MOX-Report No. 03/2023

Scalable Recovery-based Adaptation on Quadtree Meshes for Advection-Diffusion-Reaction Problems

Africa, P.C.; Perotto, S.; de Falco, C.

MOX, Dipartimento di Matematica
Politecnico di Milano, Via Bonardi 9 - 20133 Milano (Italy)

mox-dmat@polimi.it https://mox.polimi.it

Scalable Recovery-based Adaptation on Quadtree Meshes for Advection-Diffusion-Reaction Problems

Pasquale Claudio Africa, Simona Perotto, Carlo de Falco

MOX – Dipartimento di Matematica, Politecnico di Milano, P.zza L. Da Vinci 32, 20133 – Milano – Italy

Abstract

We propose a mesh adaptation procedure for Cartesian quadtree meshes, to discretize scalar advection-diffusion-reaction problems. The adaptation process is driven by a recovery-based a posteriori estimator for the $L^2(\Omega)$-norm of the discretization error, based on suitable higher order approximations of both the solution and the associated gradient. In particular, a metric-based approach exploits the information furnished by the estimator to iteratively predict the new adapted mesh. The new mesh adaptation algorithm is successfully assessed on different configurations, and turns out to perform well also when dealing with discontinuities in the data as well as in the presence of internal layers not aligned with the Cartesian directions. A cross-comparison with a standard estimate–mark–refine approach and with other adaptive strategies available in the literature shows the remarkable accuracy and parallel scalability of the proposed approach.

Keywords: recovery-based error estimator, mesh adaptation, quadtree meshes, parallel computing, finite volumes

1. Introduction

Quadtree-based decomposition of two-dimensional spatial domains is a well-established technique whereby a rectangular region is hierarchically subdivided into smaller rectangles in such a way that each refinement consists of splitting one *parent* rectangle into exactly four *children*, by halving its edges. The quadtree structure was initially devised for the organization of spatial data [1, 2], and was greatly successful due to the early development of efficient algorithms for data sorting and retrieval [3, 4].

While, in the past, simplicial meshes have been slightly favoured over quadtrees because of their superior geometric flexibility, a recent substantial surge in the research interest in methods based on Cartesian quadtree meshes has been driven by two main factors. On the one hand, the many advancements in the field of the embedded interface methods (e.g., the immersed boundary or the fictitious domain approaches), that allow to accurately track interior interfaces, i.e., lower dimensional manifolds within the computational domain where problem coefficients may exhibit discontinuities, without constructing meshes that accurately represent the geometry of the interface [5, 6, 7, 8, 9]. On the other hand, the recent development of highly scalable software libraries [10, 11, 12, 13, 14, 15, 16] based on quadtrees (or octrees in 3D) has identified Cartesian meshes as the ideal tool to efficiently manage extreme scale problems that require massively parallel computer architectures.

As a consequence, in recent years, many instances of numerical schemes based on quadtrees have been used to discretize partial differential equation problems stemming from diverse applications, such as multi-phase and free–surface fluid flows [17, 18] or semiconductor device simulation [19]. In the recent literature, some works focus on correctly adapting quadtree (or octree) meshes to the geometry of the domain boundary as well as to possible internal interfaces [17, 20, 21, 22].

In this paper, we propose a new metric-driven mesh adaptation procedure to sharply approximate the solution to a scalar advection-diffusion-reaction problem. The adaptation is driven by an a posteriori estimator for the $L^2(\Omega)$-norm of the discretization error. In particular, we adopt a recovery-based estimator. In 1987, O.C. Zienkiewicz and J.Z. Zhu propose a very practical way to evaluate the $H^1(\Omega)$-seminorm of

the discretization error, instrumental to generate a mesh able to capture the strong gradient areas of the solution [23]. The idea, successively formalized in the seminal papers [24, 25], is very straightforward. It consists in making explicitly computable the $H^1(\Omega)$-seminorm of the discretization error by replacing the exact gradient with the so-called recovered gradient obtained by properly averaging/projecting the gradient of the discrete solution.
Recovery based error estimators proved to be reliable and computationally efficient in different engineering contexts (see, e.g., [26, 27, 28, 29]), by offering a mathematical tool characterized by several good properties, such as the independence of the specific problem and of the adopted discretization, the handy implementation and the computational cheapness.
In more recent years, the idea of making computable a quantity by means of suitable recovery procedures has been exploited also to evaluate norms of the discretization error different from the $H^1(\Omega)$-seminorm. Examples can be found, for instance, in [30, 31, 32], where the authors propose different recipes to recover the exact solution in order to make computable a certain norm of the associated discretization error. Here, we pursue a similar goal in the case of non-conforming Cartesian, quadtree meshes, by generalizing the approach proposed in [31]. We build a higher-order approximation for the solution gradient by means of suitable finite difference formulas. Then, the recovered gradient is exploited to compute a higher-order approximation of the solution, in order to estimate the $L^2(\Omega)$-norm of the discretization error. In particular, we use one-sided formulas along the interior interfaces in order to guarantee accuracy when coefficient discontinuities, even not aligned with the Cartesian directions, occur within the computational domain. Moreover, in order to improve the scalability of the adaptation procedure, the same one-sided formulas are used in correspondence with the boundaries of the parallel partition of the computationl domain, thus resulting in a reduction of the inter–process communication. Finally, a metric-based approach is shown to further improve the method efficiency, by reducing the number of estimation–adaptation steps with respect to a standard estimate–mark–refine approach [33].

The manuscript is organized as follows: in Section 2 we introduce the reference partial differential equation problem, together with the corresponding discrete formulation based on edge-averaged finite elements on quadtree meshes. Section 3 is devoted to the recovery procedures, and proposes specific recipes to enrich both the discrete gradient and solution. The error estimator for the $L^2(\Omega)$-norm of the discretization error is introduced in Section 4, together with the adopted metric-based mesh adaptation strategy able to predict, in an automatic way, the level of refinement/coarsening of the grid cells. For comparison reasons, in the same section we provide also a standard marking-based mesh adaptation algorithm. An extensive numerical assessment is carried out in Section 5, while some concluding remarks are drawn in the last section.

2. The reference setting

As reference context, we adopt the conservative form of a standard scalar advection-diffusion-reaction (ADR) problem, completed with mixed boundary conditions

$$\begin{cases} -\nabla\cdot(\varepsilon\nabla u-\boldsymbol{\beta}u)+bu=f & \text{in } \Omega\subset\mathbb{R}^2 \\ u=g & \text{on } \Gamma_{\mathrm{D}}\subseteq\partial\Omega \\ \varepsilon\nabla u\cdot\boldsymbol{\nu}=0 & \text{on } \partial\Omega\setminus\Gamma_{\mathrm{D}}, \end{cases} \tag{1}$$

with $\epsilon>0$ the diffusive coefficient; $\boldsymbol{\beta}$ a conservative advective field, i.e., $\boldsymbol{\beta}=\nabla\psi$ for a certain potential ψ, and such that $\nabla\cdot\boldsymbol{\beta}=0$; $b\geq 0$, the reaction; f the source term; g the data assigned on the Dirichlet portion, Γ_D, of the domain boundary, $\partial\Omega$; $\boldsymbol{\nu}$ the unit outward normal vector to $\partial\Omega$. In principle, the problem data can be assumed discontinuous in Ω.

The two next sections are devoted to the discretization of problem (1) by resorting to a finite element scheme. In particular, we will distinguish between Cartesian-product and non-conforming quadtree meshes, after identifying Ω with a Cartesian domain. We highlight that non-Cartesian domains can be handled as well, by using a penalty approach (see, for instance, [20] and Section 5.2.2 for an example).

2.1. An edge-averaged finite element discretization on a Cartesian-product grid

A Cartesian-product mesh is a quadrilateral grid obtained by considering the Cartesian product between a one-dimensional (1D) partition of the edges of Ω aligned with the x- and the y-direction, respectively. We denote by $\tau_h = \left\{\Omega^{(k)}\right\}_{k=1}^{N_{\rm el}}$ a family of Cartesian-product partitions of Ω such that $\Omega = \cup_{k=1}^{N_{\rm el}}\Omega^{(k)}$, with $N_{\rm el}$ the total number of mesh elements and $h = \min_k\left(\mathrm{diam}\left(\Omega^{(k)}\right)\right)$ the minimum diameter of cells $\Omega^{(k)}$.
We introduce the finite element space, $Q_h^1(\tau_h)$, of the continuous piecewise bi-linear polynomials associated with the partition τ_h, where $Q_h^n(\tau_h) = Q_h^{n,n}(\tau_h)$, being

$$Q_h^{n,m}(\tau_h) = \left\{v \in \mathcal{C}^0\left(\overline{\Omega}_h\right) : v|_{\Omega^{(k)}} \in \mathbb{P}_{n,m}(\Omega^{(k)})\ \forall \Omega^{(k)} \in \tau_h\right\}$$

with n, $m \in \mathbb{N}$, and

$$\mathbb{P}_{m,n}(\Omega^{(k)}) = \left\{p_{m,n}(x,y) = \sum_{0\le i\le m,\, 0\le j\le n} a_{ij}x^i y^j,\ \text{with } \mathbf{x} = (x,y) \in \Omega^{(k)}\right\}.$$

From now on, we simplify the notation by dropping any explicit dependence on partition τ_h. After denoting by $X = \{\boldsymbol{x}_i\}_{i=1}^N$ the set of the nodes defining the partition τ_h, we consider a Lagrangian basis $\{\varphi_i\}_{i=1}^N$ of the space Q_h^1, so that

$$\varphi_i(\boldsymbol{x}_j) = \delta_{ij} \quad \forall \boldsymbol{x}_j \in X, \tag{2}$$

δ_{ij} being the Kronecker delta, and the discrete solution u_h to problem (1) can be expressed as a linear combination of such basis functions through the set of (unknown) coefficients $\{U_i\}_{i=1}^N$, being

$$u_h(\boldsymbol{x}) = \sum_{i=1}^N U_i\varphi_i(\boldsymbol{x}) \quad \forall \boldsymbol{x} \in \Omega.$$

