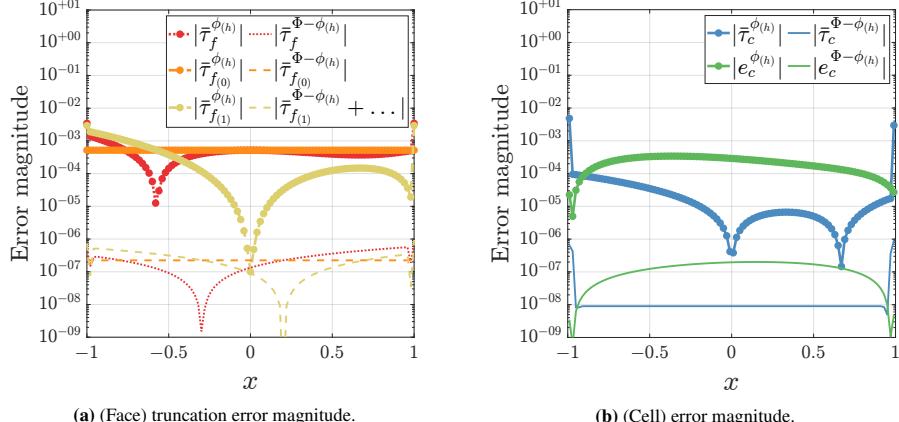


Figure 5. 1D distribution of the magnitude of the error for a uniformly spaced grid with 100 CVs (2nd-order accuracy).

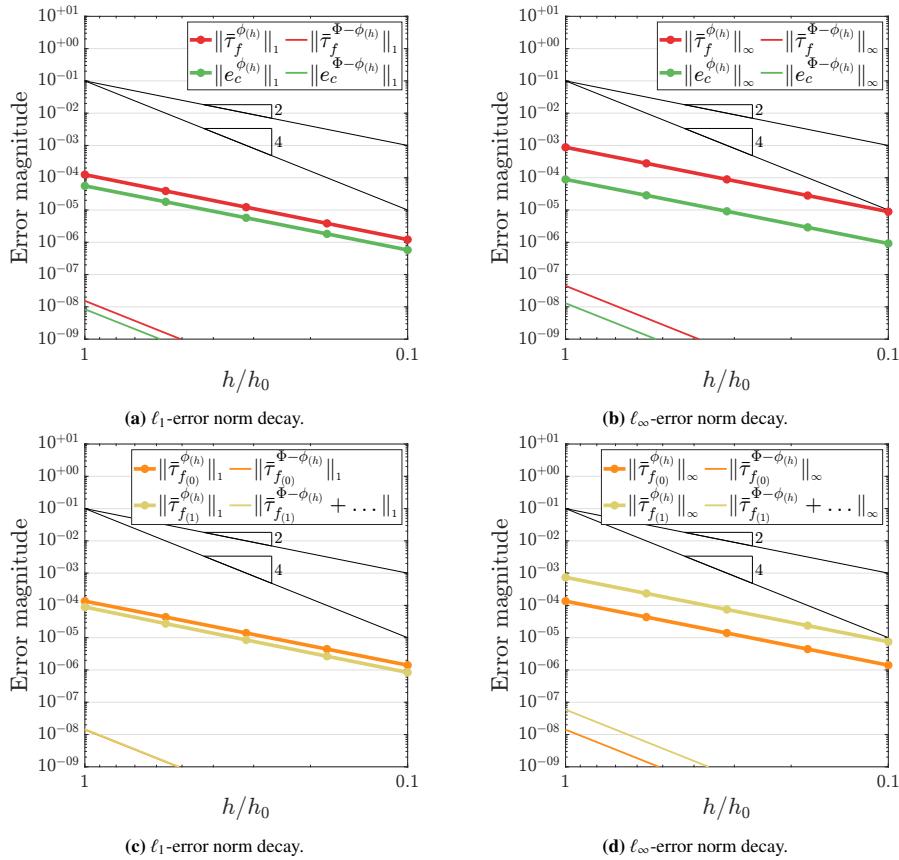


Figure 6. Error norm decay for increasingly refined uniformly spaced grids, with $h_0 = 1e-02$ (2nd-order accuracy).

277 5.3.2. Tau-extrapolation

278 The estimate presented here uses an extrapolation of the lower-order underlying discrete
 279 solution to obtain a higher-order variation of $\phi(x)$ about the face. In other words, this variation
 280 simply corresponds to substituting the q th-order accurate discrete solution in a p th-order accurate
 281 discrete operator $\mathcal{L}_{(p)}(\cdot)$, with $p > q$. This approach is referred to as the defect correction method
 282 in the works of Ervin and Layton [30], and Pierce and Giles [31].

283 Replacing the lower-order discrete operator vector with one of higher order (e.g., fourth-order
 284 accuracy), Eq. (30) becomes

$$\begin{aligned}\bar{\tau}_e^{\phi_{(l)}}(x_e) &= (\mathbf{t}_{e_{(l)}} - \mathbf{t}_{e_{(h)}})\boldsymbol{\phi}_{S_{(l)}} \\ &= u_e \left(\frac{h^2}{8} \phi_e'' \right) - \Gamma_e \left(\frac{h^2}{24} \phi_e''' \right) \\ &= a_1^{\phi_{(l)}}|_e(x_e) h^2,\end{aligned}\quad (38)$$

where

$$\begin{aligned}(\mathbf{t}_{e_{(l)}}^C - \mathbf{t}_{e_{(h)}}^C)\boldsymbol{\phi}_{S_{(l)}} &= u_e [\phi_e^{(p)}(x_e) - \phi_e(x_e)] \\ &= u_e \left[\left(\frac{\phi_C + \phi_E}{2} \right) - \phi_e(x_e) \right] \\ &= u_e \left[\left(\frac{\phi_C + \phi_E}{2} \right) - \left(-\frac{1}{16} \phi_W + \frac{9}{16} \phi_C + \frac{9}{16} \phi_E - \frac{1}{16} \phi_{EE} \right) \right] \\ &= u_e \left[\frac{h^2}{8} \phi_e'' \right]\end{aligned}\quad (39a)$$

and

$$\begin{aligned}(\mathbf{t}_{e_{(l)}}^D - \mathbf{t}_{e_{(h)}}^D)\boldsymbol{\phi}_{S_{(l)}} &= \Gamma_e [\phi_e'^{(p)}(x_e) - \phi_e'(x_e)] \\ &= \Gamma_e \left[\left(\frac{\phi_E - \phi_C}{h} \right) - \phi_e'(x_e) \right] \\ &= \Gamma_e \left[\left(\frac{\phi_E - \phi_C}{h} \right) - \left(-\frac{1}{24h} \phi_W - \frac{9}{8h} \phi_C + \frac{9}{8h} \phi_E - \frac{1}{24h} \phi_{EE} \right) \right] \\ &= \Gamma_e \left[\frac{h^2}{24} \phi_e''' \right]\end{aligned}\quad (39b)$$

285 are its (weighted) convective and diffusive components, respectively.

286 Looking back on the estimate derived in Eq. (38), we can see that the expression for $\bar{\tau}_e^{\phi_{(l)}}(x_e)$
 287 is formally equivalent to that of $\bar{\tau}_1^{\phi_{(h)}}|_e(x_e)$ in Eq. (36), with the exception that the facial values
 288 ϕ_e'' and ϕ_e''' are extrapolated from the underlying discrete solution, rather than evaluated using a
 289 higher-order one (see Figs. 7 and 8). This estimate (for the tau-k component of the face truncation
 290 error) is thus less accurate than the one derived in Section 5.3.1, since the higher-order problem
 291 does not need to be solved, only formulated.

292 In addition, even though this error estimator is not able to estimate the tau-0 component
 293 of the face truncation error, we can always confidently approximate the cell discretisation and
 294 truncation errors (note that this component is nil about the cell, see Eq. (34a)). Nevertheless, as
 295 we will see in Section 6.1.1, the tau-k component of this error is, in fact, the source of the (cell)
 296 discretisation error, not the sum of both of the aforementioned components.

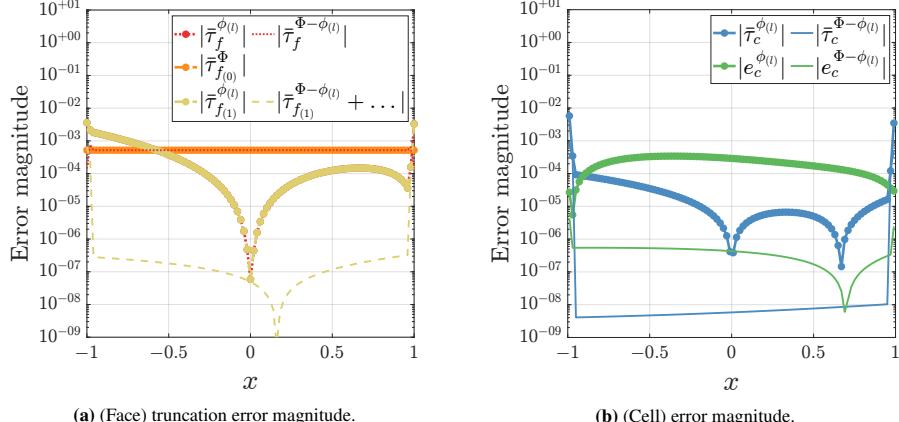


Figure 7. 1D distribution of the magnitude of the error for a uniformly spaced grid with 100 CVs (2nd-order accuracy).

