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

Filipe F. Tenreiro[a], Duarte M. S. Albuquerque[a,b,*]

[a] *IDMEC, Instituto Superior Técnico, Technical University of Lisbon, Av. Rovisco Pais 1, 1049-001, Lisbon, Portugal*
[b] *UNIDEMI, 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)

1. Literature review

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

1.1. Adaptive mesh refinement

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

1.1.1. Adaptation drivers

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

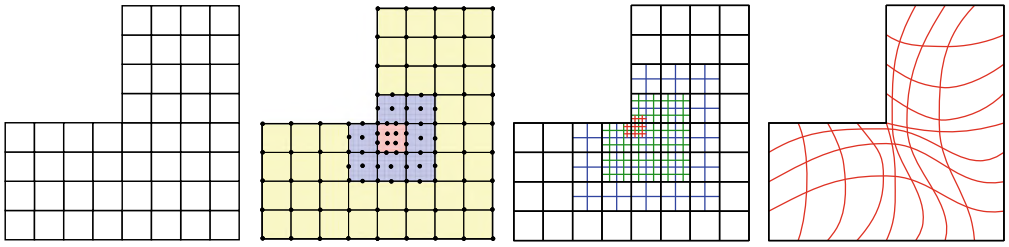
1.1.2. Adaptation methods

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

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

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

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



(a) Original mesh. **(b)** p-adapted mesh (FEM). **(c)** h-adapted mesh. **(d)** r-adapted mesh.

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

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

1.2. Higher-order methods

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

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

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

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

2. Contributions

The main contributions made during this paper are as follows:

- Proposal of a numerical reconstruction technique whose point-value interpolation is done about (the centroid of) the face.
- Derivation of two distinct *a posteriori* error estimators.
  - The consistency (i.e., asymptotic exactness) of these estimates is verified by evaluating the rate of reduction of the truncation error (or discrete residual) on successively finer grids.
  - To relate the discretisation and truncation errors, we use the discrete form of the (discretisation) error transport equation (28).
- Proposal of a novel order-based adaptation (or p-adaptation) method.
  - Since the proposed (adaptive) process does not involve changing the geometry of the grid, the error estimators derived in Sections 5.3.1 and 5.3.2 use a higher-order flux reconstruction technique.
  - The proposed refinement rules were inspired by those of the same names in Red-Green (h-) refinement [16].

3. The governing equation

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

The standard form of the transport equation reads

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

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

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

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

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

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

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

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

4. Finite volume (FV) discretisation

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

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

4.1. Flux integration over element faces

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

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

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

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

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

4.2. Source term volume integration

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

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

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

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

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

4.3. Numerical solution reconstruction

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 is done about the centroid of the face, there is no need to interpolate the cell/vertex-centred reconstruction to obtain the surface fluxes.

The local approximation of the underlying solution about the centroid of f can be modelled as a nth-degree polynomial, yielding the general polynomial regression model

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

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

The polynomial regression model

$$\phi_f(x_i, y_i) = c_0 + c_1\left(x_i - x_f\right) + c_2\left(y_i - y_f\right) + \ldots + r_{f_i}, \text{ with } i = 1, 2, \ldots \tag{10}$$

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 vector $\mathbf{c}_f \in \mathbb{R}^n$, a (fitted) response vector $\boldsymbol{\phi}_s \in \mathbb{R}^m$, and a residual vector $\mathbf{r}_f \in \mathbb{R}^m$ as

$$\underbrace{\begin{bmatrix} 1 & \left(x_1 - x_f\right) & \left(y_1 - y_f\right) & \ldots \\ \vdots & \vdots & \vdots & \ddots \\ \hline 1 & \left(x_b - x_f\right) & \left(y_b - y_f\right) & \ldots \\ \hline 0 & n_{b_x} & n_{b_y} & \ldots \\ \hline n_{b_x} + n_{b_y} & \left[\left(x_b - x_f\right) + d_{b_x}\right] \cdot n_{b_x} & \left[\left(y_b - y_f\right) + d_{b_y}\right] \cdot n_{b_y} & \ldots \end{bmatrix}}_{\mathbf{D}_f} \underbrace{\begin{bmatrix} c_1 \\ \vdots \\ \hline c_b \\ \hline c_b \\ \hline c_b \end{bmatrix}}_{\mathbf{c}_f} = \underbrace{\begin{bmatrix} \phi_1 \\ \vdots \\ \hline \phi_b \\ \hline \nabla\phi_b \cdot \mathbf{n}_b \\ \hline \left[\phi_b + \mathbf{d}_b \nabla\phi_b\right] \cdot \mathbf{n}_b \end{bmatrix}}_{\boldsymbol{\phi}_s} - \underbrace{\begin{bmatrix} r_1 \\ \vdots \\ \hline r_b \\ \hline r_b \\ \hline r_b \end{bmatrix}}_{\mathbf{r}_f}, \tag{11}$$

which when using pure matrix notation can be written as

$$\mathbf{D}_f \mathbf{c}_f = \hat{\boldsymbol{\phi}}_s, \tag{12}$$

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

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

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

4.3.1. *Unconstrained polynomial reconstruction*

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

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

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

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

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

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

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

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

where

$$\mathbf{d}_f^C(\mathbf{x}) = \begin{bmatrix} 1 & \left(x - x_f\right) & \left(y - y_f\right) & \left(x - x_f\right)^2 & \left(x - x_f\right)\left(y - y_f\right) & \left(y - y_f\right)^2 & \dots \end{bmatrix}^\mathsf{T} \tag{17a}$$

$$\mathbf{d}_f^{D_x}(\mathbf{x}) = \begin{bmatrix} 0 & 1 & 0 & 2\left(x - x_f\right) & \left(y - y_f\right) & 0 & \dots \end{bmatrix}^\mathsf{T} \tag{17b}$$

$$\mathbf{d}_f^{D_y}(\mathbf{x}) = \begin{bmatrix} 0 & 0 & 1 & 0 & \left(x - x_f\right) & 2\left(y - y_f\right) & \dots \end{bmatrix}^\mathsf{T} \tag{17c}$$

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

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

$$w_i = \frac{1}{d_i^n}, \tag{15}$$

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 ith element of the discretisation stencil. Here, we simply set n to 2.

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

4.3.2. *Constrained polynomial reconstruction*

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

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

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

which when using pure matrix notation can be written as

$$\mathbf{H}_f \mathbf{c}_f = \boldsymbol{\phi}_g, \tag{19}$$

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

$$\mathbf{d}_g \equiv \mathbf{d}\left(\mathbf{x}_g\right) = \frac{\boldsymbol{\Gamma}\left(\mathbf{x}_g\right)}{\mathbf{u}\left(\mathbf{x}_g\right)}. \tag{20}$$

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

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

$$\left[\begin{array}{cc} 2\mathbf{D}_f^\mathsf{T}\mathbf{W}_f\mathbf{D}_f & \mathbf{H}_f^\mathsf{T} \\ \hline \mathbf{H}_f & \mathbf{0} \end{array}\right] \left[\begin{array}{c} \mathbf{c}_f \\ \hline \boldsymbol{\mu} \end{array}\right] = \left[\begin{array}{c} 2\mathbf{D}_f \sqrt{\mathbf{W}}_f \boldsymbol{\phi}_s \\ \hline \boldsymbol{\phi}_g \end{array}\right], \tag{21}$$

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

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

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

$$\begin{aligned}\int_f \mathbf{J}^{\phi,C} \cdot \, d\mathbf{S} &\approx \left[\begin{bmatrix} u\left(\mathbf{x}_{g_1}\right)\omega_{g_1} & u\left(\mathbf{x}_{g_2}\right)\omega_{g_2} & \dots \\ v\left(\mathbf{x}_{g_1}\right)\omega_{g_1} & v\left(\mathbf{x}_{g_2}\right)\omega_{g_2} & \dots \end{bmatrix} \begin{bmatrix} \mathbf{d}_f^C\left(\mathbf{x}_g\right) \\ \mathbf{d}_f^C\left(\mathbf{x}_g\right) \end{bmatrix} \right] \left[\; \mathbf{P}_f \boldsymbol{\phi}_s + \mathbf{k}_f \; \right] \cdot \mathbf{S}_f \\ &= \left[\mathbf{t}_f^C \boldsymbol{\phi}_s + \mathbf{w}_f^C\right] \cdot \mathbf{S}_f \end{aligned} \tag{22a}$$

$$\begin{aligned}\int_f \mathbf{J}^{\phi,D} \cdot \, d\mathbf{S} &\approx \left[\begin{bmatrix} \Gamma_x\left(\mathbf{x}_{g_1}\right)\omega_{g_1} & \Gamma_x\left(\mathbf{x}_{g_2}\right)\omega_{g_2} & \dots \\ \Gamma_y\left(\mathbf{x}_{g_1}\right)\omega_{g_1} & \Gamma_y\left(\mathbf{x}_{g_2}\right)\omega_{g_2} & \dots \end{bmatrix} \begin{bmatrix} \mathbf{d}_f^{D_x}\left(\mathbf{x}_g\right) \\ \mathbf{d}_f^{D_y}\left(\mathbf{x}_g\right) \end{bmatrix} \right] \left[\; \mathbf{P}_f \boldsymbol{\phi}_s + \mathbf{k}_f \; \right] \cdot \mathbf{S}_f \\ &= \left[\mathbf{t}_f^D \boldsymbol{\phi}_s + \mathbf{w}_f^D\right] \cdot \mathbf{S}_f, \end{aligned} \tag{22b}$$

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 cost[3] 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 $\left[\dfrac{\mathbf{D}_f}{\mathbf{H}_f}\right]$ (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

$$\mathbf{A}\boldsymbol{\phi} = \mathbf{b}, \tag{23}$$

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.