In particular, we require that u_h belongs to the subset $V_h \subset Q_h^1$ of the functions satisfying a homogeneous Dirichlet data in correspondence with Γ_D to match the essential boundary conditions in (1). Finally, we observe that functions $\{\varphi_i\}_{i=1}^N$ satisfy a partition of unity property, namely

$$\sum_i \varphi(\boldsymbol{x}) = 1 \quad \forall \boldsymbol{x} \in \Omega. \tag{3}$$

To discretize problem (1), we adopt the Finite-Volume Scharfetter-Gummel (FVSG) method proposed in [34]. This discretization was originally introduced by D.N. De G. Allen and R.V. Southwell in [35] for 1D geometries and successively generalized to two-dimensional (2D) and three-dimensional (3D) diffusion-convection equations by M.A. Zlámal in [36] and to self-adjoint problems by P.A. Markowich and M.A. Zlámal in [37].
In particular, first the original equations are rewritten in terms of the so-called Slotboom variables, this leading to a self-adjoint problem; then, such a problem is discretized by means of a primal-mixed formulation using the harmonic average of the diffusion coefficient over the mesh edges; finally, the problem is rewritten in terms of the primal variables by inverting the Slotboom relations, so that arithmetic overflows are prevented during computations. The trapezoidal quadrature rule is adopted to approximate the integrals resulting from the formulation.
As an example, the matrix yielded by the FVSG discretization when applied to an advection-diffusion problem (e.g., to problem (1) with $b = 0$) and associated with the generic element $\Omega^{(k)}$ in τ_h, whose vertices are numbered by 1 to 4 in a lexicographical order, is given by

$$\begin{bmatrix} \mathcal{B}_{12}^- + \mathcal{B}_{31}^+ & -\mathcal{B}_{12}^+ & -\mathcal{B}_{31}^- & 0 \\ -\mathcal{B}_{12}^- & \mathcal{B}_{12}^+ + \mathcal{B}_{24}^- & 0 & -\mathcal{B}_{24}^+ \\ -\mathcal{B}_{31}^+ & 0 & \mathcal{B}_{43}^+ + \mathcal{B}_{31}^- & -\mathcal{B}_{43}^- \\ 0 & -\mathcal{B}_{24}^- & -\mathcal{B}_{43}^+ & \mathcal{B}_{43}^- + \mathcal{B}_{24}^+ \end{bmatrix}, \tag{4}$$

with

$$\mathcal{B}_{ij}^{+} = \frac{\ell_{ij} h_{ij}^{(k)}(\varepsilon) B(\psi_i - \psi_j)}{2|\Omega^{(k)}|}, \qquad \mathcal{B}_{ij}^{-} = \frac{\ell_{ij} h_{ij}^{(k)}(\varepsilon) B(\psi_j - \psi_i)}{2|\Omega^{(k)}|},$$

for $ij \in \{12, 31, 24, 43\}$, ℓ_{ij} the length of the edge, ij, joining vertices i and j, ψ_k the value of the potential ψ at node k for $k = i, j$, $B(x) = x/(e^x - 1)$ the Bernoulli function, $h_{ij}^{(k)}(f) = |\ell_{ij}|^{-1} \int_{ij} f^{-1}$ the harmonic mean operator over the edge ij, $|\varrho|$ denoting the measure of a generic set $\varrho \subset \mathbb{R}^d$ for $d = 1, 2$.

It has been checked that, under the choices above and over a Cartesian-product mesh, the local matrix (4) ensures the fully-assembled matrix to be an M-matrix. This, in turn, guarantees that the FVSG discretization is monotone, i.e., that a discrete maximum principle holds for the numerical solution to (1) [38].

2.2. An edge-averaged finite element discretization on a non-conforming grid

Non-conforming quadrilateral meshes stem from refinement and coarsening procedures applied to a Cartesian-product mesh.

The quadtree data structure characterizing a Cartesian-product grid allows to easily deal with non-conforming mesh refinement and coarsening, where the former operation consists in replacing an element with four children of equal size, while the latter occurs by replacing four children with their parent [10]. Both refinement and coarsening procedures can lead to the generation of hanging nodes. As a consequence, the mesh cannot be represented as a Cartesian-product set, and the Q_h^1 polynomial space turns out to be not well-defined. Thus, space Q_h^1 is replaced by a new function environment, $\tilde{Q}_h^1$, obtained by imposing in Q_h^1 a set of constraints in order to account for the presence of hanging nodes – which are not included as additional degrees of freedom (dofs) – and to preserve the partition of unity property (3).

We demand that neighboring cells may differ by at most one refinement level, thereby enforcing that only a single hanging node can exist per face. This procedure, referred to as 2:1 balance, simplifies the derivation of the discretization scheme and the recovery procedures presented below. This balance also ensures that the workload is well-balanced among parallel processors, since the computational burden becomes proportional to the number of local cells, rather than to the number of dofs [39], thus making the proposed approach well-suited to frequent adaptations in large-scale computing scenarios [40].

In the remaining part of the section, we analyze more into the details how to manage refinement and coarsening of non-conforming meshes, with a focus also on the definition of space $\tilde{Q}_h^1$. To this aim, we denote by $\tilde{\tau}_h$ the non-conforming mesh obtained by refinement and/or coarsening of the Cartesian-product grid τ_h, and by $\tilde{X}$ the set of nodes defining $\tilde{\tau}_h$.

Refinement of a non-conforming mesh. When refining a mesh element in a non-conforming framework, an additional node (where an additional dof is located) has to be added (see Figure 1, where node 5 denotes the added dof).

Now, we introduce the partition $\bar{\tau}_h$ obtained by uniformly refining τ_h and coinciding with the minimum-size Cartesian-product mesh containing $\tilde{\tau}_h$, and the basis function, $\bar{\varphi}_5 \in Q_h^1(\bar{\tau}_h)$, associated with dof 5 over the uniformly refined mesh $\bar{\tau}_h$.

Now, we associate functions

$$\tilde{\varphi}_i = \varphi_i - \frac{1}{4}\bar{\varphi}_5 \quad i = 1 \dots 4,$$

with the four vertices, $1, \dots, 4$, identifying the element to be refined (see Figure 1), and we define the set $\Phi = \left\{ \{\varphi_i\}_{i \neq 1,\dots,4}, \{\tilde{\varphi}_i\}_{i=1}^4, \bar{\varphi}_5 \right\}$. Thus, the new polynomial space associated with the non-conforming grid $\tilde{\tau}_h$ can be defined as

$$\tilde{Q}_h^1 = \operatorname{span}(\Phi).$$

With an abuse of notation, from now on, we denote by $\tilde{\varphi}_i$ the generic function in Φ. We remark that $\tilde{Q}_h^1$ is a linear subspace of $Q_h^1(\bar{\tau}_h)$ and that the basis functions in Φ satisfy the partition of unity property,

$$\sum_i \tilde{\varphi}_i(\boldsymbol{x}) = 1 \quad \forall \boldsymbol{x} \in \Omega, \tag{5}$$

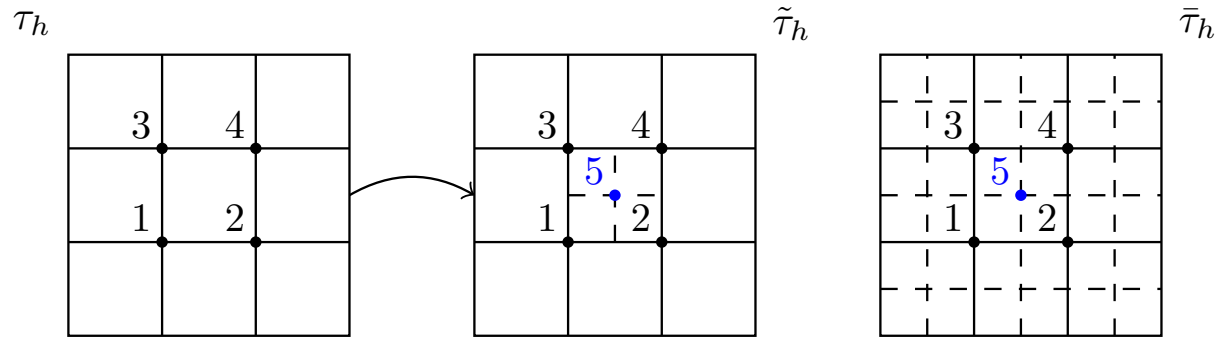


Figure 1: Example of refinement for a non-conforming mesh: initial configuration (left); refined mesh (center), where node 5 corresponds to the newly added degree of freedom; mesh used to define the basis function $\bar{\varphi}_5$ (right).

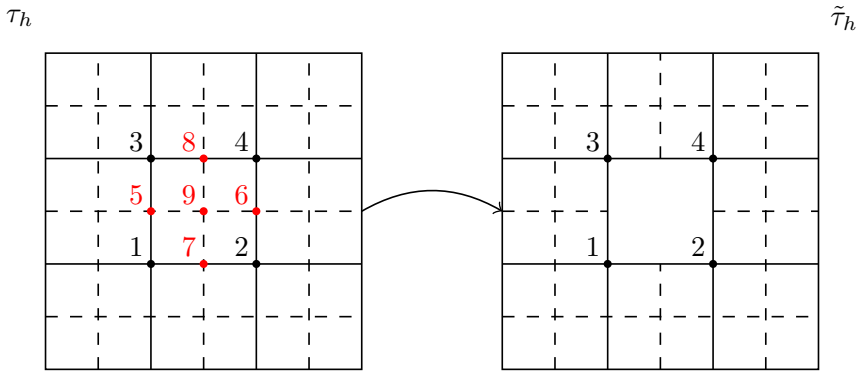


Figure 2: Example of coarsening for a non-conforming mesh: initial configuration (left) where the red nodes highlight the degrees of freedom to be removed; coarsened mesh (right).

as well as the Lagrangian property,

$$\tilde{\varphi}_i(\boldsymbol{x}_j) = \delta_{ij} \quad \forall \boldsymbol{x}_j \in \tilde{X}, \tag{6}$$

analogously as in (2) and (3).