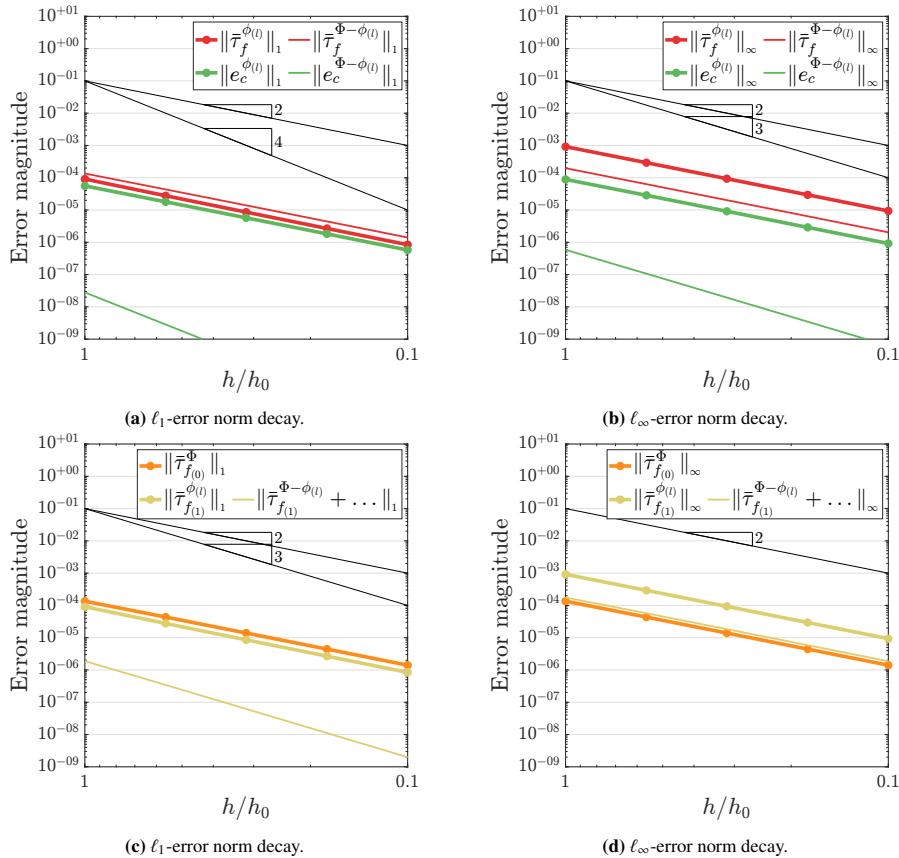


Figure 8. Error norm decay for increasingly refined uniformly spaced grids, with $h_0 = 1e-02$ (2nd-order accuracy).

297 **6. Adaptive mesh refinement**

298 In addition to estimating the discretisation error, we also desire methods for reducing it when
299 it is found to be too large. Adaptive mesh refinement strategies are generally classified as one of
300 the following: *global*, *semi-global*, or *local* refinement.

301 In global p -refinement (also called uniform p -refinement), the degree of every element in the
302 mesh is increased by the same quantity. Even though in most cases this is the simplest strategy
303 to implement, it can be very wasteful in the sense that many elements away from the area(s) of
304 interest are refined. A modification of the global strategy is semi-global p -refinement, in which
305 the degree of every element in one or more selected cross-sections is refined. In certain cases,
306 this strategy may be implemented as easily as global p -refinement and may be less wasteful.
307 However, complications occur when the desired refinement does not correspond to a natural
308 cross-section.

309 The last refinement strategy is (adaptive) local p -refinement, in which only selected elements
310 in restricted portions of the domain are refined.

311 For (adaptive) local p -refinement to be efficient, it is necessary that:

- 312 • Computation is not expensive⁶.
- 313 • Neighbouring elements are *not* too different in degree.
- 314 • The algorithm can be implemented cheaply⁷.

315 While the first and third requirements are natural for any practical algorithm, the main pur-
316 pose of the second one is to smooth the changes of the approximate solution from element to
317 element.

318 **6.1. p -adaptive meshing**

319 An automatic mesh adaptation algorithm should have the capability of detecting important
320 flow features in the same way that the user does and adapt the mesh accordingly. In p -adaptive
321 meshing, we fix the geometry of the grid and achieve a locally better approximate solution by
322 modifying (usually increasing) the local formal order of accuracy of the discretisation scheme
323 (note that since the geometry of the grid remains unchanged upon adaptation, we assume that all
324 the grids we are working with are geometrically admissible).

325 Even though the method proposed here is valid for any unstructured grid, extra care must be
326 taken when dealing with (non-isotropic) irregular grids (see Section 4.3.3). For this reason, in
327 the numerical examples presented in Section 6.2 we simply use a Cartesian grid⁸.

⁶If, upon refinement, the system of equations becomes increasingly dense, then the advantage gained by refining only a small number of elements is partially lost.

⁷If the (adaptive) process cannot be implemented cheaply, then the actual cost of computing an approximate solution to the (partial) differential equation(s) may become higher with (adaptive) local refinement than with global or semi-global refinement, even in cases where they are extremely wasteful.

⁸The ideal mesh is a Cartesian distribution, where all the points are equidistant and where all cells are perfect cubes, with $\Delta x = \Delta y = \Delta z$. This grid is associated with the highest possible accuracy of the discretised formulas, where the FVM leads to the same formulas as FDs.

328 6.1.1. Adaptation driver

329 Recall that in Section 5.2 we derived the linear discrete form of the (discretisation) error
 330 transport equation (28a). This equation is repeated here for convenience

$$\bar{\tau}^\Phi = -\mathcal{L}_{(p)}(\mathbf{e}^\Phi), \quad (40)$$

331 where $\mathcal{L}_{(p)}(\cdot)$ is the p th-order accurate discrete form of the symbolic differential operator $\mathcal{L}(\cdot)$.
 332 According to this equation, the truncation error serves as the local source of the discretisation
 333 error [32], and thus reducing the latter should result in a commensurate reduction of the latest.

334 However, as we have seen in section Section 5.2.1, the tau-0 component of the face truncation
 335 error does not *directly* contribute to the cell discretisation and truncation errors (see Eq. (34a)),
 336 and thus we should investigate whether the face truncation (or tau) error in Eq. (33) is the rightful
 337 criterion to drive the (adaptive) process, or if it is perhaps its tau-k component alone.

338 **Example 2.** Consider the 1D steady-state form of Eq. (2) initially discretised with a second-
 339 order accurate scheme over the domain $[-1, 1]$, where $u = 1$ [m/s] and $\Gamma = 1$ [m^2/s]. Dirichlet
 340 conditions were prescribed for both boundaries, and the (manufactured) solution to this equation
 341 was assumed to be the one used in the previous example (see Eq. (35)).

342 The 1D distribution of the magnitude of the error before refinement for the uniformly spaced
 343 grid with 100 CVs used in Example 1 is shown in Fig. 9.

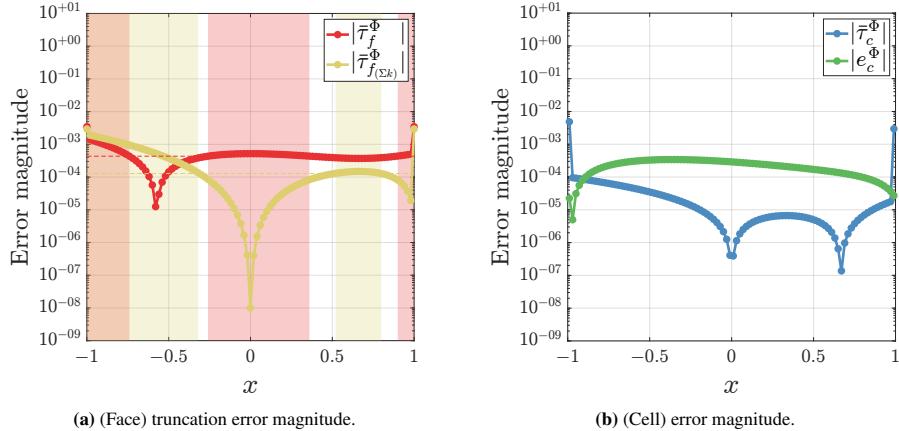


Figure 9. 1D distribution of the magnitude of the error before refinement (2nd-order accuracy).

344 By sorting the magnitude of both criteria (tau and tau-k) in descending order and flagging for
 345 refinement the first $\lambda \sim 50\%$ of elements (or faces) from this list, we compare the 1D distribution
 346 of the magnitude of the error after refinement with the one prior to it (see error thresholds in
 347 Fig. 9a and the respective refinement zones marked in red and yellow, respectively).

348 The procedure detailed here is different from the one proposed by Syrakos et al. [33] in the
 349 sense that we gauge how both criteria fare after the first adaptation cycle and not at the end of the
 350 (adaptive) process. As such, flagged elements (or faces) are refined owing to large error estimates
 351 and *not* because they were in violation of the refinement rules proposed in Section 6.1.3 (recall
 352 that the sole purpose of the adaptation criterion is to pinpoint regions of the domain deemed
 353 inaccurate).

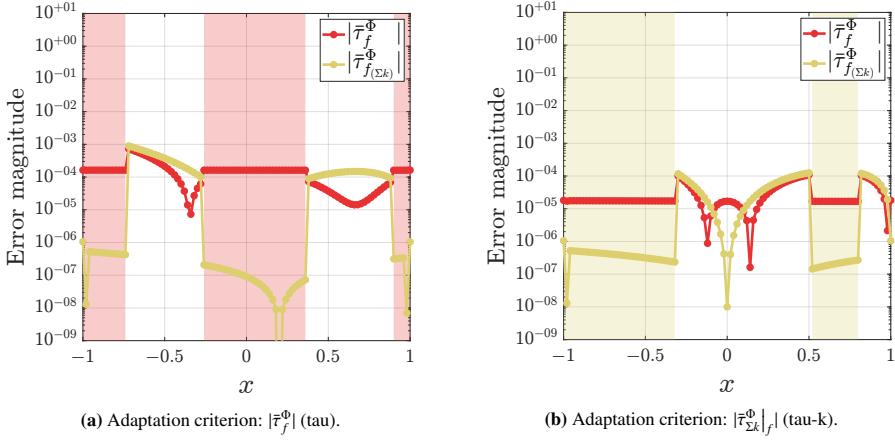


Figure 10. 1D distribution of the magnitude of the (face) truncation error after refinement.

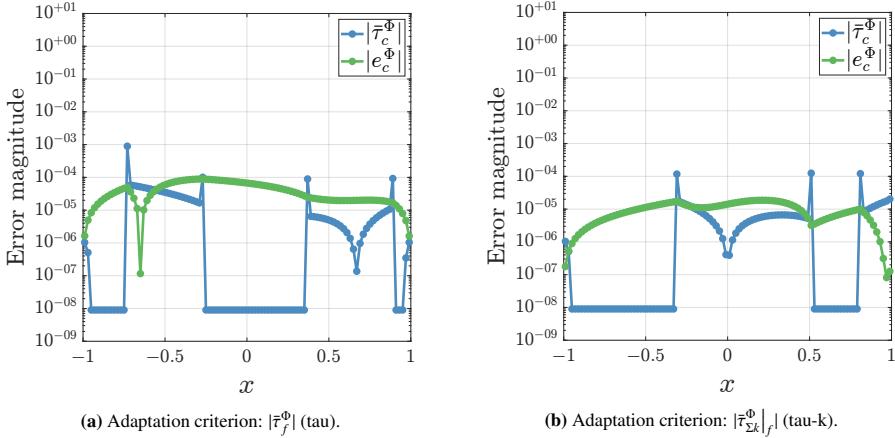


Figure 11. 1D distribution of the magnitude of the (cell) error after refinement.