[3] Here, 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).

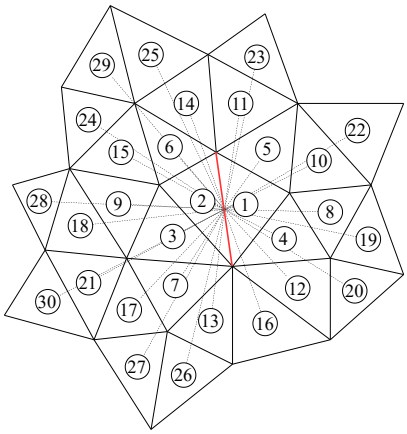


Figure 2. Central-biased stencil about the face marked in red (elements are numbered in the order that they are added).

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:

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

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

5. Numerical verification

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

5.1. Exact solutions

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

5.2. Error sources

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

$$\mathscr{L}(\Phi) = 0 \tag{25a}$$

and the difference approximation by

$$\mathscr{L}_{(p)}(\phi) = 0, \tag{25b}$$

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

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

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

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.,

$$e^{\Phi} \equiv e(\Phi) = \phi - \Phi. \tag{27}$$

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

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

Otherwise, for small enough e^{Φ}, it becomes

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

where $\mathscr{L}'_{(p)}$ is the Jacobian matrix of $\mathscr{L}_{(p)}$. 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.

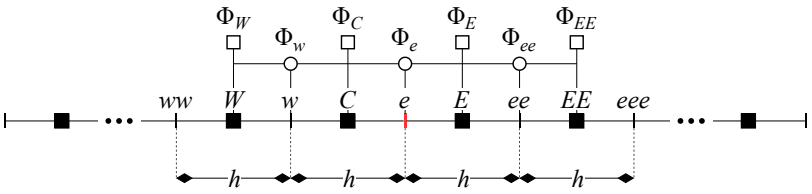


Figure 3. Uniformly spaced 1D grid (spacing h).

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

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

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

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

$$\begin{aligned}
\bar{\tau}_e^{\Phi}(x_e) &= \bar{\tau}_e^{\Phi,C}(x_e) - \bar{\tau}_e^{\Phi,D}(x_e) \\
&= \underbrace{u_e\Bigg(\underbrace{\Phi_e + \frac{h^2}{8}\Phi_e'' + \ldots}_{\Phi_e^{(p)}(x_e)}\Bigg) - \Gamma_e\Bigg(\underbrace{\Phi_e' + \frac{h^2}{24}\Phi_e'''' + \ldots}_{\Phi_e'^{(p)}(x_e)}\Bigg)}_{\mathbf{t}_e\boldsymbol{\Phi}_s} - \underbrace{\Bigg[u_e\Bigg(\underbrace{\frac{\phi_C+\phi_E}{2}}_{\phi_e^{(p)}(x_e)}\Bigg) - \Gamma_e\Bigg(\underbrace{\frac{\phi_E-\phi_C}{h}}_{\phi_e'^{(p)}(x_e)}\Bigg)\Bigg]}_{\mathbf{t}_e\boldsymbol{\phi}_s} \\
&= \underbrace{u_e\left[\Phi_e - \left(\frac{\phi_C+\phi_E}{2}\right)\right] - \Gamma_e\left[\Phi_e' - \left(\frac{\phi_E-\phi_C}{h}\right)\right]}_{\bar{\tau}_0^{\Phi}\big|_e(x_e) = a_0^{\Phi}\big|_e(x_e)} + \underbrace{u_e\left(\frac{h^2}{8}\Phi_e'' + \ldots\right) - \Gamma_e\left(\frac{h^2}{24}\Phi_e'''' + \ldots\right)}_{\bar{\tau}_1^{\Phi}\big|_e(x_e) + \ldots = a_1^{\Phi}\big|_e(x_e)\,h^2 + \ldots},
\end{aligned} \tag{30}$$

where

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

and

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

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

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

$$\begin{aligned}
\underbrace{\bar{\tau}_e^{\Phi}(x_e)}_{\text{Tau}} &= \underbrace{\bar{\tau}_0^{\Phi}\big|_e(x_e)}_{\text{Tau-0}} + \underbrace{\bar{\tau}_{\Sigma k}^{\Phi}\big|_e(x_e)}_{\text{Tau-k}} \\
&\equiv \bar{\tau}_0^{\Phi}\big|_e(x_e) + \sum_{k=m}^{\infty}\bar{\tau}_k^{\Phi}\big|_e(x_e),
\end{aligned} \tag{33}$$

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

$$\Phi_C = \Phi_e - \frac{h}{2}\Phi_e' + \frac{h^2}{8}\Phi_e'' - \frac{h^3}{48}\Phi_e''' + \frac{h^4}{384}\Phi_e^{iv} - \frac{h^5}{3840}\Phi_e^{v} + \ldots \tag{32a}$$

$$\Phi_E = \Phi_e + \frac{h}{2}\Phi_e' + \frac{h^2}{8}\Phi_e'' + \frac{h^3}{48}\Phi_e''' + \frac{h^4}{384}\Phi_e^{iv} + \frac{h^5}{3840}\Phi_e^{v} + \ldots, \tag{32b}$$

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 $x = x_e$.

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

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

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

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

5.3. Error estimation

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

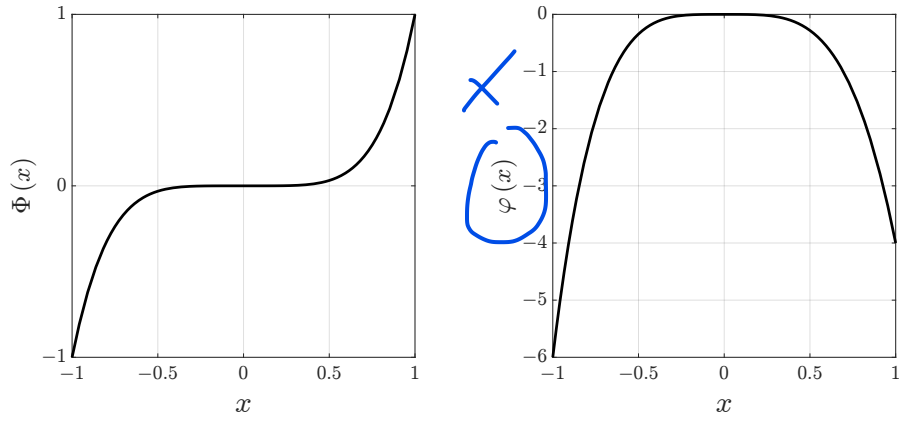
In general, the level of reliability of *a posteriori* methods is strongly problem-dependent. As detailed in Section 4.3.3, all the methods developed here are predominantly suited for elliptic problems, and as a result, they are not as well-developed for mathematical models that are parabolic or hyperbolic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

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

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

[5] $\bar{\tau}_0^{\Phi}\big|_C$ 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 $\bar{\tau}_0^{\Phi}\big|_C = 0$. If a Gaussian quadrature integration was instead used to discretise the RHS of Eq. (4), this would not be the case.

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



(a) Analytical scalar property. **(b)** Volumetric source (or sink) term.