Coarsening of a non-conforming mesh. We exemplify such an operation in Figure 2, where we assume that the cell with vertices 1, 2, 3, 4 has to be coarsened by removing node 9. This action leads to get rid of five dofs (the ones associated with the nodes 5, 6, 7, 8, 9), since nodes 5, 6, 7, 8 become hanging and node 9 does not yet represent a mesh node.
To characterize space $\tilde{Q}_h^1$, we associate the new functions

$$\tilde{\varphi}_i = \varphi_i + \frac{1}{4}\varphi_9 + \frac{1}{2}\sum_{j\in\mathcal{N}_i} \varphi_j \quad i \in 1\dots 4,$$

with vertices 1, 2, 3, 4, where $\mathcal{N}_i$ denotes the set of the hanging nodes sharing an edge with node i, and we define the new set $\Phi = \{\{\varphi_i\}_{i\neq 1,\dots,9}, \{\tilde{\varphi}_i\}_{i=1}^4\}$. Thus, the new discrete space associated with the non-conforming coarsened grid turns out to be

$$\tilde{Q}_h^1 = \text{span}\,(\Phi)\,,$$

which coincides with a linear subspace of space $Q_h^1(\tau_h)$. Functions in Φ still satisfy the partition of unity and the Lagrangian property (5) and (6), respectively.

In practice, a discretization based on a non-conforming mesh generated by refinement and/or coarsening of a Cartesian-product grid, is managed first by assembling the standard local matrices associated with

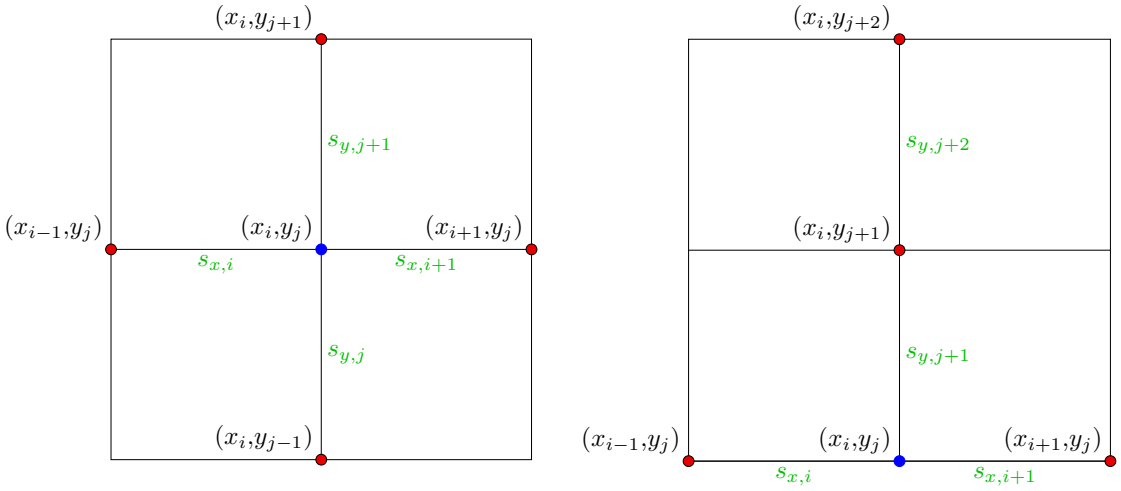


Figure 3: Gradient recovery: patch associated with the internal (left) and the boundary (right) node (x_i, y_j). The mesh edges used in the recovery procedure are labeled in green.

the basis functions of the space Q_h^1; then, the resulting system is extended through a set of equations that constrain the solution at each hanging node to be the arithmetic mean of the solution at the two parent vertices; finally, the constraints are eliminated by static condensation [41]. This technique reduces the size of the linear system by minimizing the coupling among dofs.

3. Recovery procedures

In this section, we present the recovery techniques used to define the error estimator driving the automatic mesh adaptation procedure. To this aim, we extend the results in [31] for triangular grids to quadtree meshes. In particular, given a finite element solution, $u_h \in \tilde{Q}_h^1$, and the associated gradient, $\boldsymbol{\sigma}_h = \nabla u_h \in \tilde{Q}_h^{0,1} \times \tilde{Q}_h^{1,0}$, first we derive a formula to recover an enriched gradient, $\boldsymbol{\sigma}_h^* = \nabla^* u_h \in \left[\tilde{Q}_h^1\right]^2$; successively, we employ $\boldsymbol{\sigma}_h^*$ to build an enriched solution, $u_h^* \in \tilde{Q}_h^2$, coinciding with a piecewise bi-quadratic polynomial.

3.1. Gradient Recovery

To compute the recovered gradient, we resort to an averaging technique.
We denote by $\{s_{x,i}\}_{i=1}^{N_{s,x}}$ and $\{s_{y,j}\}_{j=1}^{N_{s,y}}$ the sets of the mesh edges oriented along the x and y axis, respectively. Now, we consider the patch of elements, $\{\Omega^{(k)} \in \tau_h : (x_i, y_j) \in \Omega^{(k)}\}$, associated with the internal node $(x_i, y_j) \in \Omega$ (see Figure 3 (left)). Since the solution u_h is a bi-linear polynomial, the gradient components can be associated with the degrees of freedom of the Nédélec finite element space [42], and can be exactly computed by using the finite difference formulas

$$\boldsymbol{\sigma}_h\big|_{s_{x,i}} = \begin{bmatrix} \dfrac{u_h(x_i, y_j) - u_h(x_{i-1}, y_j)}{h_{x,i}} \\ 0 \end{bmatrix}, \quad \boldsymbol{\sigma}_h\big|_{s_{y,j}} = \begin{bmatrix} 0 \\ \dfrac{u_h(x_i, y_j) - u_h(x_i, y_{j-1})}{h_{y,j}} \end{bmatrix}, \tag{7}$$

with $h_{x,m} = x_m - x_{m-1}$ and $h_{y,n} = y_n - y_{n-1}$ the length of the generic edges $[x_{m-1}, x_m]$ and $[y_{n-1}, y_n]$. In the sequel, with an abuse of notation, we denote restrictions $\boldsymbol{\sigma}_h\big|_{s_{x,i}}$ and $\boldsymbol{\sigma}_h\big|_{s_{y,j}}$ by $\boldsymbol{\sigma}_h(s_{x,i})$ and $\boldsymbol{\sigma}_h(s_{y,j})$, respectively.

Then, following [31], we compute the recovered gradient, $\boldsymbol{\sigma}_h^*$, at (x_i, y_j) by averaging the values of the discrete gradient, $\boldsymbol{\sigma}_h$, along the x- and y-direction, respectively, with weights depending on the reciprocal

edge lengths, so that we obtain

$$\begin{aligned}\boldsymbol{\sigma}_h^*(x_i,y_j) &= \frac{1}{\dfrac{1}{h_{x,i}}+\dfrac{1}{h_{x,i+1}}}\left(\frac{\boldsymbol{\sigma}_h(s_{x,i})}{h_{x,i}}+\frac{\boldsymbol{\sigma}_h(s_{x,i+1})}{h_{x,i+1}}\right)\\ &+ \frac{1}{\dfrac{1}{h_{y,j}}+\dfrac{1}{h_{y,j+1}}}\left(\frac{\boldsymbol{\sigma}_h(s_{y,j})}{h_{y,j}}+\frac{\boldsymbol{\sigma}_h(s_{y,j+1})}{h_{y,j+1}}\right).\end{aligned} \tag{8}$$

A particular care has to be taken when (x_i, y_j) is a mesh node on the boundary $\partial\Omega$. In such a case, the patch adopted to compute the recovered gradient becomes unilateral and formula (8) has to be modified accordingly. For instance, with reference to Figure 3 (right), we have to consider a patch which is unilateral along the y-direction, while standard along the x-direction, (x_i, y_j) coinciding with a node along the bottom boundary of Ω. Formula (8) is consequently modified into

$$\begin{aligned}\boldsymbol{\sigma}_h^*(x_i,y_j) &= \frac{1}{\dfrac{1}{h_{x,i}}+\dfrac{1}{h_{x,i+1}}}\left(\frac{\boldsymbol{\sigma}_h(s_{x,i})}{h_{x,i}}+\frac{\boldsymbol{\sigma}_h(s_{x,i+1})}{h_{x,i+1}}\right)\\ &+ \frac{1}{\dfrac{1}{h_{y,j+1}}+\dfrac{1}{h_{y,j+2}}}\left[\boldsymbol{\sigma}_h(s_{y,j+1})\left(\frac{1}{h_{y,j+1}}+\frac{2}{h_{y,j+2}}\right)-\frac{\boldsymbol{\sigma}_h(s_{y,j+2})}{h_{y,j+2}}\right],\end{aligned} \tag{9}$$

which falls back to a standard three-point forward-difference formula. Once a value for the recovered gradient is computed at each dof in $\tilde{Q}_h^1$, we are able to define the recovered gradient $\boldsymbol{\sigma}_h^* \in [\tilde{Q}_h^1]^2$. Notice that the presence of hanging nodes does not modify the procedure, since formulas (8)-(9) employ only the tangential component of the gradient along each mesh edge, which is continuous also in the case of non-conforming meshes.

We remark that we exploit the one-sided formulas in (9) also at the interfaces between regions that are assigned to different processors in a distributed-memory parallel processing paradigm. This strategy allows us to highly reduce the amount of inter–process communication in the recovery procedure, thus ensuring a better scalability (see Section 5 for more details).