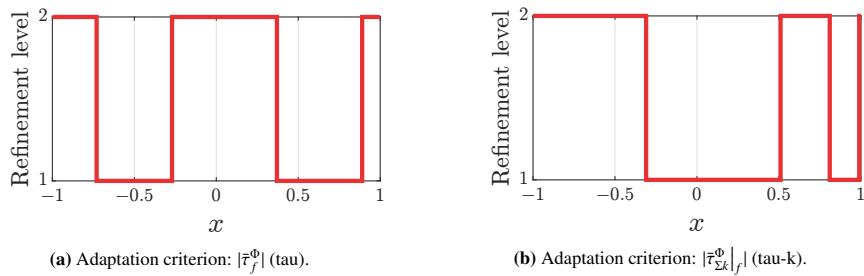


Figure 12. 1D distribution of the refinement level after refinement.

354 To decide which of these is the most appropriate criterion to drive the (adaptive) process,
355 we measure the (analytical) error reduction ratio between the error after (\mathbf{e}) and before (\mathbf{e}_0) the
356 refinement through a global norm over the entire domain Ω .

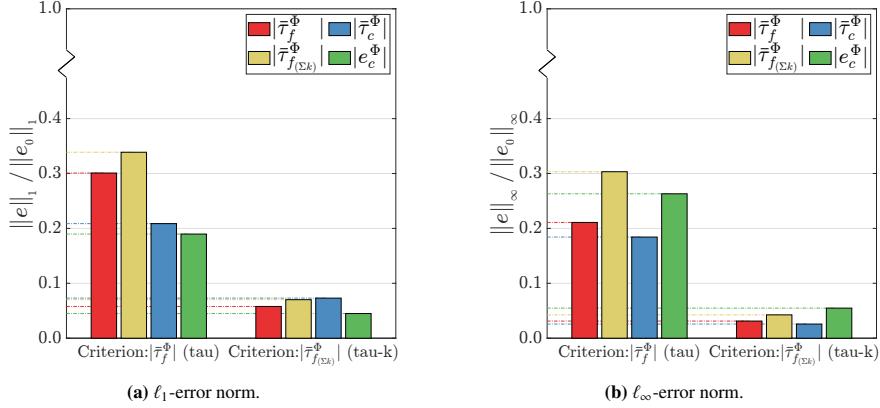


Figure 13. ℓ_1 - and ℓ_∞ -error norm reduction ratios.

As shown in Fig. 13, since the sub-problem on the right (10b, 11b and 12b) produces a significantly greater reduction of the (cell) discretisation error, we will use the tau-k component of the face truncation error alone (or perhaps its estimate) to drive the (adaptive) process.

6.1.2. Refinement mechanics

Almost all adaptation methods proceed from the fundamental premise of error *equidistribution* [34]. Equidistribution aims at producing a mesh that contains the same value of (adaptive) refinement parameter (or criterion) over the entire domain, thereby improving the accuracy of the solution. Since this parameter (or criterion) is a stand-in for the face truncation error, this goal spreads the (cell) discretisation error out evenly over the entire domain⁹. In practice, because this principle assumes that all errors are equally important, it is typically somewhat over-conservative, and more often than not, this is certainly not the case. However, without any additional guidance about what is important, equidistribution simply ensures that everything is equally correct.

Decision criteria. Bangerth and Rannacher [35] described non-trivial strategies on how to decide the subset of elements (or faces) to be flagged for refinement, based on the adaptation criterion selected in Section 6.1.1.

On the one hand, *fixed-error-reduction* or *fixed-fraction* strategies select subsets of elements whose criteria accumulate to a predefined fraction of their global sum. This strategy is only applicable when the sum of all criteria actually has meaning (e.g., local errors that add up to the global one). Even though this strategy is more sophisticated than the one introduced below, it tends to only refine (or adapt) very few elements whenever smooth rapid solution changes or singularities are encountered.

On the other hand, *fixed-rate* or *fixed-number* strategies select predefined fractions of elements with the highest (or lowest) criterion for adaptation, which not only allows us to control how aggressively each subsequent mesh is refined but also to predict the approximate increase in the computational cost of the (adaptive) process. For these reasons, we will use this strategy in our investigation.

⁹The idealised histogram of the refinement parameter (or criterion) looks like a delta function with height equal to the (total) number of faces.

Stopping criteria. One or more of the following parameters will ultimately define the termination criteria of the (adaptive) process:

- A (remaining) desired error level for the functional of interest.
 - An upper limit on the (total) number of DoFs¹⁰.
 - An upper limit on the (total) number of adaptation cycles.

One additional parameter is the maximum level of refinement (note that the conditioning of Vandermonde matrices becomes worse as the degree of the polynomial approximation increases). Hence, if, upon refinement, the local coefficient matrix becomes singular to working precision, then that element (or face) can no longer be refined.

392 6.1.3. Refinement rules

To achieve the second requirement for efficient (adaptive) local p -refinement introduced at the beginning of Section 6, we propose the following rules inspired by those of the same names in Red-Green (h -) refinement [16]. Much like any other adaptation method, the set of rules proposed here is one that we can only hope will drive the (adaptive) process to approximate the optimal mesh, while *not* compromising error estimation (note that the lack of smoothing between lower- and higher-order regions may lead the algorithm to refine elements (or faces) that would not otherwise be refined had reliable estimates been obtained).

Neighbour rule (p -version). For any element (or face) of degree p to be considered eligible for refinement, all the elements (or faces) that belong to the cell(s) that contain(s) the element (or face) flagged in Fig. 14a should be, at least, of degree p (see Fig. 14b).

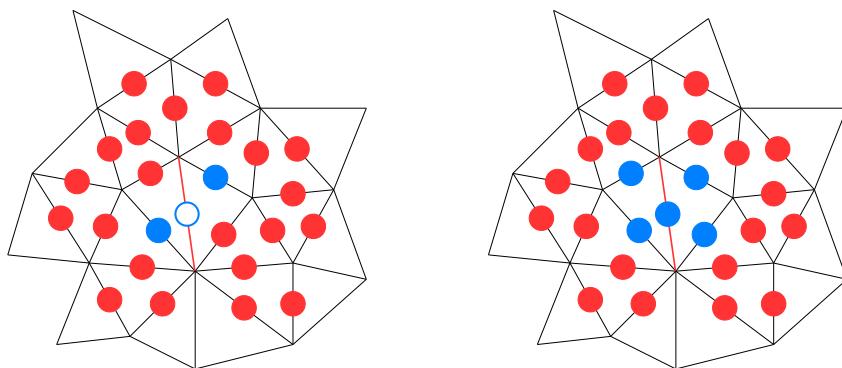


Figure 14. Neighbour rule (refinement level: $\bullet-p_1$, $\bullet-p_2$, with $p_2 \equiv p \equiv p_1 + 1$)

¹⁰The (total) number of DoFs is often used to define a length scale, especially when mesh adaptation is performed by modifying the local formal order of accuracy of the discretisation scheme without changing the geometry of the grid. The grid size is related to the number of DoFs through the following equation [36]:

$$h \sim \text{DoFs}^{-1/d}, \quad (41)$$

where d is the dimension of the problem.

403 **1-Irregular rule (p -version).** The difference in degree of elements (or faces) in neighbouring
 404 cells can be, at most, 1; that is, the degree of all the elements (or faces) that belong to the
 405 cell(s) that contain(s) an element (or face) that shares a cell with the flagged element (or face)
 406 should be, at least, of degree p (see Fig. 15b). Note that this rule is complementary to the previous
 407 one, i.e., it provides an extra layer of smoothing.

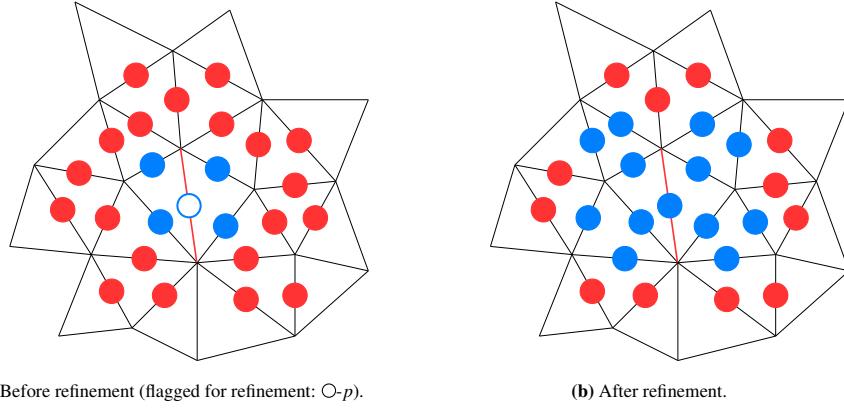


Figure 15. 1-Irregular rule (refinement level: $\bullet-p_1$, $\bullet-p_2$, with $p_2 \equiv p = p_1 + 1$).

408 6.2. Numerical examples

409 In this section, we ultimately seek to answer the following set of questions:

- 410 • Are the proposed error estimators able to accurately drive the (adaptive) process?
 - 411 – If so, how does the distribution of the refinement level using the analytical solution
 compare to that predicted by the aforementioned error estimators?
 - 412 – If so, how do the magnitude of the analytical discretisation and truncation errors
 compare to that predicted by these estimators?
- 415 • Does the (adaptive) process effectively reduce the spatial discretisation error?
 - 416 – If so, how efficiently (i.e., how does it compare to global (p -) refinement)?

Example 3. Consider the 2D steady-state form of Eq. (2) initially discretised with a second-order accurate scheme over the domain $[-1, 1] \times [-1, 1]$, where $(u, v) = (1, 1)$ [m/s] and $(\Gamma_x, \Gamma_y) = (1, 1)$ [m^2/s]. Dirichlet conditions were prescribed for all the boundaries, and the (manufactured) solutions to this equation were assumed to be:

$$\Phi_a(x, y) = \exp(-100x^2y^2) \quad (42a)$$

$$\Phi_b(x, y) = \exp(-100x^2y^2) + \sqrt{(x - \pi/8)^2 + y^2} + \sqrt{x^2 + (y - \pi/8)^2}. \quad (42b)$$

417 In addition, the (adaptive) process was run under the following conditions:

- 418 • Fixed-rate (or fixed-number) parameter: $\lambda = 10\%$.
- 419 • Stopping criterion: computational cost ratio increase¹¹ of (approximately) 3.

420 The 2D distribution of the magnitude of the analytical scalar property and respective volumetric source (or sink) term is shown in Figs. 16 and 17, respectively (note that the white contours lines correspond to regions where the respective functions have a constant value equal to the ticks of the colour bar).