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

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

5.3.1. Higher-order polynomial regression

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

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

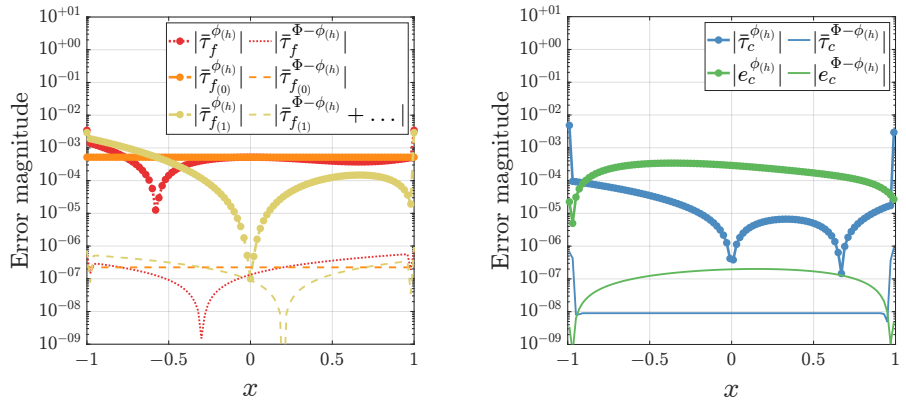
$$\begin{aligned}\bar{\tau}_e^{\phi_{(h)}}(x_e) &= \underbrace{u_e\left(\phi_e + \frac{h^2}{8}\phi_e''\right) - \Gamma_e\left(\phi_e' + \frac{h^2}{24}\phi_e'''\right)}_{\mathbf{t}_{e_{(l)}}\boldsymbol{\phi}_{s_{(h)}}} - \underbrace{\left[u_e\left(\frac{\phi_C+\phi_E}{2}\right) - \Gamma_e\left(\frac{\phi_E-\phi_C}{h}\right)\right]}_{\mathbf{t}_{e_{(l)}}\boldsymbol{\phi}_{s_{(l)}}} \\ &= \underbrace{u_e\left[\phi_e - \left(\frac{\phi_C+\phi_E}{2}\right)\right] - \Gamma_e\left[\phi_e' - \left(\frac{\phi_E-\phi_C}{h}\right)\right]}_{\bar{\tau}_0^{\phi_{(h)}}\big|_e(x_e) = a_0^{\phi_{(h)}}\big|_e(x_e)} + \underbrace{u_e\left(\frac{h^2}{8}\phi_e''\right) - \Gamma_e\left(\frac{h^2}{24}\phi_e'''\right)}_{\bar{\tau}_1^{\phi_{(h)}}\big|_e(x_e) = a_1^{\phi_{(h)}}\big|_e(x_e)\,h^2},\end{aligned} \tag{36}$$

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

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

As shown in Fig. 6, we can see that (in the asymptotic range) this difference is indeed negligible, i.e.,

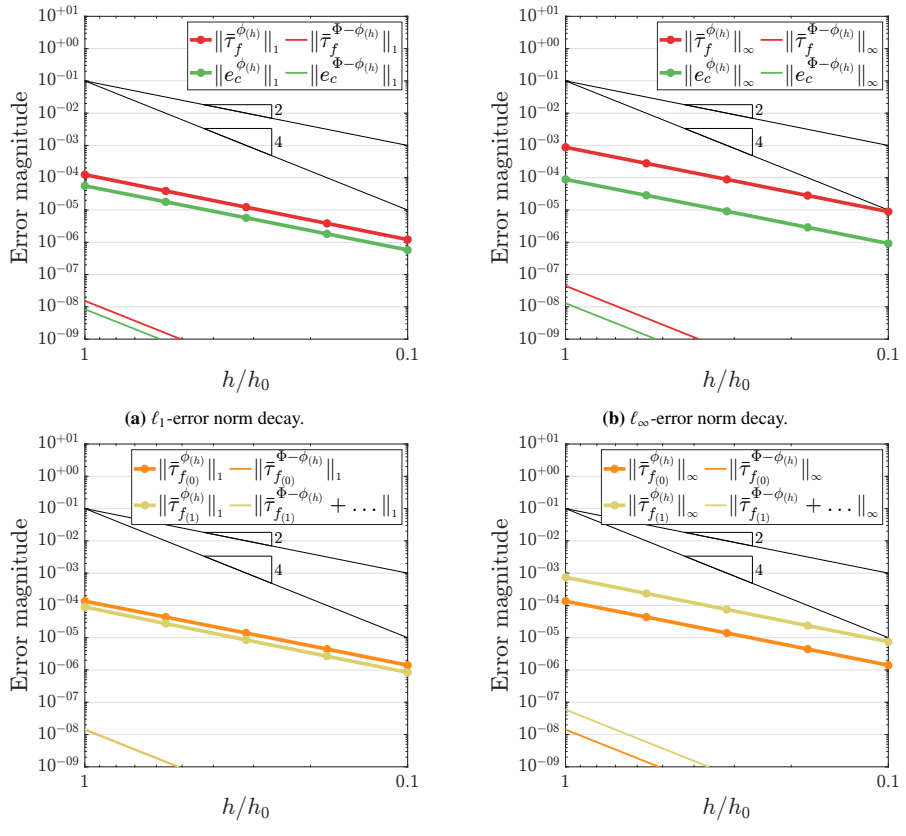
$$\mathscr{L}_{(2)}(\Phi - \phi_{(4)}) \approx O(h^4). \tag{37}$$



(a) (Face) truncation error magnitude.

(b) (Cell) error magnitude.

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



(a) ℓ_1-error norm decay.

(b) ℓ_∞-error norm decay.

(c) ℓ_1-error norm decay.

(d) ℓ_∞-error norm decay.

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

5.3.2. *Tau-extrapolation*

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

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

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

where

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

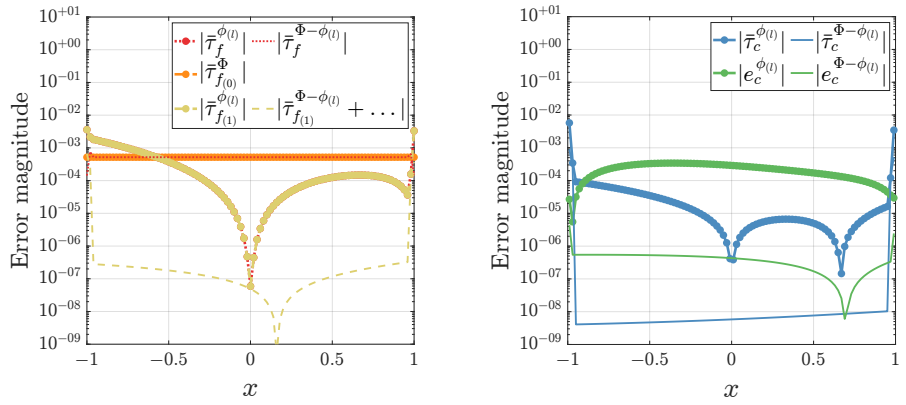
and

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

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

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

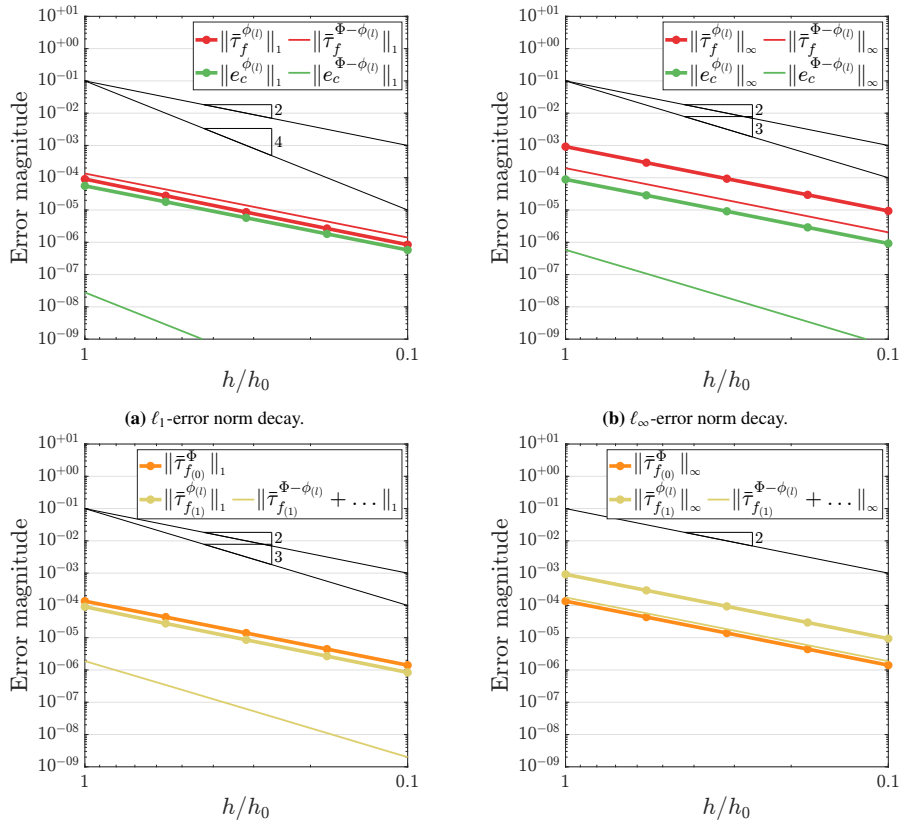
In addition, even though this error estimator is not able to estimate the tau-0 component of the face truncation error, we can always confidently approximate the cell discretisation and truncation errors (note that this component is nil about the cell, see Eq. (34a)). Nevertheless, as we will see in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



(a) (Face) truncation error magnitude.

(b) (Cell) error magnitude.

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



(a) ℓ_1-error norm decay.

(b) ℓ_∞-error norm decay.

(c) ℓ_1-error norm decay.

(d) ℓ_∞-error norm decay.

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

6. Adaptive mesh refinement

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

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

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

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

- Computation is not expensive[6].
- Neighbouring elements are *not* too different in degree.
- The algorithm can be implemented cheaply[7].