The following property features the gradient recovery procedure just formalized:

Proposition 1. *Let $\Omega \subset \mathbb{R}^2$ be an open, bounded domain and let $\tau_h = \{\Omega^{(k)}\}_k$ be a quadtree partition of Ω. Let $u \in \mathbb{P}_{2,2}(\Omega)$ be a bi-quadratic polynomial. Then, the recovery procedure (8)-(9) applied to the bi-linear interpolant $\Pi_h^1 u \in \tilde{Q}_h^1$ of u exactly recovers the gradient $\boldsymbol{\sigma} = \nabla u$ at the partition vertices.*

Proof. With reference to Figure 3, we aim to prove that $\boldsymbol{\sigma}_h^*(x_i, y_j) = \boldsymbol{\sigma}(x_i, y_j)$ for each internal node (x_i, y_j), being $\boldsymbol{\sigma}_h^*$ and $\boldsymbol{\sigma}_h$ a piecewise linear and a linear polynomial, respectively. To show this, we proceed component-wise. Since u is a bi-quadratic polynomial, the following relations hold for the x-component σ_x of the exact gradient:

$$\begin{aligned}\sigma_x(s_{x,i}) &= \frac{\Pi_h^1 u(x_i,y_j)-\Pi_h^1 u(x_{i-1},y_j)}{h_{x,i}},\\ \sigma_x(s_{x,i+1}) &= \frac{\Pi_h^1 u(x_{i+1},y_j)-\Pi_h^1 u(x_i,y_j)}{h_{x,i+1}},\\ \sigma_x(x_i,y_j) &= \frac{1}{\dfrac{1}{h_{x,i}}+\dfrac{1}{h_{x,i+1}}}\left(\frac{\sigma_x(s_{x,i})}{h_{x,i}}+\frac{\sigma_x(s_{x,i+1})}{h_{x,i+1}}\right),\end{aligned} \tag{10}$$

where the last equality follows from the linearity of $\boldsymbol{\sigma}$.

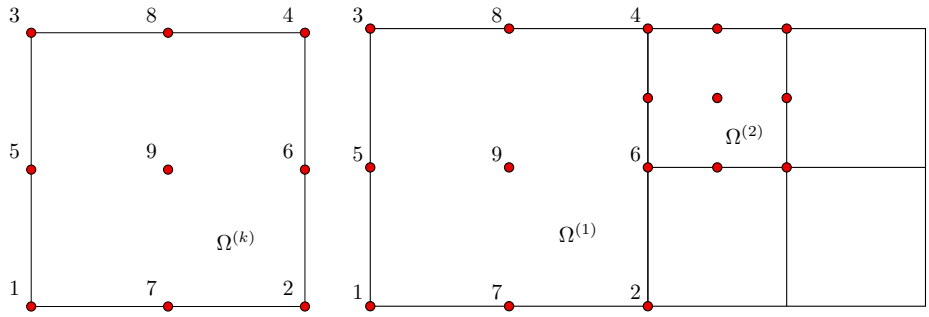


Figure 4: Solution recovery: local numbering of dofs in $\Omega^{(k)}$ characterizing the space $\tilde{Q}_h^2$ (left); sketch of a non-conforming configuration when interested in recovering the solution at dof 6 (right).

On the other hand, procedure (8) applied to the computation of the x-component $\sigma_{h,x}^*$ of the recovered gradient of $\Pi_h^1 u$ yields

$$\sigma_{h,x}^*(x_i, y_j) = \frac{1}{\dfrac{1}{h_{x,i}} + \dfrac{1}{h_{x,i+1}}} \left(\frac{\sigma_{h,x}(s_{x,i})}{h_{x,i}} + \frac{\sigma_{h,x}(s_{x,i+1})}{h_{x,i+1}} \right), \tag{11}$$

with $\sigma_{h,x}$ the x-component of the discrete gradient, which, by using (7) and the definition of interpolation, provides $\sigma_x(x_i, y_j) = \sigma_{h,x}^*(x_i, y_j)$.
The same procedure can be repeated for the y-component of the gradient and for boundary nodes, which concludes the proof. □

In particular, if $u \in \mathbb{P}_{2,2}(\Omega)$, then $\boldsymbol{\sigma}_h^*(s_{x,i}) = \boldsymbol{\sigma}(s_{x,i})$ and, analogously, $\boldsymbol{\sigma}_h^*(s_{y,j}) = \boldsymbol{\sigma}(s_{y,j})$. Thus, we can state that, on each mesh edge, the tangential component of the recovered gradient, $\boldsymbol{\sigma}_h^*$, provides an exact representation of the tangential component of the exact gradient, $\boldsymbol{\sigma}$.

As a consequence of Proposition 1, we have also numerically verified that the recovered gradient is super-convergent with respect to the mesh size h (we refer, e.g., to Fig. 7). In particular, it holds

$$\|\boldsymbol{\sigma}_h^* - \boldsymbol{\sigma}\|_{L^2(\Omega)} = O(h^2).$$

3.2. Solution Recovery

In this section we present a procedure that, starting from the recovered gradient $\boldsymbol{\sigma}_h^*$ in (8)-(9), enables us to recover an enriched solution $u_h^* \in \tilde{Q}_h^2$. We remind that the polynomials in this space are identified by 9 dofs on each mesh element $\Omega^{(k)}$.
To explain the recovery procedure, we adopt the numbering of the dofs in Figure 4 (left), and we denote by $u_{h,i} = u_h(\boldsymbol{x}_i)$ the value taken by the discrete solution u_h at the mesh node $\boldsymbol{x}_i$. Now, we set the recovered solution to coincide with the discrete solution at the vertices of $\Omega^{(k)}$, i.e.,

$$u_{h,i}^*\big|_{\Omega^{(k)}} = u_h^*\big|_{\Omega^{(k)}}(\boldsymbol{x}_i) = u_{h,i} \quad i = 1, \dots, 4, \tag{12}$$

while we properly integrate the recovered gradient $\boldsymbol{\sigma}_h^*$ to compute the values of $u_h^*\big|_{\Omega^{(k)}}$ at the dofs discriminating a $\tilde{Q}_h^2$ from a $\tilde{Q}_h^1$-approximation (namely, dofs $5, \dots, 9$). We exemplify the adopted recovery process on dof 5. After denoting by

$$u_5^{(1)} = u_{h,1} + \int_{y_1}^{y_5} \sigma_{h,y}^* \, \mathrm{d}y, \qquad u_5^{(2)} = u_{h,3} - \int_{y_5}^{y_3} \sigma_{h,y}^* \, \mathrm{d}y \tag{13}$$

with $\sigma_{h,y}^*$ the y-component of the recovered gradient, we assign the value

$$u_{h,5}^*\big|_{\Omega^{(k)}} = \frac{u_5^{(1)} + u_5^{(2)}}{2} \tag{14}$$

to dof 5. We notice that the gradient recovery procedure in Section 3.1 does not guarantee, a priori, that $\boldsymbol{\sigma}_h^*$ is a conservative vector field. Thus, the two line integrals in (13) could differ each as they depend on the integration path. The same approach as in (13)-(14) is adopted to recover the value of the solution at the midpoint dofs $6, 7, 8$.
Finally, the value associated with the centroid of the element (i.e., with dof 9) is computed as

$$u_{h,9}^*\big|_{\Omega^{(k)}} = \frac{1}{4}\sum_{i=1}^{4} u_9^{(i)}, \tag{15}$$

with

$$\begin{aligned} u_9^{(1)} &= u_{h,5} + \int_{x_5}^{x_9} \sigma_{h,x}^* \,\mathrm{d}x, \qquad & u_9^{(2)} &= u_{h,6} - \int_{x_9}^{x_6} \sigma_{h,x}^* \,\mathrm{d}x, \\ u_9^{(3)} &= u_{h,7} + \int_{y_7}^{y_9} \sigma_{h,y}^* \,\mathrm{d}y, \qquad & u_9^{(4)} &= u_{h,8} - \int_{y_9}^{y_8} \sigma_{h,y}^* \,\mathrm{d}y, \end{aligned} \tag{16}$$

and $\sigma_{h,x}^*$, $\sigma_{h,y}^*$ the x- and the y-component of the recovered gradient, respectively. All the integrals in (13) and (16) are exactly computed by the midpoint quadrature rule, since the functions to be integrated are linear polynomials.

In the presence of non-conforming meshes, the procedure to recover the solution at hanging nodes is properly modified. With reference to Figure 4 (right), let us assume to be interested in recovering u_h^* at node 6. Instead of setting $u_{h,6}^*|_{\Omega^{(2)}} = u_{h,6} = \frac{1}{2}(u_{h,2} + u_{h,4})$, we assign

$$u_{h,6}^*\big|_{\Omega^{(2)}} = u_{h,6}^*\big|_{\Omega^{(1)}}, \tag{17}$$

i.e., the value recovered at the same point as a dof of the element $\Omega^{(1)}$. It is straightforward to verify that this choice guarantees that the recovered solution u_h^* is continuous also across non-conforming edges.

A result analogous to Proposition 1 can be proved also for the recovered solution, as stated below.

Proposition 2. *Let $\Omega \subset \mathbb{R}^2$ be an open, bounded domain and let $\tau_h = \{\Omega^{(k)}\}_k$ be a quadtree partition of Ω. Let $u \in \mathbb{P}_{2,2}(\Omega)$ be a bi-quadratic polynomial. Then, the solution recovery procedure* (12)-(17) *applied to the bi-linear interpolant $\Pi_h^1 u \in \tilde{Q}_h^1$ of u exactly recovers the solution u at the partition vertices.*

Also for the recovered solution, we have numerically verified (see Fig. 7) a convergence of order 2 to the exact solution with respect to the $L^2(\Omega)$-norm, being

$$\|u_h^* - u\|_{L^2(\Omega)} = O(h^2).$$

4. The mesh adaptation procedure

To carry out the mesh adaptation process, we resort to the generalization of the approach proposed in [31] to a quadtree tassellation of the domain Ω. In particular, we adopt the a posteriori error estimator

$$\eta = \left[\sum_{k=1}^{N_{\mathrm{el}}} \eta_k^2\right]^{\frac{1}{2}}$$

for the L^2-norm, $\|u - u_h\|_{L^2(\Omega)}$, of the discretization error, with

$$\eta_k = \|u_h^* - u_h\|_{L^2\left(\Omega^{(k)}\right)} \tag{18}$$

the estimator associated with the mesh element $\Omega^{(k)}$ (notice that the enhanced gradient, $\boldsymbol{\sigma}_h^*$, is used only to recover the enriched solution u_h^*).
The current goal is to employ the information provided by η and, in particular, by η_k, to generate a

computational mesh able to follow the features of the solution at hand. To this aim, we exploit an iterative procedure which, starting from an initial (possibly uniform) grid, predicts a new mesh characterized by a refinement of the elements where the solution features an irregular behaviour, while exhibiting an element coarsening where a smooth trend is detected.