424 While the smooth rapid change in the middle of the domain makes the solution on the left (16a and 17a) tailor-made for p -adaptation, the addition of singularities at $(\pi/8, 0)$ and $(0, \pi/8)$ (16b and 17b) has the opposite effect (recall that the partial derivatives of $\Phi_b(x, y)$ with respect to x and y are respectively unbounded at these points).

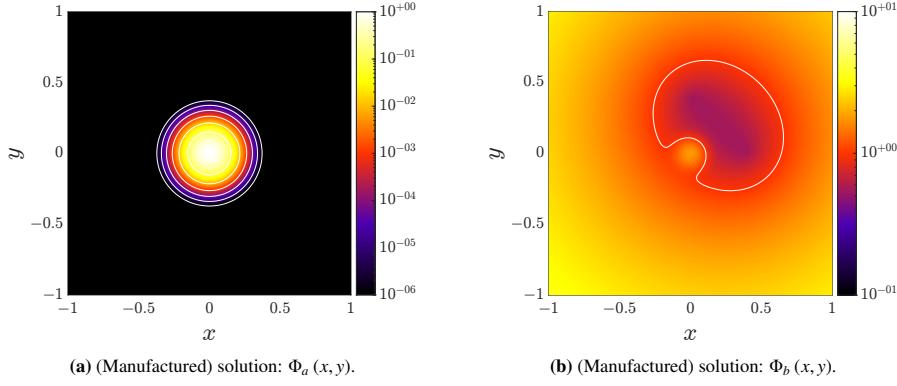


Figure 16. 2D distribution of the magnitude of the analytical scalar property.

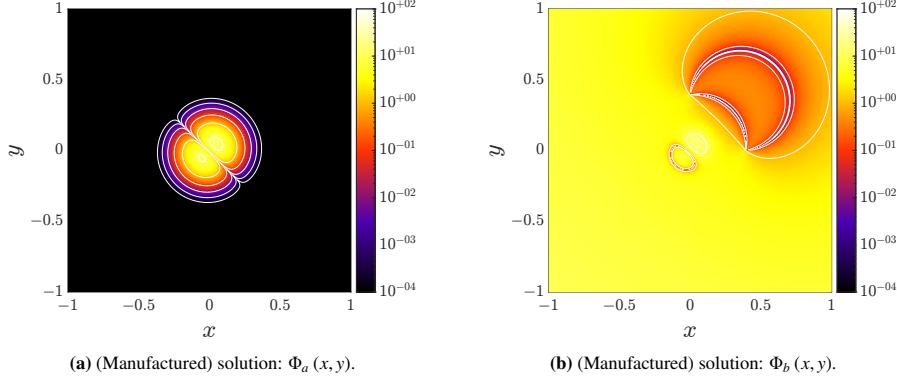


Figure 17. 2D distribution of the magnitude of volumetric source (or sink) term.

428 The 2D distribution of the magnitude of the analytical (cell) discretisation and truncation errors for these solutions in a Cartesian grid with $10k$ CVs ($h = 1e-02$) is shown in Figs. 18 and 19, respectively.

431 As expected, the regions that contain elements (or faces) near the origin of the exponential and two singularities are the ones where the greater error magnitude occurs (see Figs. 18a and 18b), and so it is expected that the algorithm will prioritise their refinement.

¹¹The computational cost increase (or “gain”) for the n th adaptation cycle refers to the ratio between the number of DoFs of cycle n and the number of DoFs of the uniform 2nd-order accurate case.

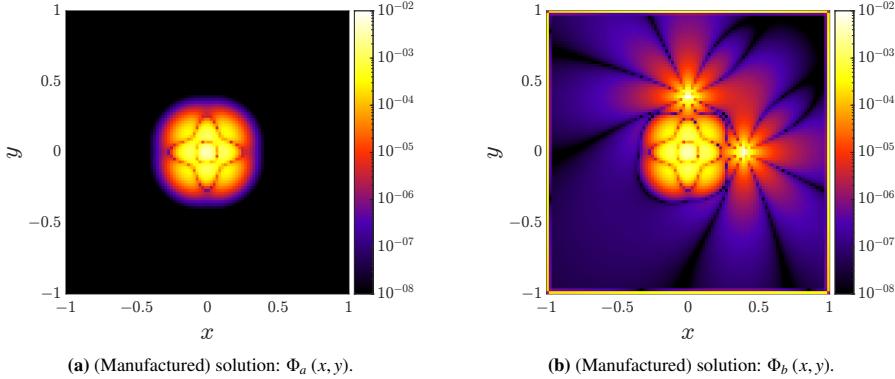


Figure 18. 2D distribution of the magnitude of the analytical (cell) truncation error (2nd-order accuracy).

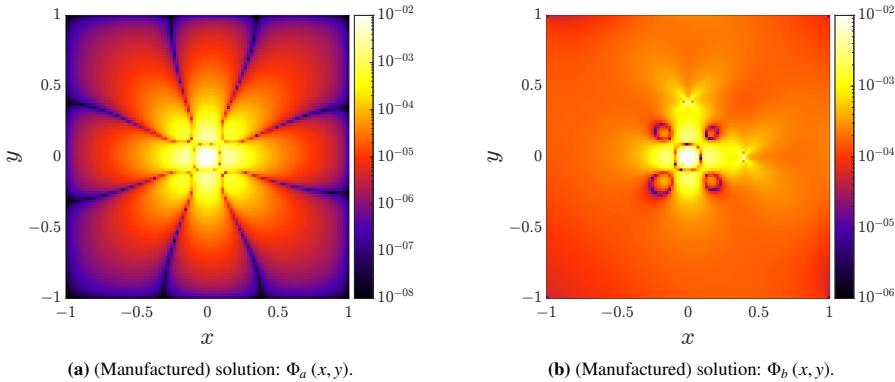


Figure 19. 2D distribution of the magnitude of the analytical (cell) discretisation error (2nd-order accuracy).

434 The 2D distribution of the refinement level (in the last adaptation cycle) for the cases where
 435 the value of the adaptation criterion is evaluated using the analytical solution or estimated using
 436 the two error estimators derived in Sections 5.3.1 and 5.3.2 is shown in Figs. 20 to 22, respec-
 437 tively.

438 From a distribution point of view, it is clear that both estimators were able to reliably identify
 439 where further resolution was needed, even for the case on the right ($\Phi_b(x, y)$), where one may
 440 have initially thought that the two singularities would have prevented the algorithm from doing
 441 SO.

442 While for the case on the left (20a, 21a and 22a) only the region near the origin of the expo-
 443 nential is targeted for refinement, the one on the right (20b, 21b and 22b) targets both this region
 444 and the one near the two singularities. However, if we were to instead employ a fixed-error-
 445 reduction (or fixed-fraction strategy), the algorithm would eventually stop flagging elements (or
 446 faces) near the origin of the exponential and strictly flag the ones near the two singularities.
 447 Nevertheless, since the proposed method is not able to reduce the magnitude of the error near
 448 discontinuities and/or singularities, the only meaningful difference between these strategies is
 449 that the latest would become less wasteful.

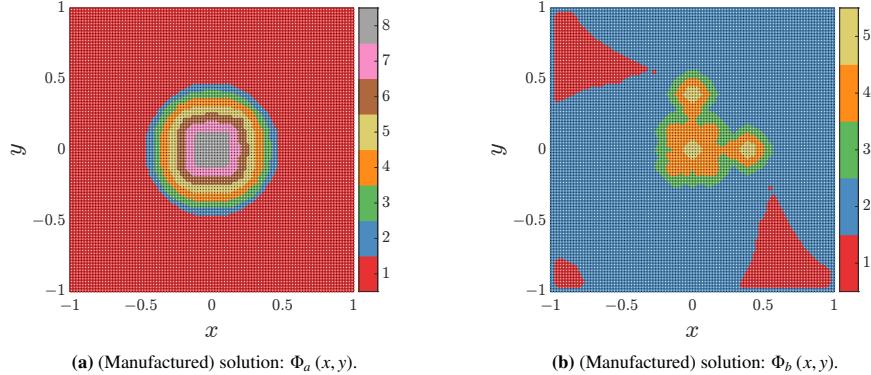


Figure 20. 2D distribution of the refinement level using the analytical solution.

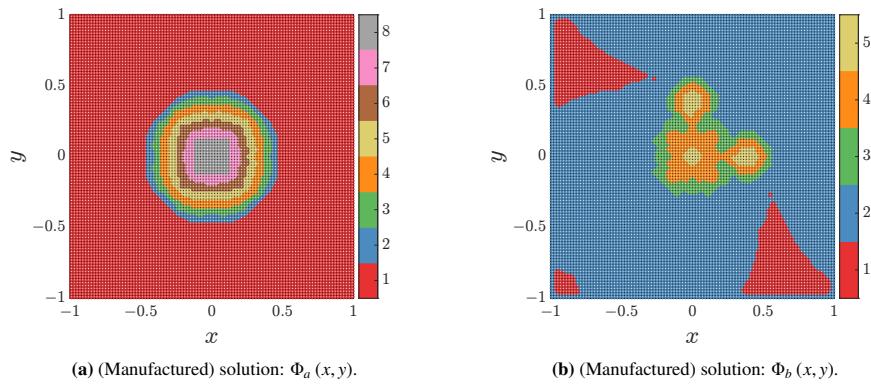


Figure 21. 2D distribution of the refinement level using the estimate provided by the error estimator in Section 5.3.1.

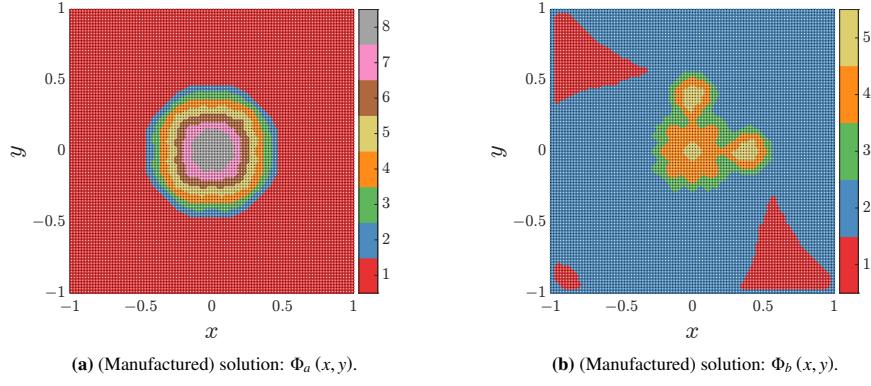


Figure 22. 2D distribution of the refinement level using the estimate provided by the error estimator in Section 5.3.2.