While the first and third requirements are natural for any practical algorithm, the main purpose of the second one is to smooth the changes of the approximate solution from element to element.

6.1. p-adaptive meshing

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

Even though the method proposed here is valid for any unstructured grid, extra care must be taken when dealing with (non-isotropic) irregular grids (see Section 4.3.3). For this reason, in the numerical examples presented in Section 6.2 we simply use a Cartesian grid[8].

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

[7] If 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.

[8] The 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.

6.1.1. *Adaptation driver*

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

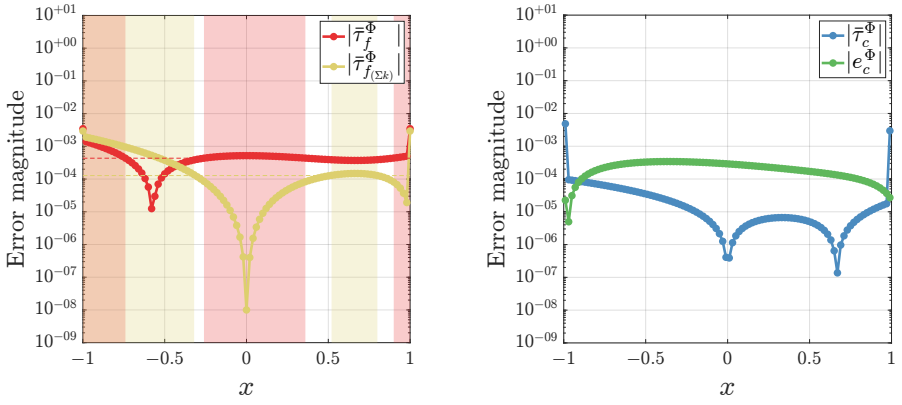
$$\bar{\boldsymbol{\tau}}^{\Phi} = -\mathscr{L}_{(p)}(\mathbf{e}^{\Phi}), \tag{40}$$

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

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

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

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

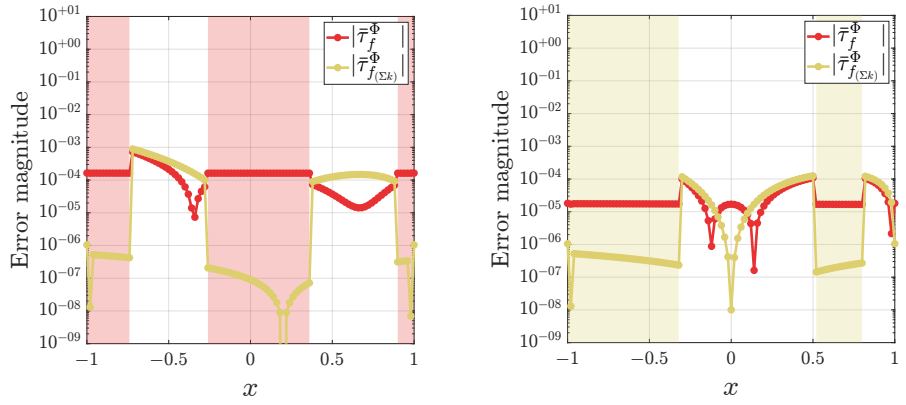


(a) (Face) truncation error magnitude. **(b)** (Cell) error magnitude.

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

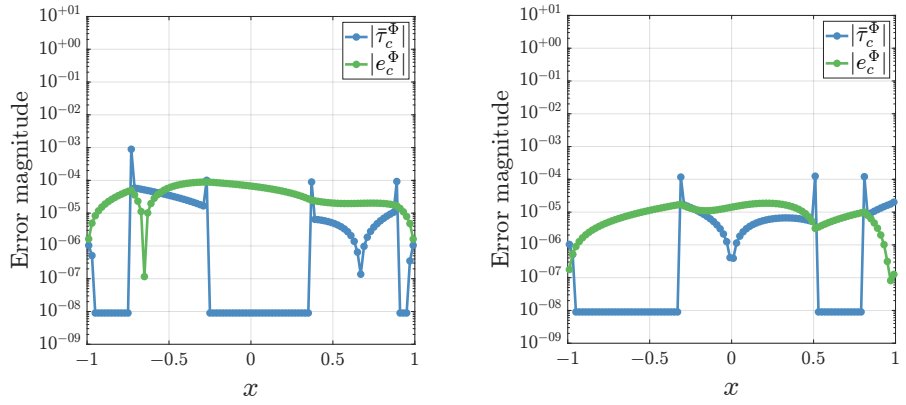
By sorting the magnitude of both criteria (tau and tau-k) in descending order and flagging for refinement the first λ~50% of elements (or faces) from this list, we compare the 1D distribution of the magnitude of the error after refinement with the one prior to it (see error thresholds in Fig. 9a and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

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



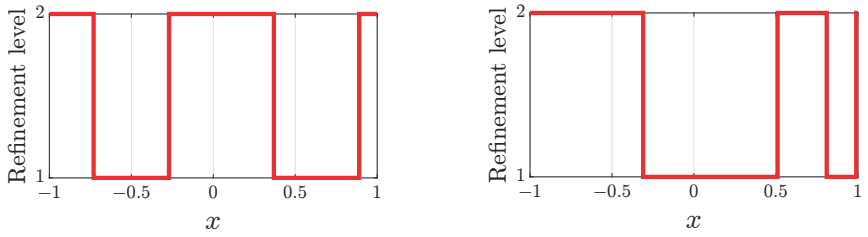
(a) Adaptation criterion: $|\bar{\tau}_f^{\Phi}|$ (tau). **(b)** Adaptation criterion: $|\bar{\tau}_{\Sigma k}^{\Phi}|_f|$ (tau-k).

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



(a) Adaptation criterion: $|\bar{\tau}_f^{\Phi}|$ (tau). **(b)** Adaptation criterion: $|\bar{\tau}_{\Sigma k}^{\Phi}|_f|$ (tau-k).

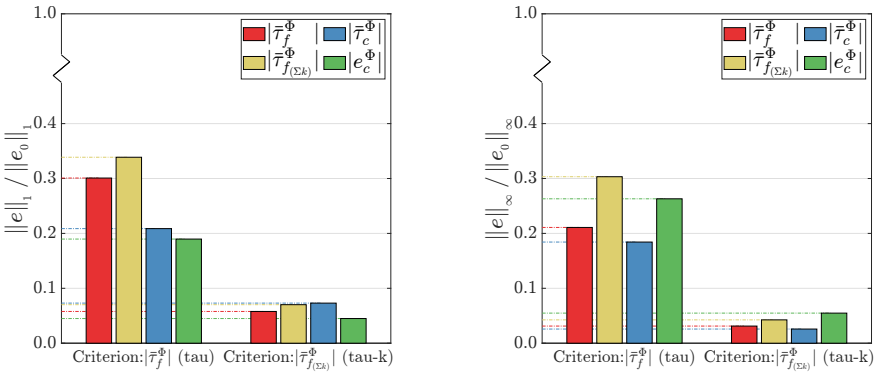
Figure 11. 1D distribution of the magnitude of the (cell) error after refinement.



(a) Adaptation criterion: $|\bar{\tau}_f^{\Phi}|$ (tau). **(b)** Adaptation criterion: $|\bar{\tau}_{\Sigma k}^{\Phi}|_f|$ (tau-k).

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

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



(a) ℓ_1-error norm. **(b)** ℓ_∞-error norm.

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 domain[9]. 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.

[9] The 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:

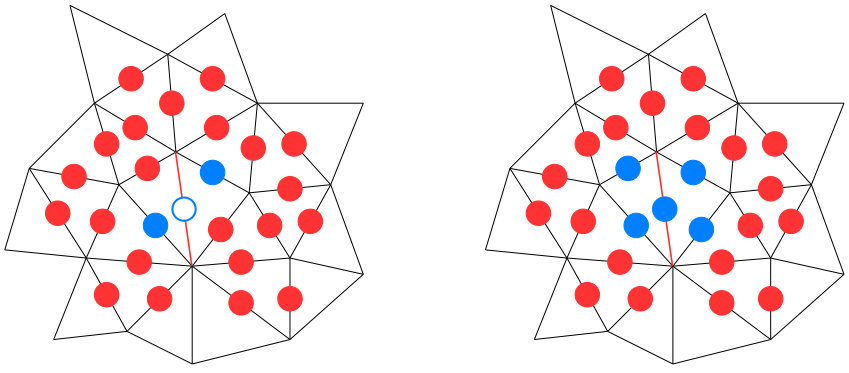
- A (remaining) desired error level for the functional of interest.
- An upper limit on the (total) number of DoFs[10].
- An upper limit on the (total) number of adaptation cycles.

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.

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).



(a) Before refinement (flagged for refinement: ○-p). **(b)** After refinement.