Below, we investigate two alternative strategies to drive the mesh adaptation. The former is based on a standard solve $\rightarrow$ estimate $\rightarrow$ mark $\rightarrow$ adapt algorithm [33], whereas the latter exploits the concept of metric [43]. In either cases, the strategy we adopt consists in minimizing the mesh cardinality while ensuring a user-defined accuracy, tol, on the $L^2(\Omega)$-norm of the discretization error (i.e., on the estimator η), together with an error equidistribution across the grid elements, so that

$$\eta_k \simeq \frac{\text{tol}}{\sqrt{N_{\text{el}}}}. \tag{19}$$

This check can be tuned by a constant in order to penalize the elements that most infringe criterion (19), both from above and from below (we refer to parameters δ_1 and δ_2 in Algorithm 1).

To stop the iterative procedure, different choices can be made. We set a maximum number of iterations to keep affordable the computational effort. As an alternative, we could monitor the stagnation of mesh elements by checking whether the relative variation in the mesh cardinality of two successive grids is smaller than a fixed threshold.

In the next sections, we introduce the marking-based and the metric-based approaches followed to generate adapted meshes, starting from estimator η. All the numerical tests presented hereby have been performed using `bim++`, an in-house `C++` library, with utility classes and methods to solve ADR problems, with an interface to `p4est` for quadtree mesh support.

4.1. A marking-based mesh adaptation algorithm

As a first approach, we resort to a standard solve $\rightarrow$ estimate $\rightarrow$ mark $\rightarrow$ adapt iterative algorithm. The whole procedure is itemized in Algorithm 1, whose input parameters coincide with the initial mesh, $\tau_h^{(0)}$, the tolerance, tol, set by the user on the estimator η, and the maximum number of iterations, $i_{\max}$, constraining the adaptive loop.
After computing the discrete solution on the current mesh, the local error estimator (18) is evaluated, and the mesh elements are consequently marked to be refined, coarsened or kept unaltered, according to an error equidistribution principle. Then, cells $\Omega^{(k)}$ are modified consistently with the received label by resorting to the local hierarchical cell refinement and coarsening operations, according to what detailed in Section 2.2.
Parameters δ_1 and δ_2 are meant to confine the refinement and the coarsening phase to the elements characterized by the largest and by the smallest values for η_k , respectively in terms of error equidistribution. In particular, we set $\delta_1 = 1.5$ and $\delta_2 = 0.5$ in the numerical assessment below.

4.2. A metric-based mesh adaptation algorithm

The second adaptive approach coincides with a a metric-based procedure. In such a case, the error estimator is used to directly predict the spacing (i.e., the metric) characterizing the new mesh, instead of selecting the operation (refinement/coarsening/preservation) to be performed on each cell. In more detail, according to Section 4.1, the prediction of the new spacing is driven by a certain accuracy to be guaranteed to the error estimator, together with the minimization of the mesh cardinality and the equidistribution of the error (namely, of the estimator).
Since we deal with isotropic meshes, the prediction of the spacing characterizing the new grid reduces to the computation of the cell side, for each cell of the new mesh. To this aim, we first scale the local error estimator in (18) with respect to the area of the corresponding element, in order to confine, at least asymptotically (i.e., when the mesh is sufficiently fine), the information related to the measure of $\Omega^{(k)}$. This leads us to define the scaled estimator

$$\hat{\eta}_k^2 = \frac{\eta_k^2}{\left[h_x^{(k)}\right]^2}, \tag{20}$$

Input: $\tau_h^{(0)}$; tol; $i_{\max}$;
$i = 0$;
while $i \leq i_{\max}$ **do**
  compute u_h;
  for $k = 1, \ldots, N_{\text{el}}$ **do**
    compute η_k;
    if $\eta_k \geq \delta_1 \dfrac{\text{tol}}{\sqrt{N_{\text{el}}}}$ **then**
      mark $\Omega^{(k)}$ for refinement;
    else if $\eta_k \leq \delta_2 \dfrac{\text{tol}}{\sqrt{N_{\text{el}}}}$ **then**
      mark $\Omega^{(k)}$ for coarsening;
    else
      keep $\Omega^{(k)}$
    end
  end
  refine and/or coarsen the mesh;
  $i = i + 1$;
end

Algorithm 1: Workflow for the marking-based adaptation.

where $h_x^{(k)}$ denotes a characteristic size of $\Omega^{(k)}$ such as, for instance, the cell edge length along the x-direction. Now, we exploit the equidistribution constraint (19) to predict the new size $h_{x,\text{new}}^{(k)}$ of the element $\Omega^{(k)}$, being

$$\hat{\eta}_k^2 \left[h_{x,\text{new}}^{(k)}\right]^2 \approx \frac{\text{tol}^2}{N_{\text{el}}}. \tag{21}$$

Relations (20) and (21), combined together, yield

$$\frac{h_{x,\text{new}}^{(k)}}{h_x^{(k)}} \approx \frac{\text{tol}}{\eta_k \sqrt{N_{\text{el}}}}.$$

We recall that, in the case of quadtree meshes, the relation $h_{x,\text{new}}^{(k)}/h_x^{(k)} = 2^{-\ell_k}$ holds, so that the number, ℓ_k, of refinement or coarsening steps required to reach the desired size, $h_{x,\text{new}}^{(k)}$, can be computed as

$$\ell_k = \left\lceil \log_2 \left(\frac{\eta_k \sqrt{N_{\text{el}}}}{\text{tol}} \right) \right\rceil, \tag{22}$$

where the choice between refinement and coarsening depends on the sign of the ratio $h_{x,\text{new}}^{(k)}/h_x^{(k)}$.
Finally, to have a sharper control on the quality of the generated mesh, we resort to two additional parameters, n_{ref} and n_{coarsen}, with $0 \leq n_{\text{ref}}, n_{\text{coarsen}} \leq \ell_k$, which modify the prediction for ℓ_k in (22) into the new quantity

$$\ell_k = \begin{cases} \max(0, \ell_k - n_{\text{ref}}) & \text{if } \ell_k \geq 0, \\ \min(0, \ell_k + n_{\text{coarsen}}) & \text{if } \ell_k < 0. \end{cases}$$

Parameters n_{ref} and n_{coarsen} have a role similar to the one played by δ_1 and δ_2 in the marking-based approach, and provide a bound to the predicted maximum number of refinement/coarsening steps. This becomes essential when small tolerance values are assigned or when dealing with large meshes.
The metric-based adaptation procedure is itemized in Algorithm 2.
The predictive feature represents the main advantage of a metric-based approach. In general, a metric-based approach converges to the adapted mesh after a smaller number of iterations when compared with a

Input: $\tau_h^{(0)}$; tol; $i_{\max}$;
$i = 0$;
while $i \leq i_{\max}$ **do**
  compute u_h;
  for $k = 1, \ldots, N_{\text{el}}$ **do**
    compute η_k;
    compute the number ℓ_k of refinement/coarsening steps;
    refine/coarsen $\Omega^{(k)}$ ℓ_k times;
  end
  $i = i + 1$;
end

Algorithm 2: Workflow for the metric-based adaptation.

marking-based algorithm. Nevertheless, as we will verify in the next section, Algorithm 2 requires the initial mesh to be properly selected: for instance, should it be too coarse, ℓ_k would be overestimated during the first steps, thus generating overly-refined grids at intermediate iterations.

5. Numerical results

In this section, we assess the performance of both the marking-based and the metric-based mesh adaptation procedures on diffusion-reaction and advection-diffusion problems. The presence of boundary and internal layers as well as of a possible discontinuity in the problem data further support the adoption of a mesh customized to the problem at hand. Finally, the computational footprint of Algorithms 1 and 2 are analyzed, thus confirming the efficiency and the scalability of the proposed adaptation procedures.

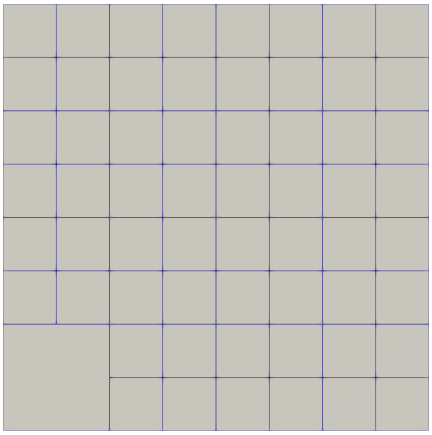
5.1. Test case 1: a diffusion-reaction problem with multiple layers

We solve problem (1) in the domain $\Omega = (0, 1)^2$ when dealing with constant coefficients. In particular, we set $\varepsilon = 10^{-4}$, $\beta = 0$, $b = 1$, while f and g are chosen such that the exact solution is

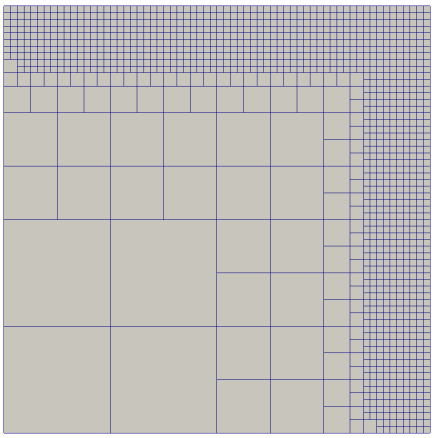
$$u_{\text{ex}}(x, y) = \left(1 - \frac{\sinh(x/\sqrt{\varepsilon})}{\sinh(1/\sqrt{\varepsilon})}\right)\left(1 - \frac{\sinh(y/\sqrt{\varepsilon})}{\sinh(1/\sqrt{\varepsilon})}\right),$$

and Γ_D coincides with the whole boundary $\partial\Omega$. Solution u_{ex} exhibits a boundary layer both along the top and the right side of Ω.
Algorithms 1 and 2 are run starting from the same initial mesh, i.e., a uniform mesh consisting of 16 squared cells, and by setting $i_{\max} = 10$ and $\text{tol} = 10^{-5}$. Algorithm 1 converges after 9 iterations, while the metric-based procedure demands only 3 iterations. Figures 5 and 6 show the evolution of the mesh throughout the two different adaptive procedures. The boundary layers are correctly refined by both algorithms. However, it is evident that the metric-based approach takes less iterations (3 adaptations are enough to capture the two layers in contrast to the 9 iterations demanded by Algorithm 1) and detects the two layers with a higher sharpness, as highlighted by the thinner refined areas.