Finally, we present the cycle-by-cycle evolution of both the adaptation criteria and the discretisation error (see Figs. 23 to 26). In these figures, we denote by Φ , $\phi_{(h)}$ and $\phi_{(l)}$ the three cases in Figs. 20 to 22, respectively, and by $\hat{\mathbf{p}}_k$ the analytical uniform k -degree accurate response.

453 As for the first case (23 and 24), we can see that the (adaptive) process is able to reduce the
 454 spatial discretisation error to similar levels to that of a uniform 14th-order accurate scheme (\hat{p}_{13})
 455 for a computational cost inferior to its 6th-order accurate counterpart (\hat{p}_5), i.e., at roughly $\sim 15\%$
 456 of the cost (note that it only makes sense to use methods of this high of an order if the asymptotic
 457 range has been reached and provided that the round-off error is small enough to where it does
 458 not pollute the solution).

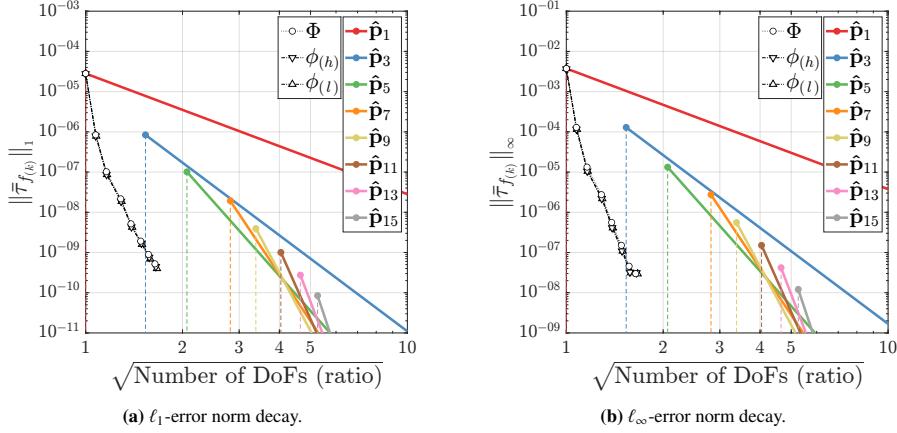


Figure 23. ℓ_1 - and ℓ_∞ -(face) truncation error (tau-k) norm decay for $\Phi_a(x, y)$.

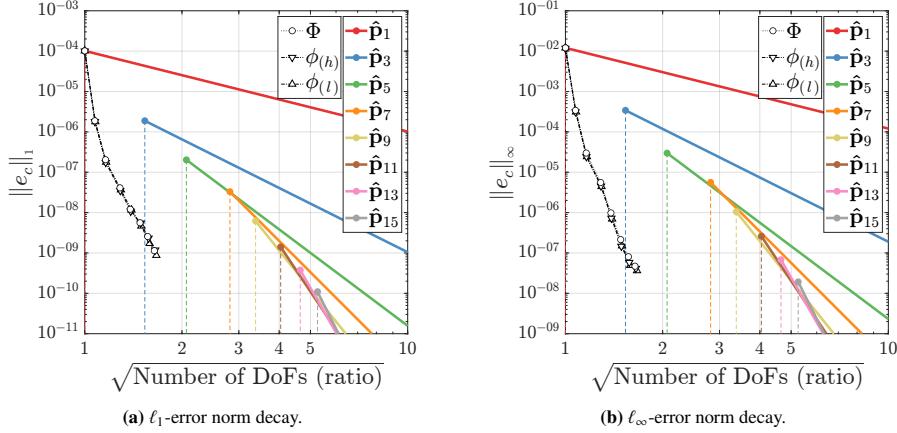


Figure 24. ℓ_1 - and ℓ_∞ -(cell) discretisation error norm decay for $\Phi_a(x, y)$.

459 As for the second case (25 and 26), unlike the previous one, neither estimator was able to
 460 provide reliable estimates for the magnitude of the error. In fact, this was to be expected, as
 461 the discretisation schemes were not formulated to handle discontinuities and/or singularities.
 462 Once the algorithm starts targeting the region of the domain that contains the two singularities
 463 and the magnitude of the error in this region surpasses the one that contains the origin of the
 464 exponential (i.e., after the first adaptation cycle), both estimators appear to grossly underestimate
 465 the magnitude of the *true* error, thereby misleading the user into thinking that the (adaptive)
 466 process is accomplishing something that it was not meant to: reduce the magnitude of the error.

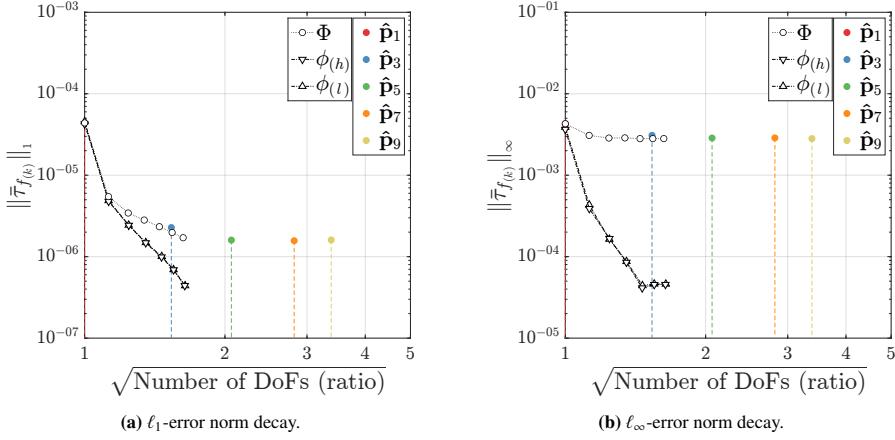


Figure 25. ℓ_1 - and ℓ_∞ -(face) truncation error (tau-k) norm decay for $\Phi_b(x,y)$.

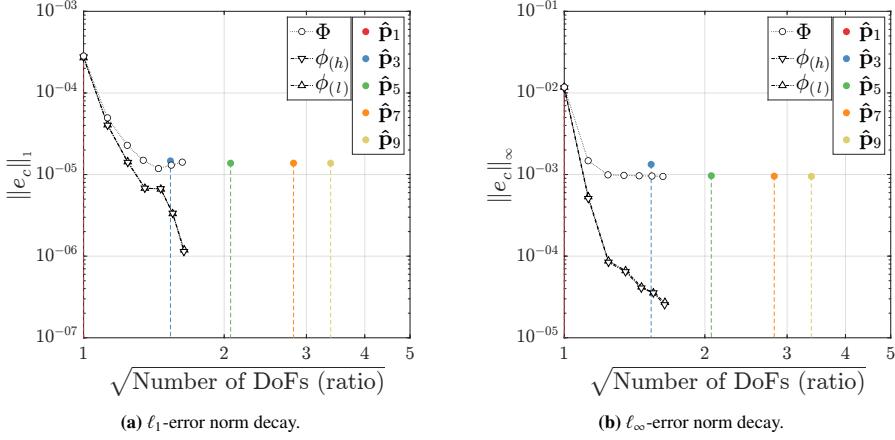


Figure 26. ℓ_1 - and ℓ_∞ -(cell) discretisation error norm decay for $\Phi_b(x,y)$.

467 7. Conclusions

468 The main conclusions drawn from the work developed in this paper include:

- 469 • In Section 4, we used a least-squares approach to approximate the local underlying solution
 470 about the centroid of the face. Using least-squares methods to compute the coefficients
 471 of the polynomial regression model [37] offers more flexibility with regard to the order of
 472 accuracy achieved [38] and the discretisation stencil used [39]. Nevertheless, this flexibil-
 473 ity comes at a cost, as proper weighting is needed (note that the inverse-distance weighting
 474 function in Eq. (15) is simply an “educated guess”; the *optimal* choice of weights should
 475 be made on the basis of the element residuals. However, since we can only compute them
 476 once the solution to the (linear) system of equations in Eq. (23) is obtained, this would
 477 make for an iterative process).

- 478 • In Section 5.2, we used the discrete form of the (discretisation) error transport equation
 479 to derive two distinct *a posteriori* error estimators, which were then used to drive the
 480 (adaptive) process in Section 6.
- 481 – Even though both estimators were able to reliably pinpoint regions of the domain
 482 deemed inaccurate, they grossly underestimated the magnitude of both the (face)
 483 truncation and (cell) discretisation errors in problems with singularities.
- 484 – Seeing that both estimators yielded similar results for the numerical examples pre-
 485 sented in Section 6.2, we can conclude that the tau-extrapolation estimate derived in
 486 Section 5.3.2 is computationally more attractive since the higher-order problem does
 487 not need to be solved, only formulated.
- 488 • As far as the effectiveness of the (adaptive) process goes, we were able to corroborate our
 489 initial assumption that, despite having proven accurate for smooth problems, order-based
 490 adaptation (or p -adaptation) is typically not as successful when applied in the presence of
 491 discontinuities and/or singularities [4].
- 492 – The work of Nguyen [40] in a FE framework suggests that hp -adaptive methods pro-
 493 vide the best approach to solving this class of problems, where one can use smaller
 494 elements of lower degree near the discontinuities and/or singularities and bigger ele-
 495 ments of higher degree elsewhere.
- 496 • The present study is only concerned with the steady-state case. Transient problems would
 497 come with the following additional complexities:
- 498 – The truncation error would contain terms that are a function of both the grid spacing
 499 (h) and the time step (Δt), and thus the general expressions of the error estimators de-
 500 rived in Sections 5.3.1 and 5.3.2 should be reformulated to account for this additional
 501 component (for further reference, see [41]).
- 502 – The (adaptive) process should not be re-run for each time step. Instead, it should be
 503 run once and coarsened (or unrefined) and refined simultaneously for the subsequent
 504 ones, so as to not drastically increase the computational cost of the initial problem.

505 **Declaration of competing interest**

506 The authors declare that they have no known competing financial interests or personal rela-
 507 tionships that could have appeared to influence the work reported in this paper.

508 **Data availability**

509 No data was used for the research described in this article.

510 **Acknowledgments**

511 The authors would like to acknowledge the financial support received by FCT under the
 512 research project: Higher-order Immersed Boundary Method for Moving Body Problems (HIB-
 513 forMBP) with the reference PTDC/EME-EME/32315/2017. Institutional acknowledgment: This
 514 work was supported by FCT, through IDMEC, under LAETA, project UIDB/50022/2020.

515 **Appendix A. Least-squares fitting**

516 The term *least-squares* describes a frequently used approach to solving overdetermined (or
 517 inexactly specified) systems of equations in an approximate sense; i.e., rather than solving the
 518 equations exactly, we seek to minimise the “energy” (or “sum of squares”) of the residual.