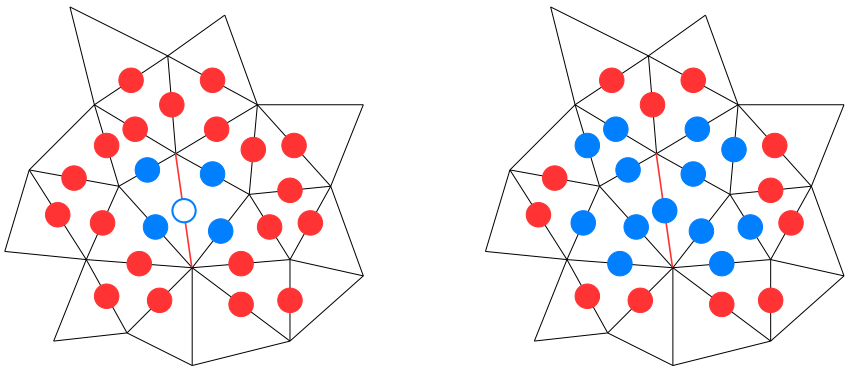
Figure 14. Neighbour rule (refinement level: ●-p_1, ●-p_2, with $p_2 \equiv p = p_1 + 1$).

[10]The (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]:

$$h \sim \text{DoFs}^{-1/d}, \tag{41}$$

where d is the dimension of the problem.

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



(a) Before refinement (flagged for refinement: ○-p). **(b)** After refinement.

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

6.2. Numerical examples

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

- Are the proposed error estimators able to accurately drive the (adaptive) process?
  - If so, how does the distribution of the refinement level using the analytical solution compare to that predicted by the aforementioned error estimators?
  - If so, how do the magnitude of the analytical discretisation and truncation errors compare to that predicted by these estimators?
- Does the (adaptive) process effectively reduce the spatial discretisation error?
  - 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 $\left(\Gamma_x, \Gamma_y\right) = (1,1)$ $[m^2/s]$. Dirichlet conditions were prescribed for all the boundaries, and the (manufactured) solutions to this equation were assumed to be:

$$\Phi_a\left(x,y\right) = \exp\left(-100x^2y^2\right) \tag{42a}$$

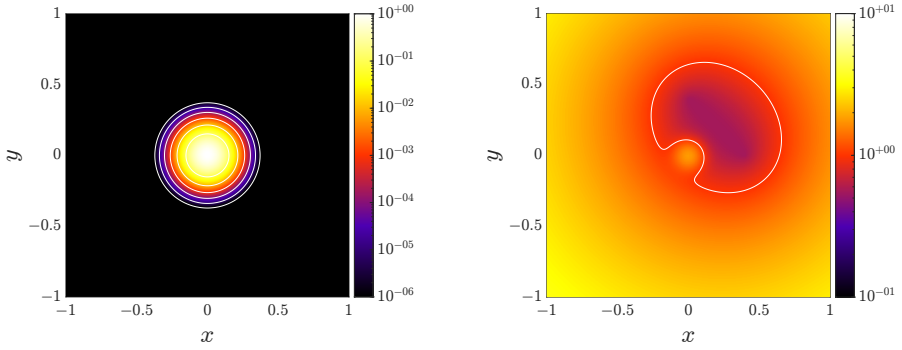
$$\Phi_b\left(x,y\right) = \exp\left(-100x^2y^2\right) + \sqrt{(x-\pi/8)^2+y^2} + \sqrt{x^2+(y-\pi/8)^2}. \tag{42b}$$

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

- Fixed-rate (or fixed-number) parameter: $\lambda = 10\%$.
- Stopping criterion: computational cost ratio increase[11] of (approximately) 3.

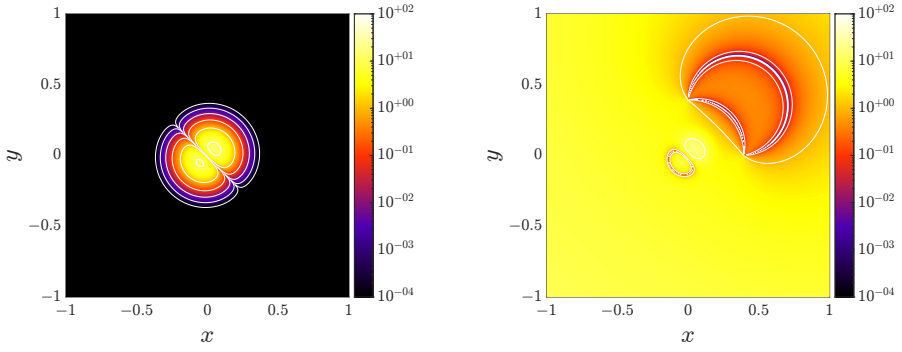
The 2D distribution of the magnitude of the analytical scalar property and respective volumetric source (or sink) term is shown in Figs. 16 and 17, respectively (note that the white contours lines correspond to regions where the respective functions have a constant value equal to the ticks of the colour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

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).



(a) (Manufactured) solution: $\Phi_a\,(x, y)$. **(b)** (Manufactured) solution: $\Phi_b\,(x, y)$.

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



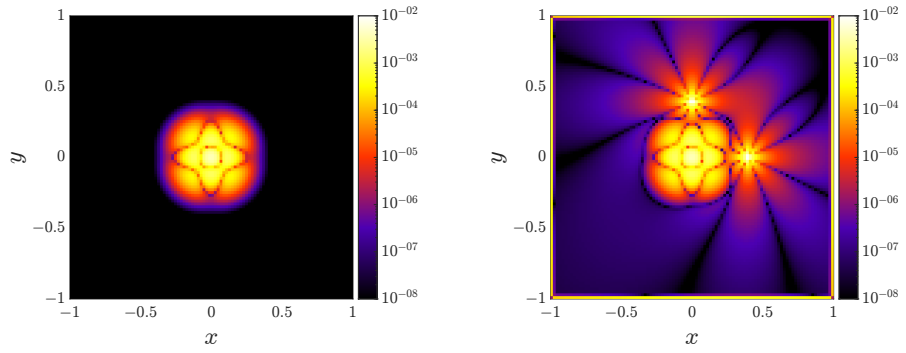
(a) (Manufactured) solution: $\Phi_a\,(x, y)$. **(b)** (Manufactured) solution: $\Phi_b\,(x, y)$.

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

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.

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.

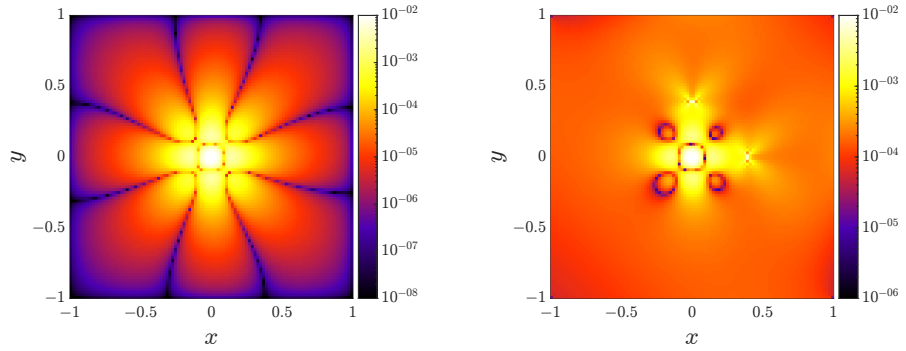
[11]The computational cost increase (or "gain") for the nth adaptation cycle refers to the ratio between the number of DoFs of cycle n and the number of DoFs of the uniform 2nd-order accurate case.



(a) (Manufactured) solution: $\Phi_a\,(x, y)$.

(b) (Manufactured) solution: $\Phi_b\,(x, y)$.

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



(a) (Manufactured) solution: $\Phi_a\,(x, y)$.

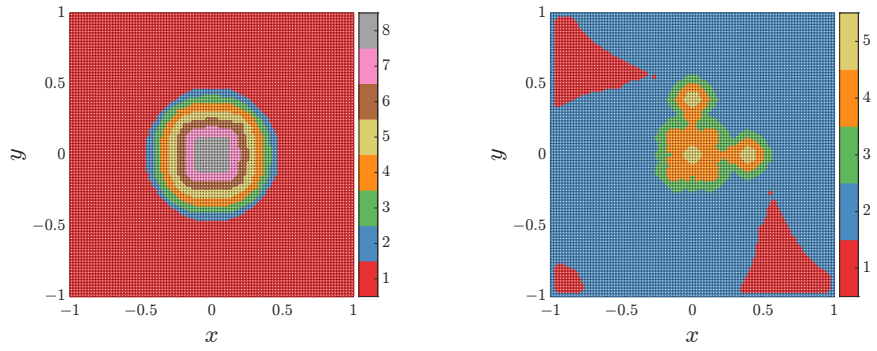
(b) (Manufactured) solution: $\Phi_b\,(x, y)$.