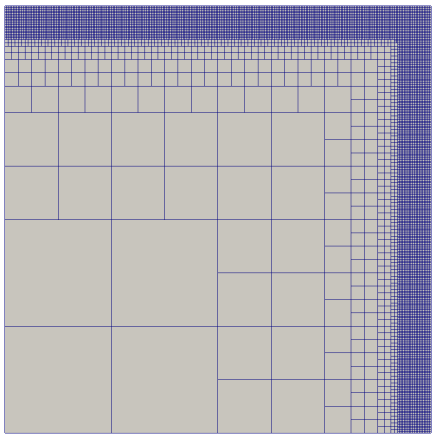
In Figure 7 (left) we investigate the approximation property of the recovered quantities u_h^* and $\boldsymbol{\sigma}_h^*$. In particular, we compare the convergence of the $L^2(\Omega)$-norm of the discretization errors, $(u_{\text{ex}} - u_h)$ and $(\nabla u_{\text{ex}} - \nabla u_h)$, as a function of the minimum mesh size, $h_{\min}$, with the trend of the quantities, $(u_{\text{ex}} - u_h^*)$ and $(\nabla u_{\text{ex}} - \boldsymbol{\sigma}_h^*)$, respectively. The expected order of convergence is guaranteed, at least asymptotically, by all the errors, with a superconvergent behaviour for the error associated with the recovered gradient. Additionally, it turns out that both the recovered gradient and solution provide a better approximation when compared with the corresponding discrete quantities, thus confirming the reliability of the recovered procedures set in Sections 3.1 and 3.2.
The right panel in Figure 7 highlights the gain in terms of dofs ensured by the metric-based adaptive



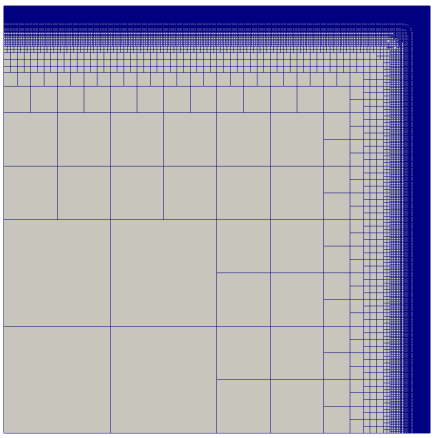
(a) Adapted mesh: $i = 1$.



(b) Adapted mesh: $i = 4$.

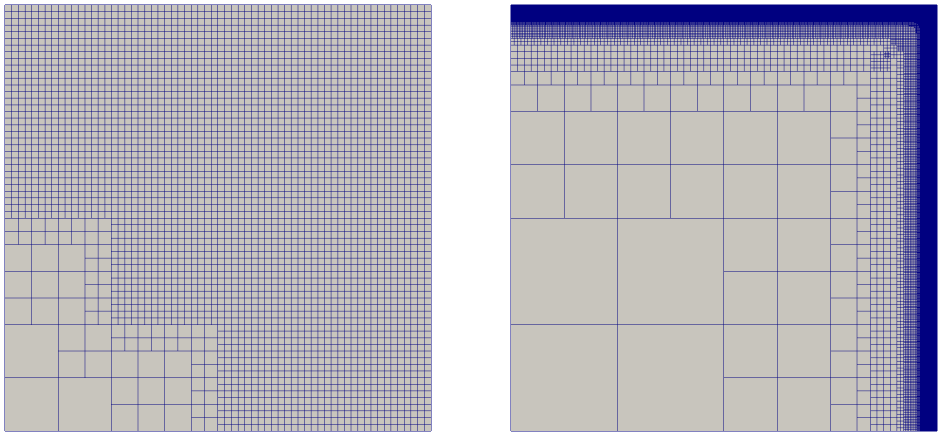


(c) Adapted mesh: $i = 6$.



(d) Adapted mesh: $i = 9$.

Figure 5: Test case 1. Marking-based adaptation.



(a) Adapted mesh: $i = 1$.

(b) Adapted mesh: $i = 3$.

Figure 6: Test case 1. Metric-based adaptation.

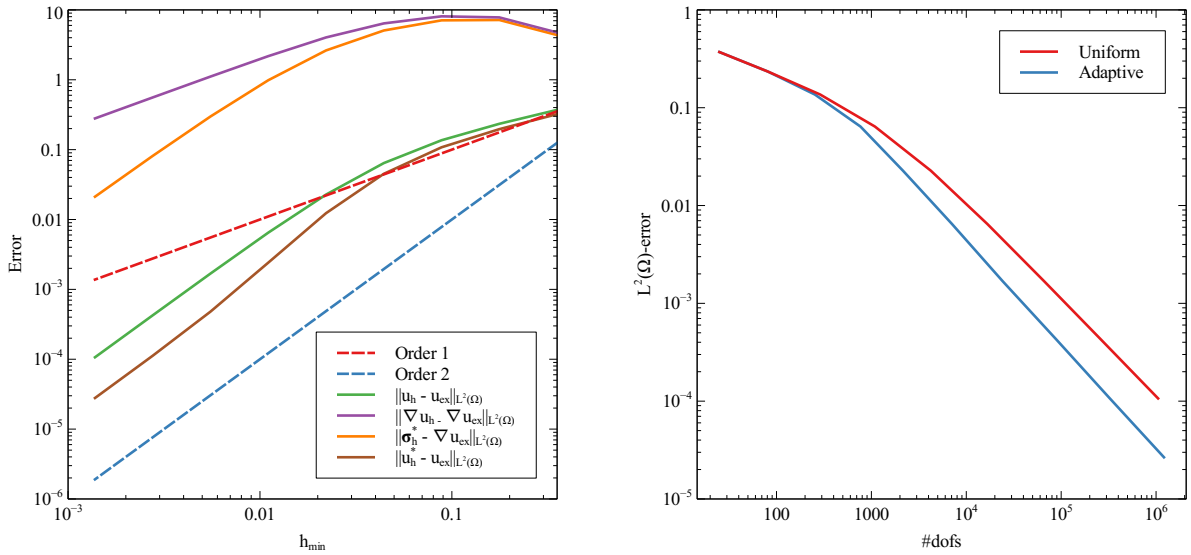


Figure 7: Test case 1. Convergence analysis of the error associated with the discrete and the recovered solution and gradient, as a function of the mesh size $h_{\min}$ (left); log-log plot of the norm $\|u - u_h\|_{L^2(\Omega)}$ as a function of the number of dofs on a uniform and on a metric-based adapted mesh (right).

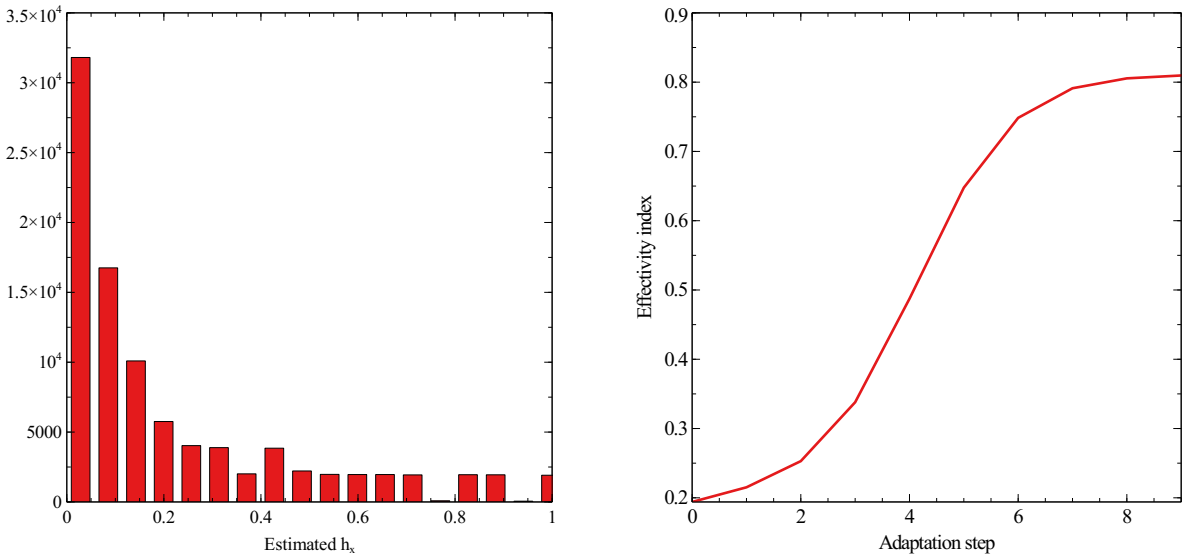


Figure 8: Test case 1. Histogram of the element size computed at the last iteration of the metric-based adaptive procedure (left); trend of the effectivity index as a function of the adaptation step (right).

procedure when compared to a uniform refinement of τ_h. For a sufficiently large number of dofs, about half of the unknowns required by a uniform refinement process is demanded by the metric-based adaptive algorithm to ensure a certain accuracy on the discrete solution. In particular, in Figure 8 (left), we show the histogram of the elements size predicted by the metric at the last iteration of the metric-based adaptive procedure.