519 *Appendix A.1. Unconstrained least-squares (ULS) formulation*

520 Consider the following (linear) system of equations

$$\mathbf{Ax} = \mathbf{b}, \quad (\text{A.1})$$

521 where $\mathbf{A} \in \mathbb{R}^{m \times n}$ ($m \geq n$) has full (column) rank, $\mathbf{x} \in \mathbb{R}^n$, and $\mathbf{b} \in \mathbb{R}^m$.

522 If there is no solution to this system, we can always solve it in the “least-squares” sense. In
 523 other words, if a given system with more equations than unknowns cannot be solved because \mathbf{b}
 524 does not lie in the *range* of \mathbf{A} , then the best approximation is one that minimises the “sum of
 525 squares” of the residuals ($\mathbf{r} = \mathbf{b} - \mathbf{Ax}$):

$$\begin{aligned} \min_{\mathbf{x}} J(\mathbf{x}) &= \min_{\mathbf{x}} \|\mathbf{b} - \mathbf{Ax}\|_2^2 \\ &\equiv \min_{\mathbf{x}} \left(\sum_{i=1}^m (b_i - \mathbf{a}_i^\top \mathbf{x})^2 \right). \end{aligned} \quad (\text{A.2})$$

526 Expanding the gradient of $J(\mathbf{x})$

$$\begin{aligned} \nabla_{\mathbf{x}} J(\mathbf{x}) &= \nabla_{\mathbf{x}} (\mathbf{b}^\top \mathbf{b} - 2\mathbf{b}^\top \mathbf{Ax} + \mathbf{x}^\top \mathbf{A}^\top \mathbf{Ax}) \\ &= \underbrace{\nabla_{\mathbf{x}} (\mathbf{b}^\top \mathbf{b})}_{0} - 2\nabla_{\mathbf{x}} (\mathbf{b}^\top \mathbf{Ax}) + \nabla_{\mathbf{x}} (\mathbf{x}^\top \mathbf{A}^\top \mathbf{Ax}) \\ &= -2\mathbf{A}^\top \mathbf{b} + [\mathbf{A}^\top \mathbf{A} + (\mathbf{A}^\top \mathbf{A})^\top] \mathbf{x} \\ &= -2\mathbf{A}^\top \mathbf{b} + 2\mathbf{A}^\top \mathbf{Ax}, \end{aligned} \quad (\text{A.3})$$

527 and setting it to zero gives

$$\mathbf{A}^\top \mathbf{Ax}^* = \mathbf{A}^\top \mathbf{b}, \quad (\text{A.4})$$

528 which is a symmetric (linear) system of equations known as the *normal equations*.

529 Since \mathbf{A} has full (column) rank (and provided that \mathbf{b} does not lie in the null space $N(\mathbf{A}^*)$), we
 530 can now solve Eq. (A.4) for a unique solution \mathbf{x}^* as

$$\mathbf{x}^* = (\mathbf{A}^\top \mathbf{A})^{-1} \mathbf{A}^\top \mathbf{b}. \quad (\text{A.5})$$

531 In this derivation, we have used the condition $\nabla_{\mathbf{x}} J(\mathbf{x}) = \mathbf{0}$, which is a *necessary* but not *sufficient*
 532 condition for optimality. We found a critical point, but in general, such a point could be a
 533 local minimum, a local maximum, or a saddle point. Fortunately, the objective function is *convex*,
 534 which means that any critical point is indeed a local minimum. To show that $J(\mathbf{x})$ is convex,
 535 it suffices to compute its Hessian and show that it is positive semi-definite:

$$\begin{aligned} \forall \mathbf{x}, \mathbf{x}^\top (2\mathbf{A}^\top \mathbf{A}) \mathbf{x} &= 2(\mathbf{Ax})^\top \mathbf{Ax} \\ &= 2\|\mathbf{Ax}\|_2^2 \geq 0. \end{aligned} \quad (\text{A.6})$$

536 *Appendix A.1.1. Problem conditioning*

537 Even though we have presented the normal equations as the most straightforward approach
 538 to solving overdetermined systems of equations, there are several undesirable aspects of this
 539 theoretical formulation (note that the normal equations will always be more poorly conditioned
 540 than the original system in Eq. (A.1)). In fact, (the ℓ_2 -norm of) the condition number is squared:

$$\kappa(\mathbf{A}^\top \mathbf{A}) = \kappa(\mathbf{A})^2. \quad (\text{A.7})$$

541 With finite-precision computation, the normal equations can actually become singular and
 542 $\mathbf{A}^\top \mathbf{A}$ non-existent. This inconvenience does not make the method of normal equations useless
 543 but provides motivation to seek more robust methods to solve least-squares problems.

544 *Appendix A.1.2. Solving the unconstrained least-squares (ULS) problem via QR decomposition*

545 In this subsection, we consider the QR decomposition of $\mathbf{A} \in \mathbb{R}^{m \times n}$ into a product of an
 546 orthogonal matrix $\mathbf{Q} \in \mathbb{R}^{m \times m}$ and an upper triangular matrix $\mathbf{R} \in \mathbb{R}^{m \times n}$ as

$$\mathbf{A} = \mathbf{QR} = \left[\begin{array}{c|c} \tilde{\mathbf{Q}} & \tilde{\mathbf{Q}}_\perp \end{array} \right] \left[\begin{array}{c} \tilde{\mathbf{R}} \\ \mathbf{0} \end{array} \right] = \tilde{\mathbf{Q}} \tilde{\mathbf{R}}, \quad (\text{A.8})$$

547 where the columns of $\tilde{\mathbf{Q}}_\perp$ span a vector space that is complementary and orthogonal to that
 548 spanned by $\tilde{\mathbf{Q}}$.

549 The ULS problem in Eq. (A.2) may now be written as follows:

$$\begin{aligned} \min_{\mathbf{x}} \|\mathbf{b} - \mathbf{Ax}\|_2^2 &= \min_{\mathbf{x}} \|\mathbf{Q}^\top(\mathbf{b} - \mathbf{Ax})\|_2^2 \\ &= \min_{\mathbf{x}} \left\| \left[\begin{array}{c} \tilde{\mathbf{Q}}^\top \\ \tilde{\mathbf{Q}}_\perp^\top \end{array} \right] \mathbf{b} - \left[\begin{array}{c} \tilde{\mathbf{R}} \\ \mathbf{0} \end{array} \right] \mathbf{x} \right\|_2^2 \\ &= \min_{\mathbf{x}} \|\tilde{\mathbf{Q}}^\top \mathbf{b} - \tilde{\mathbf{R}} \mathbf{x}\|_2^2 + \|\tilde{\mathbf{Q}}_\perp^\top \mathbf{b}\|_2^2 \\ &\geq \min_{\mathbf{x}} \|\tilde{\mathbf{Q}}^\top \mathbf{b}\|, \end{aligned} \quad (\text{A.9})$$

550 where $\tilde{\mathbf{Q}}^\top \mathbf{b}$ is the minimum residual norm of $\|\mathbf{b} - \mathbf{Ax}\|_2$.

551 Since \mathbf{A} has full (column) rank, then $\tilde{\mathbf{R}}$ is non-singular, and thus we can solve the triangular
 552 system $\tilde{\mathbf{R}} \mathbf{x}^* = \tilde{\mathbf{Q}}^\top \mathbf{b}$ for a unique solution $\mathbf{x}^* = \tilde{\mathbf{R}}^{-1} \tilde{\mathbf{Q}}^\top \mathbf{b}$.

553 *Appendix A.2. Constrained least-squares (CLS) formulation*

554 The (linearly) CLS problem refers to the problem of finding a “least-squares” solution that
 555 exactly satisfies q additional constraints of the form

$$\mathbf{Cx} = \mathbf{d}, \quad (\text{A.10})$$

556 where $\mathbf{C} \in \mathbb{R}^{q \times n}$ and $\mathbf{d} \in \mathbb{R}^q$.

557 Minimisation with constraints can be done using the method of Lagrange multipliers. Consider
 558 the Lagrangian function $\mathcal{L}(\mathbf{x}, \boldsymbol{\mu})$ defined as

$$\mathcal{L}(\mathbf{x}, \boldsymbol{\mu}) = \|\mathbf{b} - \mathbf{Ax}\|_2^2 + \boldsymbol{\mu}^\top (\mathbf{Cx} - \mathbf{d}), \quad (\text{A.11})$$

where $\mu \in \mathbb{R}^q$ is a set of Lagrange multipliers. This method tells us that if \mathbf{x}^* is a solution to the CLS problem, then there is a set of Lagrange multipliers μ^* that satisfies

$$\nabla_{\mathbf{x}} \mathcal{L}(\mathbf{x}, \mu) = \mathbf{0} \quad (\text{A.12a})$$

$$\nabla_{\mu} \mathcal{L}(\mathbf{x}, \mu) = \mathbf{0}. \quad (\text{A.12b})$$

559 These are the *optimality conditions* for the CLS problem; any feasible solution must satisfy them.
560 Combining this set of (linear) equations, we can rewrite the optimality conditions as

$$\begin{bmatrix} 2\mathbf{A}^T \mathbf{A} & \mathbf{C}^T \\ \mathbf{C} & \mathbf{0} \end{bmatrix} \begin{bmatrix} \mathbf{x}^* \\ \mu^* \end{bmatrix} = \begin{bmatrix} 2\mathbf{A}^T \mathbf{b} \\ \mathbf{d} \end{bmatrix}. \quad (\text{A.13})$$

561 The $(n+q) \times (n+q)$ matrix on the LHS of Eq. (A.13) is called the KKT matrix. It is in-
562 vertible if and only if \mathbf{C} has linearly independent rows and the stacked matrix $\begin{bmatrix} \mathbf{A} \\ \mathbf{C} \end{bmatrix}$ has linearly
563 independent columns. While the first condition requires \mathbf{C} to be wide (or square), i.e., that there
564 are fewer constraints than variables, the second one depends on both \mathbf{A} and \mathbf{C} , and it can be
565 satisfied even when the columns of \mathbf{A} are linearly dependent.

566 *Appendix A.2.1. Solving the constrained least-squares (CLS) problem via QR decomposition*

567 Assuming that \mathbf{C} has linearly independent rows and the stacked matrix $\begin{bmatrix} \mathbf{A} \\ \mathbf{C} \end{bmatrix}$ has linearly
568 independent columns, we start by rewriting Eq. (A.13) as

$$\begin{bmatrix} \mathbf{A}^T \mathbf{A} + \mathbf{C}^T \mathbf{C} & \mathbf{C}^T \\ \mathbf{C} & \mathbf{0} \end{bmatrix} \begin{bmatrix} \mathbf{x}^* \\ \nu^* \end{bmatrix} = \begin{bmatrix} \mathbf{A}^T \mathbf{b} \\ \mathbf{d} \end{bmatrix}, \quad (\text{A.14})$$

569 where $\nu^* = \frac{1}{2}\mu^* - \mathbf{d}$.