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

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

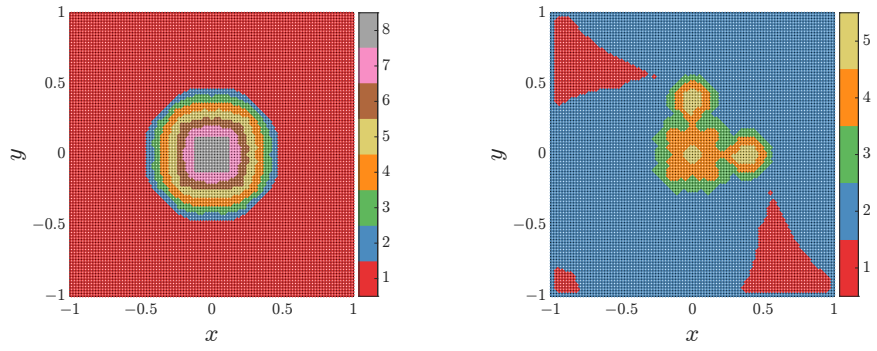
From a distribution point of view, it is clear that both estimators were able to reliably identify where further resolution was needed, even for the case on the right ($\Phi_b\,(x, y)$), where one may have initially thought that the two singularities would have prevented the algorithm from doing so.

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



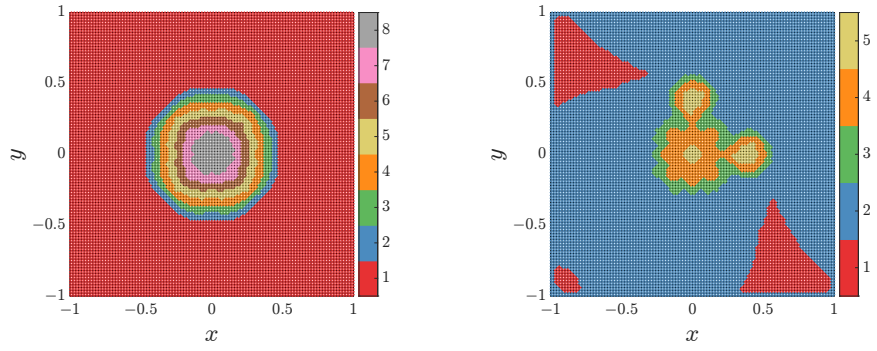
(a) (Manufactured) solution: $\Phi_a\,(x, y)$. **(b)** (Manufactured) solution: $\Phi_b\,(x, y)$.

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



(a) (Manufactured) solution: $\Phi_a\,(x, y)$. **(b)** (Manufactured) solution: $\Phi_b\,(x, y)$.

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

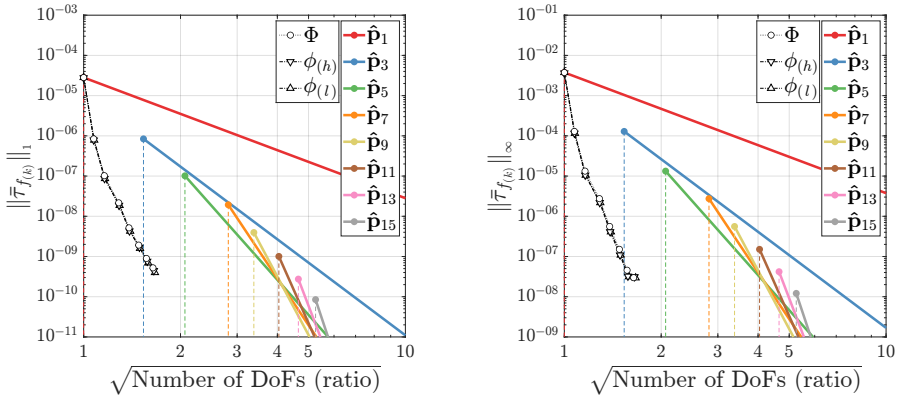


(a) (Manufactured) solution: $\Phi_a\,(x, y)$. **(b)** (Manufactured) solution: $\Phi_b\,(x, y)$.

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

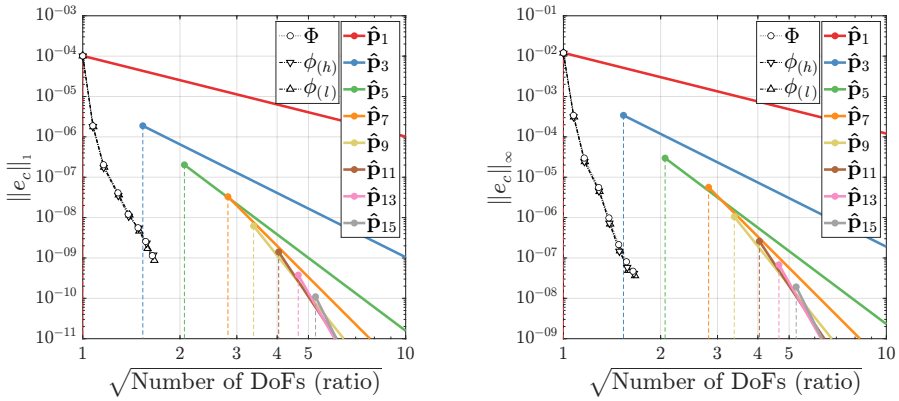
Finally, we present the cycle-by-cycle evolution of both the adaptation criteria and the discretisation error (see Figs. 23 to 26). In these figures, we denote by Φ, $\phi_{(h)}$ and $\phi_{(l)}$ the three cases in Figs. 20 to 22, respectively, and by $\hat{\mathbf{p}}_k$ the analytical uniform k-degree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

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



(a) ℓ_1-error norm decay. **(b)** ℓ_∞-error norm decay.

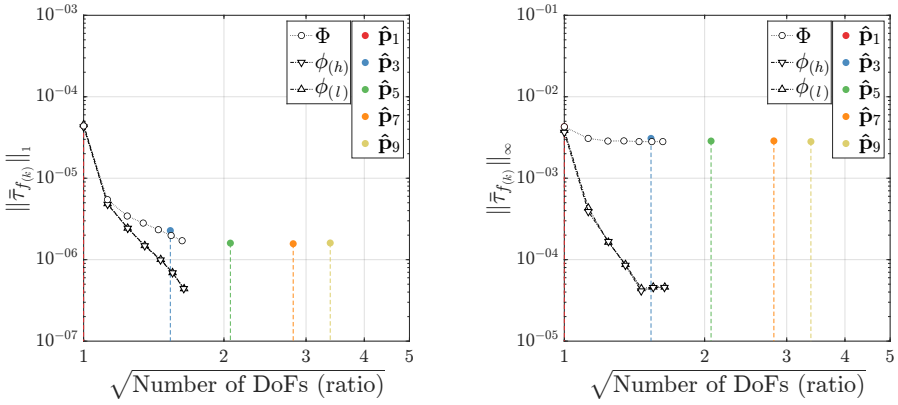
Figure 23. ℓ_1- and ℓ_∞-(face) truncation error (tau-k) norm decay for $\Phi_a\,(x, y)$.



(a) ℓ_1-error norm decay. **(b)** ℓ_∞-error norm decay.

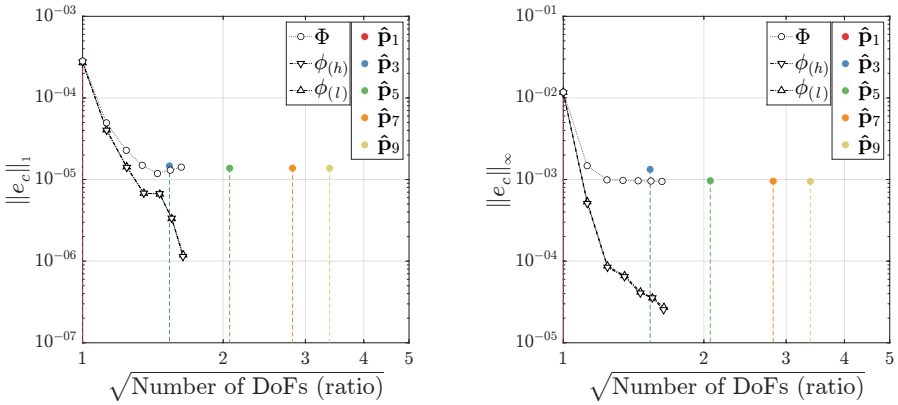
Figure 24. ℓ_1- and ℓ_∞-(cell) discretisation error norm decay for $\Phi_a\,(x, y)$.

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



(a) ℓ_1-error norm decay. **(b)** ℓ_∞-error norm decay.

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



(a) ℓ_1-error norm decay. **(b)** ℓ_∞-error norm decay.