Furthermore, we verify the robustness of the error estimator η by computing the effectivity index

$$\xi = \frac{\eta}{\|u - u_h\|_{L^2(\Omega)}},$$

with an ideal value equal to 1. The value of ξ at each adaptation step is provided in Figure 8 (right). We observe that η slightly under-estimates the exact error, with a scaling factor between 0.8 and 1. Nevertheless, ξ exhibits a stagnation trend across the adaptive iterations, which confirms the robustness of the proposed estimator.

In order to assess the scalability performance of the marking-based adaptation procedure, we have run Algorithm 1 on a different number of parallel processes. The results are summarized in Figure 9. In the left panel, we show the speedup as a function of the number of the parallel processes, for the different phases of the algorithm, namely the recovery of the gradient $\boldsymbol{\sigma}_h^*$ (as described in Section 3.1) and of the solution u_h^* (as described in Section 3.2), the computation of the error estimator η_k over each mesh element (as described in Section 4) and the mesh adaptation. All these phases exhibit an almost linear speedup, up to 1024 cores, with the only exception being the gradient recovery whose trend deteriorates starting from 128 cores. This can be ascribed to an unbalanced load among processes due to the presence of the two very narrow boundary layers. Indeed, while the solution recovery can be performed independently on each cell, the gradient recovery involves an exchange of data among neighboring elements. The demanding computational footprint characterizing the recovery of the gradient is confirmed by the right panel in Figure 9, which displays the percentage of the computational time associated with each phase of the marking-based algorithm. Finally, we remark that the whole adaptation procedure requires a contained computational time (between 1 and 50s), thus providing an efficient tool to deal with more challenging scenarios, such as large-scale or time-dependent problems.

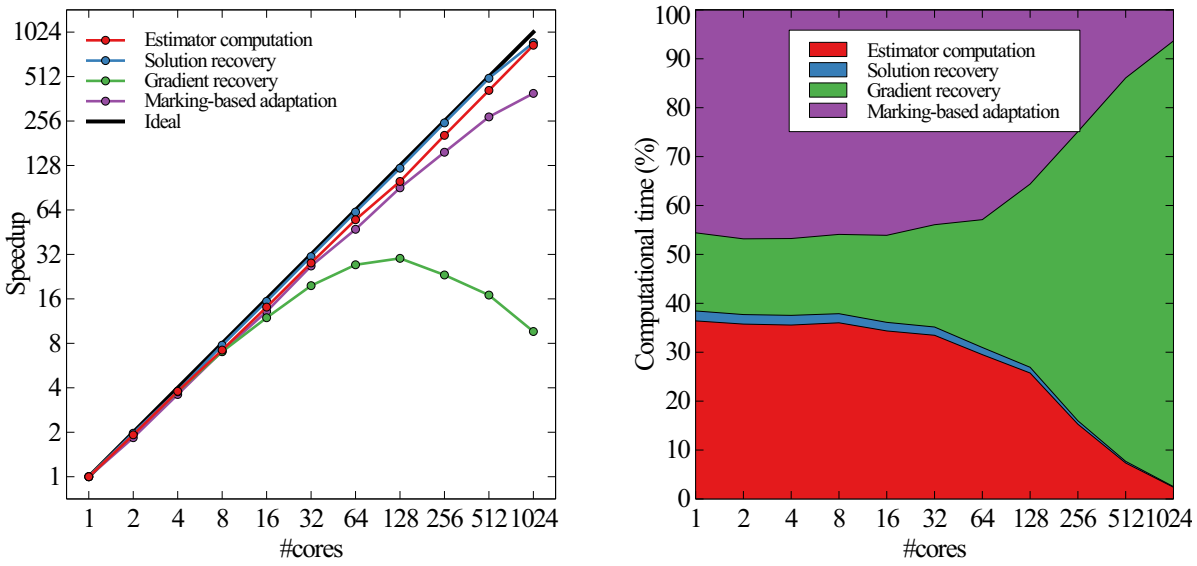


Figure 9: Test case 1. Scalabilty analysis of the different phases of the marking-based adaptation procedure: parallel speedup (left) and stacked barplot of the time percentage associated with the different phases of the algorithm (right) as a function of the number of parallel processes.

5.2. Test case 2: diffusion-reaction problems with discontinuous data

In this section we examine two configurations which exhibit a discontinuity, rectilinear and circular, respectively, The latter setting offers a challenging benchmark since dealing with a Cartesian mesh adaptation procedure.

5.2.1. Rectilinear discontinuity

We consider a diffusion-reaction problem firstly proposed in [44], characterized by a discontinuous diffusion and reaction. We solve problem (1) in the unit square $\Omega = (0,1)^2$, by setting

$$\varepsilon = \begin{cases} \varepsilon_1 = 5 \cdot 10^{-5} & \text{in } \Omega_1 \\ \varepsilon_2 = 10^{-1} & \text{in } \Omega_2, \end{cases}$$

with $\Omega_1 = \{(x,y) : y \leq 0.5\}$ and $\Omega_2 = \Omega \setminus \Omega_1$, $\beta = 0$,

$$b = \begin{cases} 1 & \text{in } \Omega_1 \\ 0 & \text{in } \Omega_2, \end{cases} \qquad f = \begin{cases} 1 & \text{in } \Omega_1 \\ \varepsilon_2 & \text{in } \Omega_2. \end{cases}$$

Concerning the boundary data, we select

$$g = \begin{cases} 1 & \text{on } \Gamma_{D1} \\ 0 & \text{on } \Gamma_{D2}, \end{cases}$$

with $\Gamma_{D1} = \{(x,y) : y = 0\}$, $\Gamma_{D2} = \{(x,y) : y = 1\}$. This choice of data leads us to identify function

$$u_{\text{ex}}(x,y) = \begin{cases} 1 + 2c_1 \sinh(y/\sqrt{\varepsilon_1}) & \text{in } \Omega_1 \\ -0.5(y-1)(y+2c_2) & \text{in } \Omega_2, \end{cases}$$

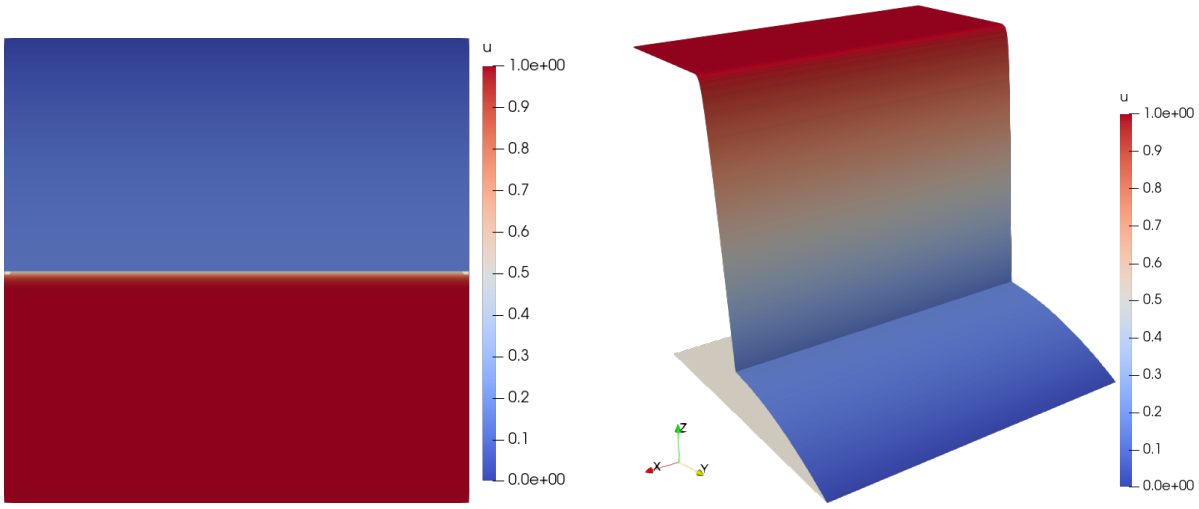


Figure 10: Test case 2 (rectilinear discontinuity). 2D (left) and 3D (right) view of the exact solution.

with the exact solution to the problem, for

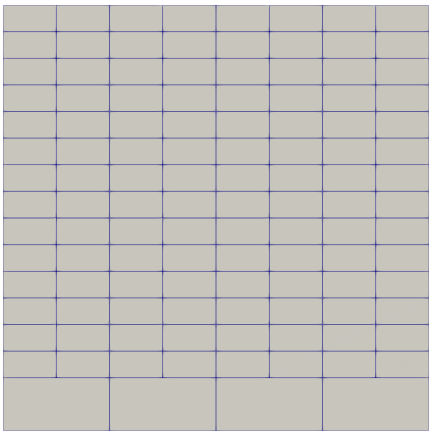
$$c_1 = -\frac{7\varepsilon_2}{8\sqrt{\varepsilon_1}\cosh(0.5/\sqrt{\varepsilon_1}) + 16\varepsilon_2\sinh(0.5/\sqrt{\varepsilon_1})}$$

$$c_2 = \frac{7\sqrt{\varepsilon_1}\cosh(0.5/\sqrt{\varepsilon_1})}{4\sqrt{\varepsilon_1}\cosh(0.5/\sqrt{\varepsilon_1}) + 8\varepsilon_2\sinh(0.5/\sqrt{\varepsilon_1})}.$$

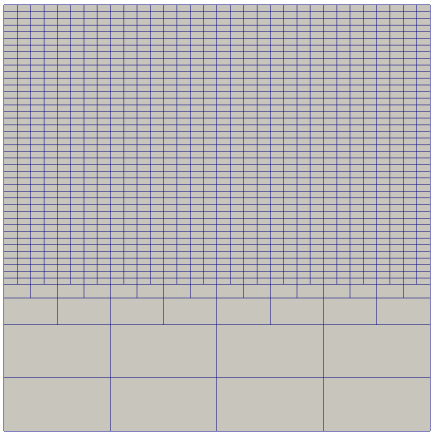
Function u_{ex} exhibits a strictly one-dimensional dynamics along the y-direction, together with a jump across the line $\{(x, y) : y = 0.5\}$ (see Fig. 10).