570 Next, we use QR factorisation

$$\tilde{\mathbf{Q}}\tilde{\mathbf{R}} = \begin{bmatrix} \mathbf{A} \\ \mathbf{C} \end{bmatrix} = \begin{bmatrix} \tilde{\mathbf{Q}}_1 \\ \tilde{\mathbf{Q}}_2 \end{bmatrix} \tilde{\mathbf{R}} \quad (\text{A.15})$$

571 to simplify Eq. (A.14)

$$\begin{bmatrix} \tilde{\mathbf{R}}^T \tilde{\mathbf{R}} & \tilde{\mathbf{R}}^T \tilde{\mathbf{Q}}_2^T \\ \tilde{\mathbf{Q}}_2^T \tilde{\mathbf{R}} & \mathbf{0} \end{bmatrix} \begin{bmatrix} \mathbf{x}^* \\ \nu^* \end{bmatrix} = \begin{bmatrix} \tilde{\mathbf{R}}^T \tilde{\mathbf{Q}}_1^T \mathbf{b} \\ \mathbf{d} \end{bmatrix}. \quad (\text{A.16})$$

572 Multiplying the first set of m equations by $\tilde{\mathbf{R}}^{-T}$ and rewriting Eq. (A.16) in terms of $\mathbf{y}^* = \tilde{\mathbf{R}}\mathbf{x}^*$
573 gives

$$\begin{bmatrix} \mathbf{I} & \tilde{\mathbf{Q}}_2^T \\ \tilde{\mathbf{Q}}_2 & \mathbf{0} \end{bmatrix} \begin{bmatrix} \mathbf{y}^* \\ \nu^* \end{bmatrix} = \begin{bmatrix} \tilde{\mathbf{Q}}_1^T \mathbf{b} \\ \mathbf{d} \end{bmatrix}. \quad (\text{A.17})$$

574 Substituting the first set of m equations of Eq. (A.17) into $\tilde{\mathbf{Q}}_2 \mathbf{y}^* = \mathbf{d}$ gives the following set
575 of equations in ν^* :

$$\tilde{\mathbf{Q}}_2 \tilde{\mathbf{Q}}_2^T \nu^* = \tilde{\mathbf{Q}}_2 \tilde{\mathbf{Q}}_1^T \mathbf{b} - \mathbf{d}. \quad (\text{A.18})$$

576 Next, we use QR factorisation

$$\tilde{\mathbf{Q}}_2^T = \tilde{\mathbf{T}}\tilde{\mathbf{U}} \quad (\text{A.19})$$

577 to simplify Eq. (A.18) and solve for ν^* as

$$\nu^* = \tilde{\mathbf{U}}^{-1} [\tilde{\mathbf{T}}^\top \tilde{\mathbf{Q}}_1^\top \mathbf{b} - \tilde{\mathbf{U}}^{-\top} \mathbf{d}]. \quad (\text{A.20})$$

578 Plugging this result into $\mathbf{y}^* + \tilde{\mathbf{Q}}_2^\top \nu^* = \tilde{\mathbf{Q}}_1^\top \mathbf{b}$ and solving $\mathbf{y}^* = \tilde{\mathbf{R}} \mathbf{x}^*$ for a unique solution \mathbf{x}^* we
579 get:

$$\mathbf{x}^* = \tilde{\mathbf{R}}^{-1} \{ [\tilde{\mathbf{Q}}_1^\top - \tilde{\mathbf{Q}}_2^\top (\tilde{\mathbf{U}}^{-1} \tilde{\mathbf{T}}^\top) \tilde{\mathbf{Q}}_1^\top] \mathbf{b} + (\tilde{\mathbf{U}}^{-1} \tilde{\mathbf{T}}^\top)^\top \mathbf{d} \}. \quad (\text{A.21})$$

580 Appendix B. Polynomial fit standardisation

581 When the \mathbf{X} - (independent data) values are large and the degree of the polynomial approxi-
582 mation is high enough to produce numbers with drastically different orders of magnitude in the
583 Vandermonde matrix (which is constructed during the fitting procedure), scaling problems may
584 arise. To improve the precision of the computed coefficients (or parameters), the columns of the
585 Vandermonde matrix can be brought to the same order of magnitude by centring the \mathbf{X} -data at
586 zero mean and scaling it to unit standard deviation:

$$\hat{\mathbf{X}} = \frac{\mathbf{X} - \bar{\mathbf{X}}}{\sigma}, \quad (\text{B.1})$$

587 where $\bar{\mathbf{X}}$ and σ are the mean and standard deviation of the original \mathbf{X} -values, respectively.

588 2D polynomial regression models for curves are given by

$$\phi(x, y) = \sum_{i=0}^n \sum_{j=0}^{n-i} c_{ij} x^i y^j, \quad (\text{B.2})$$

589 where n is the degree of the polynomial fit.

590 Writing the polynomial function $\hat{\phi}(x, y)$ in terms of (\hat{x}, \hat{y}) and of the standardised coefficients
591 (or parameters) $\hat{\mathbf{c}}$ gives

$$\begin{aligned} \hat{\phi}(x, y) &= \sum_{i=0}^n \sum_{j=0}^{n-i} \hat{c}_{ij} \hat{x}^i \hat{y}^j \\ &= \sum_{i=0}^n \sum_{j=0}^{n-i} \hat{c}_{ij} \left(\frac{x - \bar{x}}{\sigma_x} \right)^i \left(\frac{y - \bar{y}}{\sigma_y} \right)^j \\ &= \sum_{i=0}^n \sum_{j=0}^{n-i} \frac{\hat{c}_{ij}}{(\sigma_x)^i (\sigma_y)^j} \sum_{k=0}^i \sum_{l=0}^j \binom{i}{k} \binom{j}{l} x^{i-k} y^{j-l} (-\bar{x})^k (-\bar{y})^l. \end{aligned} \quad (\text{B.3})$$

592 To relate the coefficients (or parameters) of $\hat{\phi}(x, y)$ with those of $\phi(x, y)$, we use the following
593 transformation:

$$c_{ij} = \sum_{k=i}^n \sum_{l=j}^{n-k} \binom{k}{k-i} \binom{l}{l-j} \frac{\hat{c}_{kl}}{(\sigma_x)^k (\sigma_y)^l} (-\bar{x})^{k-i} (-\bar{y})^{l-j}. \quad (\text{B.4})$$

594 In a *weighted* least-squares formulation, we also need to rewrite the weighting function in
595 terms of (\hat{x}, \hat{y}) . The inverse-distance weighting function proposed in Section 4.3.1 now becomes

$$\hat{w}_i = \frac{1}{\hat{d}_i^n}, \quad (\text{B.5})$$

596 where the standardised distance between the centroid of face f and the reference location of the
 597 i th element of the discretisation stencil is given by

$$\begin{aligned}\hat{d}_i &= \sqrt{\left(\frac{x_i - \mu}{\sigma_x} - \frac{x_f - \mu}{\sigma_x}\right)^2 + \left(\frac{y_i - \mu}{\sigma_y} - \frac{y_f - \mu}{\sigma_y}\right)^2} \\ &\equiv \sqrt{\left(\frac{x_i - x_f}{\sigma_x}\right)^2 + \left(\frac{y_i - y_f}{\sigma_y}\right)^2}.\end{aligned}\quad (\text{B.6})$$

598 As shown in Figs. B.27 and B.28, (the ℓ_2 -norm of) the condition number of the Vandermonde
 599 \mathbf{D}_f and weighted Vandermonde $\sqrt{\mathbf{W}_f} \mathbf{D}_f$ matrices defined in Section 4.3 no longer depends on
 600 the grid spacing, only on the degree of the polynomial fit and on the exponent to which the
 601 standardised distance is elevated.

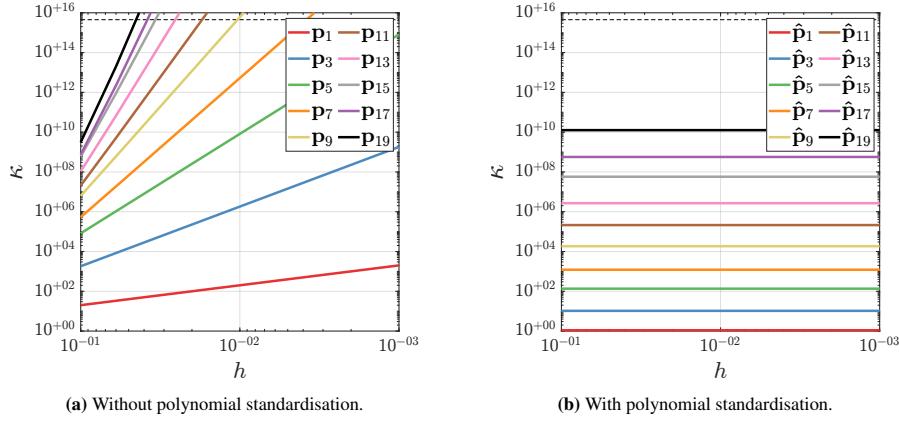


Figure B.27. (Matrix) \mathbf{D}_f conditioning for polynomial fits of different degree.

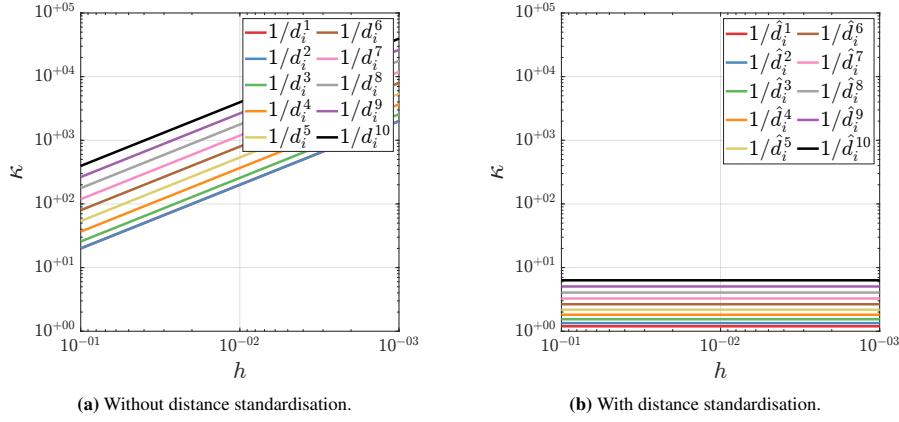


Figure B.28. (Matrix) $\sqrt{\mathbf{W}_f} \mathbf{D}_f$ conditioning for different inverse-distance weighting functions (linear polynomial fit).