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

7. Conclusions

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

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

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

Declaration of competing interest

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

Data availability

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

Acknowledgments

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

Appendix A. Least-squares fitting

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

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

Consider the following (linear) system of equations

$$\mathbf{A}\mathbf{x} = \mathbf{b}, \tag{A.1}$$

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

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

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

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

$$\begin{aligned} \nabla_{\mathbf{x}} J(\mathbf{x}) &= \nabla_{\mathbf{x}}(\mathbf{b}^\mathsf{T}\mathbf{b} - 2\mathbf{b}^\mathsf{T}\mathbf{A}\mathbf{x} + \mathbf{x}^\mathsf{T}\mathbf{A}^\mathsf{T}\mathbf{A}\mathbf{x}) \\ &= \underbrace{\nabla_{\mathbf{x}}(\mathbf{b}^\mathsf{T}\mathbf{b})}_{\mathbf{0}} - 2\nabla_{\mathbf{x}}(\mathbf{b}^\mathsf{T}\mathbf{A}\mathbf{x}) + \nabla_{\mathbf{x}}(\mathbf{x}^\mathsf{T}\mathbf{A}^\mathsf{T}\mathbf{A}\mathbf{x}) \\ &= -2\mathbf{A}^\mathsf{T}\mathbf{b} + \left[\mathbf{A}^\mathsf{T}\mathbf{A} + (\mathbf{A}^\mathsf{T}\mathbf{A})^\mathsf{T}\right]\mathbf{x} \\ &= -2\mathbf{A}^\mathsf{T}\mathbf{b} + 2\mathbf{A}^\mathsf{T}\mathbf{A}\mathbf{x}, \end{aligned} \tag{A.3}$$

and setting it to zero gives

$$\mathbf{A}^\mathsf{T}\mathbf{A}\mathbf{x}^* = \mathbf{A}^\mathsf{T}\mathbf{b}, \tag{A.4}$$

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

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

$$\mathbf{x}^* = (\mathbf{A}^\mathsf{T}\mathbf{A})^{-1}\mathbf{A}^\mathsf{T}\mathbf{b}. \tag{A.5}$$

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

$$\begin{aligned} \forall \mathbf{x},\ \mathbf{x}^\mathsf{T}(2\mathbf{A}^\mathsf{T}\mathbf{A})\mathbf{x} &= 2(\mathbf{A}\mathbf{x})^\mathsf{T}\mathbf{A}\mathbf{x} \\ &= 2\|\mathbf{A}\mathbf{x}\|_2^2 \geq 0. \end{aligned} \tag{A.6}$$

Appendix A.1.1. Problem conditioning

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

$$\kappa(\mathbf{A}^\mathsf{T}\mathbf{A}) = \kappa(\mathbf{A})^2. \tag{A.7}$$

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

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

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

$$\mathbf{A} = \mathbf{Q}\mathbf{R} = \left[\begin{array}{c|c} \tilde{\mathbf{Q}} & \tilde{\mathbf{Q}}_\perp \end{array}\right] \left[\begin{array}{c} \tilde{\mathbf{R}} \\ \hline \mathbf{0} \end{array}\right] = \tilde{\mathbf{Q}}\tilde{\mathbf{R}}, \tag{A.8}$$

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

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

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

where $\tilde{\mathbf{Q}}^\mathsf{T}\mathbf{b}$ is the minimum residual norm of $\|\mathbf{b} - \mathbf{A}\mathbf{x}\|_2$.

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

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

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

$$\mathbf{C}\mathbf{x} = \mathbf{d}, \tag{A.10}$$

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

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

$$\mathcal{L}(\mathbf{x}, \boldsymbol{\mu}) = \|\mathbf{b} - \mathbf{A}\mathbf{x}\|_2^2 + \boldsymbol{\mu}^\mathsf{T}(\mathbf{C}\mathbf{x} - \mathbf{d}), \tag{A.11}$$

where $\boldsymbol{\mu} \in \mathbb{R}^q$ 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 $\boldsymbol{\mu}^*$ that satisfies

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

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

These are the *optimality conditions* for the CLS problem; any feasible solution must satisfy them.

Combining this set of (linear) equations, we can rewrite the optimality conditions as

$$\left[\begin{array}{cc} 2\mathbf{A}^\mathsf{T}\mathbf{A} & \mathbf{C}^\mathsf{T} \\ \hline \mathbf{C} & \mathbf{0} \end{array}\right] \left[\begin{array}{c} \mathbf{x}^* \\ \hline \boldsymbol{\mu}^* \end{array}\right] = \left[\begin{array}{c} 2\mathbf{A}^\mathsf{T}\mathbf{b} \\ \hline \mathbf{d} \end{array}\right]. \tag{A.13}$$

The $(n+q) \times (n+q)$ matrix on the LHS of Eq. (A.13) is called the KKT matrix. It is invertible if and only if $\mathbf{C}$ has linearly independent rows and the stacked matrix $\left[\begin{array}{c} \mathbf{A} \\ \hline \mathbf{C} \end{array}\right]$ has linearly independent columns. While the first condition requires $\mathbf{C}$ to be wide (or square), i.e., that there are fewer constraints than variables, the second one depends on both $\mathbf{A}$ and $\mathbf{C}$, and it can be satisfied even when the columns of $\mathbf{A}$ are linearly dependent.

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

Assuming that $\mathbf{C}$ has linearly independent rows and the stacked matrix $\left[\begin{array}{c} \mathbf{A} \\ \hline \mathbf{C} \end{array}\right]$ has linearly independent columns, we start by rewriting Eq. (A.13) as

$$\left[\begin{array}{cc} \mathbf{A}^\mathsf{T}\mathbf{A} + \mathbf{C}^\mathsf{T}\mathbf{C} & \mathbf{C}^\mathsf{T} \\ \hline \mathbf{C} & \mathbf{0} \end{array}\right] \left[\begin{array}{c} \mathbf{x}^* \\ \hline \boldsymbol{\nu}^* \end{array}\right] = \left[\begin{array}{c} \mathbf{A}^\mathsf{T}\mathbf{b} \\ \hline \mathbf{d} \end{array}\right], \tag{A.14}$$

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

Next, we use QR factorisation

$$\tilde{\mathbf{Q}}\tilde{\mathbf{R}} = \left[\begin{array}{c} \mathbf{A} \\ \hline \mathbf{C} \end{array}\right] = \left[\begin{array}{c} \tilde{\mathbf{Q}}_1 \\ \hline \tilde{\mathbf{Q}}_2 \end{array}\right] \tilde{\mathbf{R}} \tag{A.15}$$

to simplify Eq. (A.14)

$$\left[\begin{array}{cc} \tilde{\mathbf{R}}^\mathsf{T}\tilde{\mathbf{R}} & \tilde{\mathbf{R}}^\mathsf{T}\tilde{\mathbf{Q}}_2^\mathsf{T} \\ \hline \tilde{\mathbf{Q}}_2\tilde{\mathbf{R}} & \mathbf{0} \end{array}\right] \left[\begin{array}{c} \mathbf{x}^* \\ \hline \boldsymbol{\nu}^* \end{array}\right] = \left[\begin{array}{c} \tilde{\mathbf{R}}^\mathsf{T}\tilde{\mathbf{Q}}_1^\mathsf{T}\mathbf{b} \\ \hline \mathbf{d} \end{array}\right]. \tag{A.16}$$

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

$$\left[\begin{array}{cc} \mathbf{I} & \tilde{\mathbf{Q}}_2^\mathsf{T} \\ \hline \tilde{\mathbf{Q}}_2 & \mathbf{0} \end{array}\right] \left[\begin{array}{c} \mathbf{y}^* \\ \hline \boldsymbol{\nu}^* \end{array}\right] = \left[\begin{array}{c} \tilde{\mathbf{Q}}_1^\mathsf{T}\mathbf{b} \\ \hline \mathbf{d} \end{array}\right]. \tag{A.17}$$

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

$$\tilde{\mathbf{Q}}_2\tilde{\mathbf{Q}}_2^\mathsf{T}\boldsymbol{\nu}^* = \tilde{\mathbf{Q}}_2\tilde{\mathbf{Q}}_1^\mathsf{T}\mathbf{b} - \mathbf{d}. \tag{A.18}$$

Next, we use QR factorisation

$$\tilde{\mathbf{Q}}_2^\mathsf{T} = \tilde{\mathbf{T}}\tilde{\mathbf{U}} \tag{A.19}$$

to simplify Eq. (A.18) and solve for $\boldsymbol{\nu}^*$ as

$$\boldsymbol{\nu}^* = \tilde{\mathbf{U}}^{-1}\left[\tilde{\mathbf{T}}^{\mathsf{T}}\tilde{\mathbf{Q}}_1^{\mathsf{T}}\mathbf{b} - \tilde{\mathbf{U}}^{-\mathsf{T}}\mathbf{d}\right]. \tag{A.20}$$

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

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

Appendix B. Polynomial fit standardisation

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

$$\hat{\mathbf{X}} = \frac{\mathbf{X} - \bar{\mathbf{X}}}{\boldsymbol{\sigma}}, \tag{B.1}$$

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

2D polynomial regression models for curves are given by

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

where n is the degree of the polynomial fit.