602 In the context of the problem at hand, the standardisation procedure introduced here is espe-
 603 cially useful when dealing with extremely dense grids and/or polynomial fits of very high order.

604 **References**

- 605 [1] W. L. Oberkampf, T. G. Trucano, Verification and validation benchmarks, Nuclear Engineering and Design 238
606 (2008) 716–743. Benchmarking of CFD Codes for Application to Nuclear Reactor Safety.
- 607 [2] J. Abanto, D. Pelletier, A. Garon, J.-Y. Trepanier, M. Reggio, Verification of some Commercial CFD Codes
608 on Atypical CFD Problems, in: 43rd AIAA Aerospace Sciences Meeting and Exhibit, American Institute of
609 Aeronautics and Astronautics, Reno, Nevada, 2005.
- 610 [3] D. F. Hawken, J. J. Gottlieb, J. S. Hansen, Review of some adaptive node-movement techniques in finite-element
611 and finite-difference solutions of partial differential equations, Journal of Computational Physics 95 (1991) 254–
612 302.
- 613 [4] Y. Li, S. Premasuthan, A. Jameson, Comparison of Adaptive h and p Refinements for Spectral Difference Methods,
614 in: 40th Fluid Dynamics Conference and Exhibit, American Institute of Aeronautics and Astronautics, Chicago,
615 Illinois, 2010.
- 616 [5] J. P. P. Magalhães, D. M. Albuquerque, J. M. Pereira, J. C. Pereira, Adaptive mesh finite-volume calculation of 2d
617 lid-cavity corner vortices, Journal of Computational Physics 243 (2013) 365–381.
- 618 [6] D. M. S. Albuquerque, J. M. C. Pereira, J. C. F. Pereira, Residual Least-Squares Error Estimate for Unstructured
619 h-Adaptive Meshes, Numerical Heat Transfer, Part B: Fundamentals 67 (2015) 187–210.
- 620 [7] D. McRae, r-refinement grid adaptation algorithms and issues, Computer Methods in Applied Mechanics and
621 Engineering 189 (2000) 1161–1182. Adaptive Methods for Compressible CFD.
- 622 [8] C. Burg, Analytic study of 2D and 3D grid motion using modified Laplacian, International Journal for Numerical
623 Methods in Fluids 52 (2006) 163–197.
- 624 [9] L. D. Lorenzis, A. Düster, Modeling in Engineering Using Innovative Numerical Methods for Solids and Fluids,
625 CISM International Centre for Mechanical Sciences, 1st ed., Springer, 2021.
- 626 [10] A. Patra, J. T. Oden, Computational techniques for adaptive hp finite element methods, Finite Elements in Analysis
627 and Design 25 (1997) 27–39. Adaptive Meshing, Part 1.
- 628 [11] B. R. Ahrabi, W. K. Anderson, J. C. Newman, An adjoint-based hp-adaptive stabilized finite-element method
629 with shock capturing for turbulent flows, Computer Methods in Applied Mechanics and Engineering 318 (2017)
630 1030–1065.
- 631 [12] Z. J. Wang, K. Fidkowski, R. Abgrall, F. Bassi, D. Caraeni, A. Cary, H. Deconinck, R. Hartmann, K. Hillewaert,
632 H. T. Huynh, N. Kroll, G. May, P.-O. Persson, B. van Leer, M. Visbal, High-order CFD methods: current status
633 and perspective, International Journal for Numerical Methods in Fluids 72 (2013) 811–845.
- 634 [13] Z. J. Wang, High-order computational fluid dynamics tools for aircraft design, Philosophical Transactions of the
635 Royal Society A: Mathematical, Physical and Engineering Sciences 372 (2014) 20130318.
- 636 [14] M. Kronbichler, P.-O. Persson, Efficient High-Order Discretizations for Computational Fluid Dynamics, volume
637 602 of *CISM International Centre for Mechanical Sciences*, 1st ed., Springer, 2021.
- 638 [15] J. Loffeld, J. A. F. Hittinger, On the arithmetic intensity of high-order finite-volume discretizations for hyperbolic
639 systems of conservation laws, The International Journal of High Performance Computing Applications 33 (2019)
640 25–52.
- 641 [16] R. E. Bank, A. H. Sherman, A. Weiser, Some Refinement Algorithms And Data Structures For Regular Local Mesh
642 Refinement, Scientific Computing 1 (1983) 3–17.
- 643 [17] J. A. White, H. Nishikawa, R. A. Baurle, Weighted Least-squares Cell-Average Gradient Construction Methods For
644 The VULCAN-CFD Second-Order Accurate Unstructured Grid Cell-Centered Finite-Volume Solver, in: AIAA
645 Scitech 2019 Forum, American Institute of Aeronautics and Astronautics, San Diego, California, 2019.
- 646 [18] P. Tsoutsanis, V. A. Titarev, D. Drikakis, WENO schemes on arbitrary mixed-element unstructured meshes in three
647 space dimensions, Journal of Computational Physics 230 (2011) 1585–1601.
- 648 [19] M. Dumbser, M. Käser, V. A. Titarev, E. F. Toro, Quadrature-free non-oscillatory finite volume schemes on un-
649 structured meshes for nonlinear hyperbolic systems, Journal of Computational Physics 226 (2007) 204–243.
- 650 [20] S. Diot, S. Clain, R. Loubère, Improved detection criteria for the Multi-dimensional Optimal Order Detection
651 (MOOD) on unstructured meshes with very high-order polynomials, Computers & Fluids 64 (2012) 43–63.
- 652 [21] S. Clain, S. Diot, R. Loubère, A high-order finite volume method for systems of conservation laws—Multi-
653 dimensional Optimal Order Detection (MOOD), Journal of Computational Physics 230 (2011) 4028–4050.
- 654 [22] M. Dumbser, M. Castro, C. Parés, E. F. Toro, ADER schemes on unstructured meshes for nonconservative hyper-
655 bolic systems: Applications to geophysical flows, Computers & Fluids 38 (2009) 1731–1748.
- 656 [23] A. Jalali, C. F. O. Gooch, Higher-Order Finite Volume Solution Reconstruction on Highly Anisotropic Meshes, in:
657 21st AIAA Computational Fluid Dynamics Conference, American Institute of Aeronautics and Astronautics, San
658 Diego, California, 2013.
- 659 [24] X. Nogueira, L. Cueto-Felgueroso, I. Colominas, F. Navarrina, M. Casteleiro, A new shock-capturing technique
660 based on Moving Least Squares for higher-order numerical schemes on unstructured grids, Computer Methods in
661 Applied Mechanics and Engineering 199 (2010) 2544–2558.

- 662 [25] M. C. Bessemoulin-Chatard, Development and analysis of finite volume schemes motivated by the preservation of
 663 asymptotic behaviors. Application to models from physics and biology., PhD dissertation, Université Blaise Pascal
 664 - Clermont-Ferrand II, 2012.
- 665 [26] K. J. Fidkowski, Output-based error estimation and mesh adaptation for unsteady turbulent flow simulations,
 666 Computer Methods in Applied Mechanics and Engineering 399 (2022) 115322.
- 667 [27] C. Roy, Strategies for Driving Mesh Adaptation in CFD (Invited), in: 47th AIAA Aerospace Sciences Meeting in-
 668 cluding The New Horizons Forum and Aerospace Exposition, American Institute of Aeronautics and Astronautics,
 669 Orlando, Florida, 2009.
- 670 [28] A. Hay, M. Visonneau, Error estimation using the error transport equation for finite-volume methods and arbitrary
 671 meshes, International Journal of Computational Fluid Dynamics 20 (2006) 463–479.
- 672 [29] A. Hay, A. Leroyer, M. Visonneau, H-adaptive Navier-Stokes simulations of free-surface flows around moving
 673 bodies, Journal of Marine Science and Technology 11 (2006) 1–18.
- 674 [30] V. Ervin, W. Layton, An Analysis of a Defect-Correction Method for a Model Convection-Diffusion Equation,
 675 SIAM Journal on Numerical Analysis 26 (1989) 169–179.
- 676 [31] N. A. Pierce, M. B. Giles, Adjoint and defect error bounding and correction for functional estimates, Journal of
 677 Computational Physics 200 (2004) 769–794.
- 678 [32] J. H. Ferziger, M. Perić, Computational Methods for Fluid Dynamics, 3rd ed., Springer, 2002.
- 679 [33] A. Syrakos, G. Efthimiou, J. G. Bartzis, A. Goulas, Numerical experiments on the efficiency of local grid refinement
 680 based on truncation error estimates, Journal of Computational Physics 231 (2012) 6725–6753.
- 681 [34] M. Aftosmis, M. Berger, Multilevel error estimation and adaptive h-refinement for Cartesian meshes with embed-
 682 ded boundaries, in: 40th AIAA Aerospace Sciences Meeting & Exhibit, American Institute of Aeronautics and
 683 Astronautics, Reno, Nevada, 2002.
- 684 [35] W. Bangerth, R. Rannacher, Adaptive Finite Element Methods for Differential Equations, 1st ed., Birkhäuser, 2003.
- 685 [36] Z. Wang, A perspective on high-order methods in computational fluid dynamics, Science China Physics, Mechanics
 686 & Astronomy 59 (2015) 614701.
- 687 [37] S. Muzaferija, D. Gosman, Finite-Volume CFD Procedure and Adaptive Error Control Strategy for Grids of Arbi-
 688 trary Topology, Journal of Computational Physics 138 (1997) 766–787.
- 689 [38] T. J. Barth, D. C. Jespersen, The design and application of upwind schemes on unstructured meshes, in: 27th
 690 AIAA Aerospace Sciences Meeting, American Institute of Aeronautics and Astronautics, Moffett Field, California,
 691 1989.
- 692 [39] C. Ollivier-Gooch, M. V. Altena, A High-Order-Accurate Unstructured Mesh Finite-Volume Scheme for the Ad-
 693 vection–Diffusion Equation, Journal of Computational Physics 181 (2002) 729–752.
- 694 [40] H. T. Nguyen, p-Adaptive and Automatic h-Adaptive Finite Element Methods for Elliptic Partial Differential Equa-
 695 tions, PhD dissertation, University of California, San Diego, 2010.
- 696 [41] P. M. P. Costa, D. M. S. Albuquerque, A novel approach for temporal simulations with very high-order finite
 697 volume schemes on polyhedral unstructured grids, Journal of Computational Physics 453 (2022) 110960.