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

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

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

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

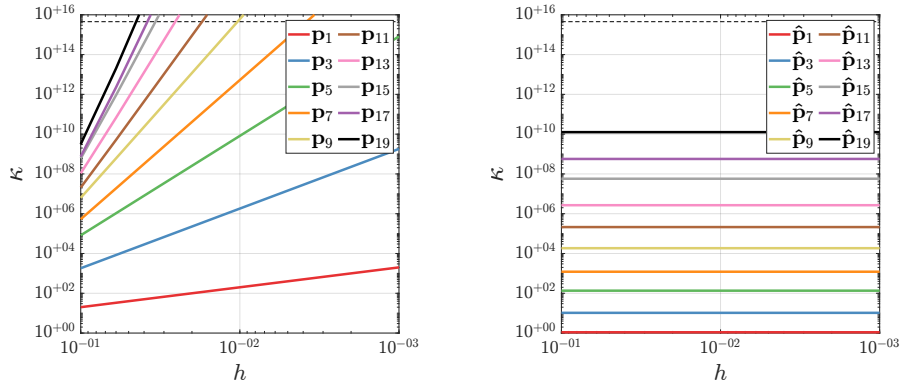
In a *weighted* least-squares formulation, we also need to rewrite the weighting function in terms of $(\hat{x}, \hat{y})$. The inverse-distance weighting function proposed in Section 4.3.1 now becomes

$$\hat{w}_i = \frac{1}{\hat{d}_i^n}, \tag{B.5}$$

where the standardised distance between the centroid of face f and the reference location of the ith element of the discretisation stencil is given by

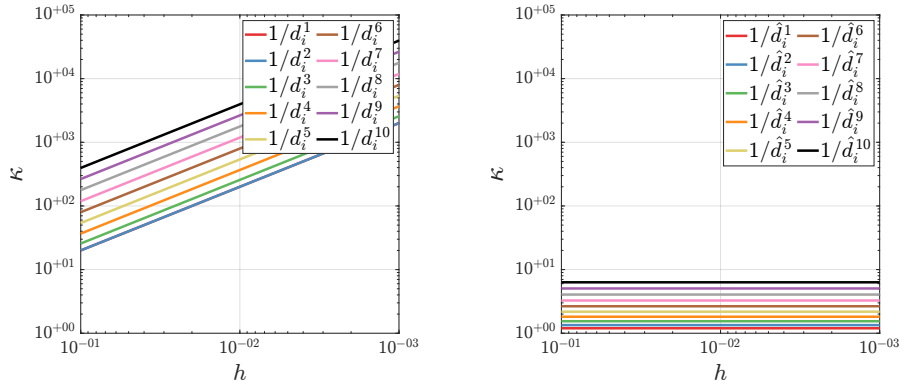
$$\begin{aligned}\hat{d}_i &= \sqrt{\left(\frac{x_i-\mu}{\sigma_x}-\frac{x_f-\mu}{\sigma_x}\right)^2+\left(\frac{y_i-\mu}{\sigma_y}-\frac{y_f-\mu}{\sigma_y}\right)^2}\\ &\equiv \sqrt{\left(\frac{x_i-x_f}{\sigma_x}\right)^2+\left(\frac{y_i-y_f}{\sigma_y}\right)^2}.\end{aligned} \tag{B.6}$$

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



(a) Without polynomial standardisation. **(b)** With polynomial standardisation.

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



(a) Without distance standardisation. **(b)** With distance standardisation.

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

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

References

[1] W. L. Oberkampf, T. G. Trucano, Verification and validation benchmarks, Nuclear Engineering and Design 238 (2008) 716–743. Benchmarking of CFD Codes for Application to Nuclear Reactor Safety.

[2] J. Abanto, D. Pelletier, A. Garon, J.-Y. Trepanier, M. Reggio, Verification of some Commercial CFD Codes on Atypical CFD Problems, in: 43rd AIAA Aerospace Sciences Meeting and Exhibit, American Institute of Aeronautics and Astronautics, Reno, Nevada, 2005.

[3] D. F. Hawken, J. J. Gottlieb, J. S. Hansen, Review of some adaptive node-movement techniques in finite-element and finite-difference solutions of partial differential equations, Journal of Computational Physics 95 (1991) 254–302.

[4] Y. Li, S. Premasuthan, A. Jameson, Comparison of Adaptive h and p Refinements for Spectral Difference Methods, in: 40th Fluid Dynamics Conference and Exhibit, American Institute of Aeronautics and Astronautics, Chicago, Illinois, 2010.

[5] J. P. P. Magalhães, D. M. Albuquerque, J. M. Pereira, J. C. Pereira, Adaptive mesh finite-volume calculation of 2d lid-cavity corner vortices, Journal of Computational Physics 243 (2013) 365–381.

[6] D. M. S. Albuquerque, J. M. C. Pereira, J. C. F. Pereira, Residual Least-Squares Error Estimate for Unstructured h-Adaptive Meshes, Numerical Heat Transfer, Part B: Fundamentals 67 (2015) 187–210.

[7] D. McRae, r-refinement grid adaptation algorithms and issues, Computer Methods in Applied Mechanics and Engineering 189 (2000) 1161–1182. Adaptive Methods for Compressible CFD.

[8] C. Burg, Analytic study of 2D and 3D grid motion using modified Laplacian, International Journal for Numerical Methods in Fluids 52 (2006) 163–197.

[9] L. D. Lorenzis, A. Düster, Modeling in Engineering Using Innovative Numerical Methods for Solids and Fluids, CISM International Centre for Mechanical Sciences, 1st ed., Springer, 2021.

[10] A. Patra, J. T. Oden, Computational techniques for adaptive hp finite element methods, Finite Elements in Analysis and Design 25 (1997) 27–39. Adaptive Meshing, Part 1.

[11] B. R. Ahrabi, W. K. Anderson, J. C. Newman, An adjoint-based hp-adaptive stabilized finite-element method with shock capturing for turbulent flows, Computer Methods in Applied Mechanics and Engineering 318 (2017) 1030–1065.

[12] Z. J. Wang, K. Fidkowski, R. Abgrall, F. Bassi, D. Caraeni, A. Cary, H. Deconinck, R. Hartmann, K. Hillewaert, H. T. Huynh, N. Kroll, G. May, P.-O. Persson, B. van Leer, M. Visbal, High-order CFD methods: current status and perspective, International Journal for Numerical Methods in Fluids 72 (2013) 811–845.

[13] Z. J. Wang, High-order computational fluid dynamics tools for aircraft design, Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences 372 (2014) 20130318.

[14] M. Kronbichler, P.-O. Persson, Efficient High-Order Discretizations for Computational Fluid Dynamics, volume 602 of *CISM International Centre for Mechanical Sciences*, 1st ed., Springer, 2021.

[15] J. Loffeld, J. A. F. Hittinger, On the arithmetic intensity of high-order finite-volume discretizations for hyperbolic systems of conservation laws, The International Journal of High Performance Computing Applications 33 (2019) 25–52.

[16] R. E. Bank, A. H. Sherman, A. Weiser, Some Refinement Algorithms And Data Structures For Regular Local Mesh Refinement, Scientific Computing 1 (1983) 3–17.

[17] J. A. White, H. Nishikawa, R. A. Baurle, Weighted Least-squares Cell-Average Gradient Construction Methods For The VULCAN-CFD Second-Order Accurate Unstructured Grid Cell-Centered Finite-Volume Solver, in: AIAA Scitech 2019 Forum, American Institute of Aeronautics and Astronautics, San Diego, California, 2019.

[18] P. Tsoutsanis, V. A. Titarev, D. Drikakis, WENO schemes on arbitrary mixed-element unstructured meshes in three space dimensions, Journal of Computational Physics 230 (2011) 1585–1601.

[19] M. Dumbser, M. Käser, V. A. Titarev, E. F. Toro, Quadrature-free non-oscillatory finite volume schemes on unstructured meshes for nonlinear hyperbolic systems, Journal of Computational Physics 226 (2007) 204–243.

[20] S. Diot, S. Clain, R. Loubère, Improved detection criteria for the Multi-dimensional Optimal Order Detection (MOOD) on unstructured meshes with very high-order polynomials, Computers & Fluids 64 (2012) 43–63.

[21] S. Clain, S. Diot, R. Loubère, A high-order finite volume method for systems of conservation laws—Multi-dimensional Optimal Order Detection (MOOD), Journal of Computational Physics 230 (2011) 4028–4050.

[22] M. Dumbser, M. Castro, C. Parés, E. F. Toro, ADER schemes on unstructured meshes for nonconservative hyperbolic systems: Applications to geophysical flows, Computers & Fluids 38 (2009) 1731–1748.

[23] A. Jalali, C. F. O. Gooch, Higher-Order Finite Volume Solution Reconstruction on Highly Anisotropic Meshes, in: 21st AIAA Computational Fluid Dynamics Conference, American Institute of Aeronautics and Astronautics, San Diego, California, 2013.

[24] X. Nogueira, L. Cueto-Felgueroso, I. Colominas, F. Navarrina, M. Casteleiro, A new shock-capturing technique based on Moving Least Squares for higher-order numerical schemes on unstructured grids, Computer Methods in Applied Mechanics and Engineering 199 (2010) 2544–2558.

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