We resort to both Algorithms 1 and 2 to approximate u_{ex} in order to compare the performance of the two adaptive procedures. To this aim, we pick a uniform initial mesh characterized by 32 rectangular cells (4 and 8 subdivisions along the x- and the y-direction, respectively), while assigning values 10^{-6} and 10 to the input parameters tol and $i_{\max}$, respectively.
The marking-based procedure takes 9 iterations to converge (see Figure 11), while the metric-based algorithm breaks after only 3 iterations by delivering the adapted mesh shown in Figure 12 (right). Both the adaptive procedures start by refining the upper part of the domain; successively, the refinement is further increased in correspondence with the horizontal layer, while the mesh elements are coarsened in the bottom portion of Ω. In particular, we observe that, at the first iteration, the metric-based approach more massively refine the upper part of Ω with respect to the marking-based algorithm. This is compliant with the quickest convergence of Algorithm 2, analogously to Test case 1. Moreover, the layer turns out to be more sharply captured by the metric-based approach.

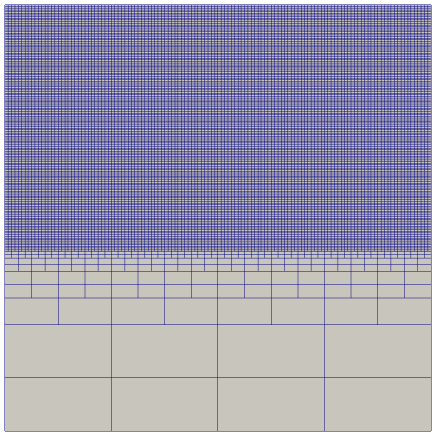
The quantitative analysis in Figures 7 and 8 is replicated for the configuration at hand. The convergence analysis in Figure 13 (left) confirms the high reliability of the recovered solution, u^*, and gradient, $\boldsymbol{\sigma}_h^*$, which better approximate u and ∇u when compared with u_h and ∇u_h, respectively. The orders of convergence are preserved, with a significant superconvergence exhibited by the recovered gradient. Concerning the computational advantages lead by a metric-based adapted mesh in terms of number of dofs, the discrepancy with a standard uniform refinement is more remarkable in this case with respect to what observed for Test case 1 (compare the right panels in Figures 7 and 13). The same accuracy on the solution is guaranteed by more than one order of dofs less, when the number of unknowns is sufficiently large (about 10^5 dofs).
Figure 14 (left) shows the histogram for the element size distribution predicted by Algorithm 2. By comparing this trend with the corresponding one in Figure 7, we observe that values are more concentrated around zero in this test case, the refined cells covering a larger part of the domain. Finally, the effectivity index in Figure 14 (right) exhibits a trend similar to the one in Figure 8 (right), although the stagnation trend is



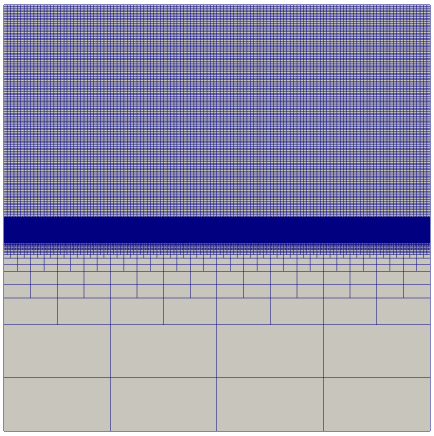
(a) Adapted mesh: $i = 1$.



(b) Adapted mesh: $i = 3$.

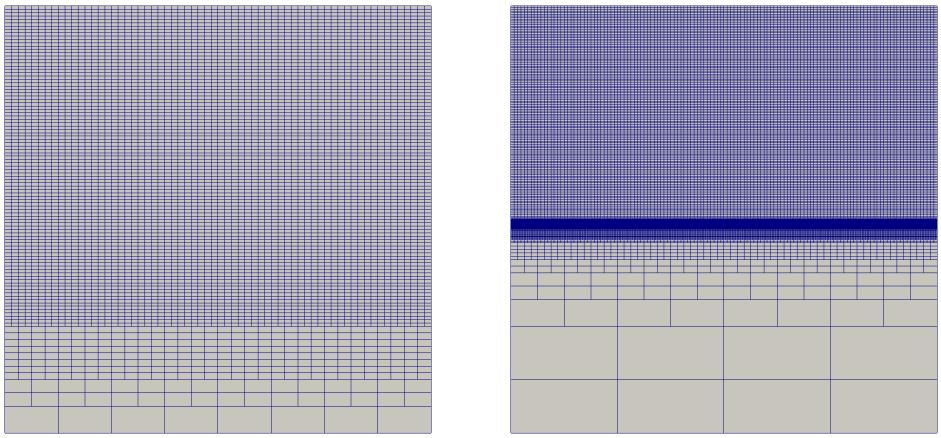


(c) Adapted mesh: $i = 5$.



(d) Adapted mesh: $i = 9$.

Figure 11: Test case 2 (rectilinear discontinuity). Marking-based adaptation.



(a) Adapted mesh: $i = 1$.

(b) Adapted mesh: $i = 3$.

Figure 12: Test case 2 (rectilinear discontinuity). Metric-based adaptation.

less evident at the last iterations.

5.2.2. Circular discontinuity

Discontinuities across not rectilinear layers constitute a challenging task for Cartesian grids. The adoption of non-conforming meshes turns out to be instrumental in such a direction, since these grids allow to confine the refinement of the elements where it is strictly necessary.

We assess the skills of the metric-based algorithm to deal with not straight discontinuities by considering a setting where the diffusion coefficient is discontinuous along a circular interface [20]. We solve a purely diffusive problem (1) in $\Omega = (0,1)^2$ after choosing

$$\varepsilon = \begin{cases} \varepsilon_G = 1 & \text{in } G \\ \varepsilon_S = 100 & \text{in } S, \end{cases} \tag{23}$$

where $G = \{(x-0.5)^2 + (y-0.5)^2 \geq R^2\}$ for $R = 0.25$, $S = \Omega \setminus G$, $\beta = 0$, $b = 0$, $f = 1$. The exact solution coincides with the function

$$u_{\text{ex}}(x,y) = \begin{cases} \dfrac{1}{8} - \dfrac{1}{4\varepsilon_G}\left((x-0.5)^2 + (y-0.5)^2\right) & \text{in } G \\ \dfrac{1}{8} - \dfrac{1}{4\varepsilon_S}\left((x-0.5)^2 + (y-0.5)^2\right) - \dfrac{R^2}{4}\left(1 - \dfrac{1}{\varepsilon_S}\right) & \text{in } S. \end{cases}$$

We impose the Dirichlet boundary data $u = u_{\text{ex}}$ on $\partial\Omega$ and a continuity condition at the interface γ between G and S on both the solution u and the associated normal flux $\varepsilon\nabla u$. Figure 15 provides a two- and three-dimensional view of the solution, which clearly highlights the significant variation of u_{ex} in G. Moreover, the penalization applied in S mimics the presence of a hole centered in the domain.

Algorithm 2 is run on a uniform initial mesh with 8 subdivisions along both the x- and y-direction, and by setting $\text{tol} = 10^{-10}$ and $i_{\max} = 10$. The procedure returns a mesh with a maximum level of refinement equal to 14, which is shown in Figure 16 (bottom-left) together with two intermediate adapted meshes (top panels) and a zoom-in on the interface (bottom-right panel). We recognize a refinement of the grid along the interface between areas S and G, consistently with the discontinuity of the diffusive coefficient. In addition, a massive uniform refinement is applied to the region G; this is due to the fact that estimator η tracks the

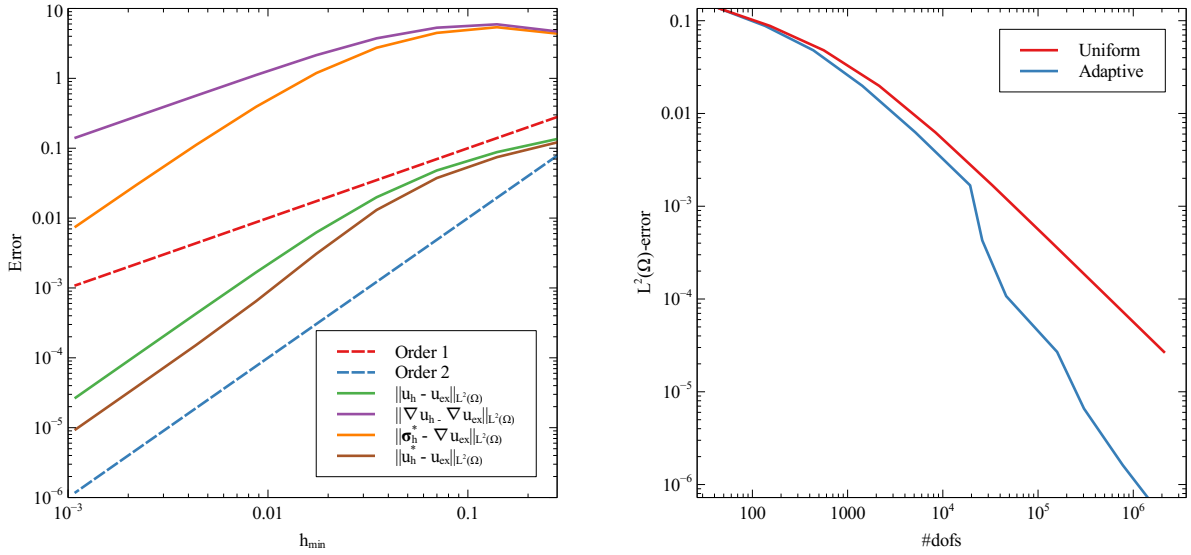


Figure 13: Test case 2 (rectilinear discontinuity). Convergence analysis of the error associated with the discrete and the recovered solution and gradient, as a function of the mesh size $h_{\min}$ (left); log-log plot of the norm $\|u - u_h\|_{L^2(\Omega)}$ as a function of the number of dofs on a uniform and on a metric-based adapted mesh (right).

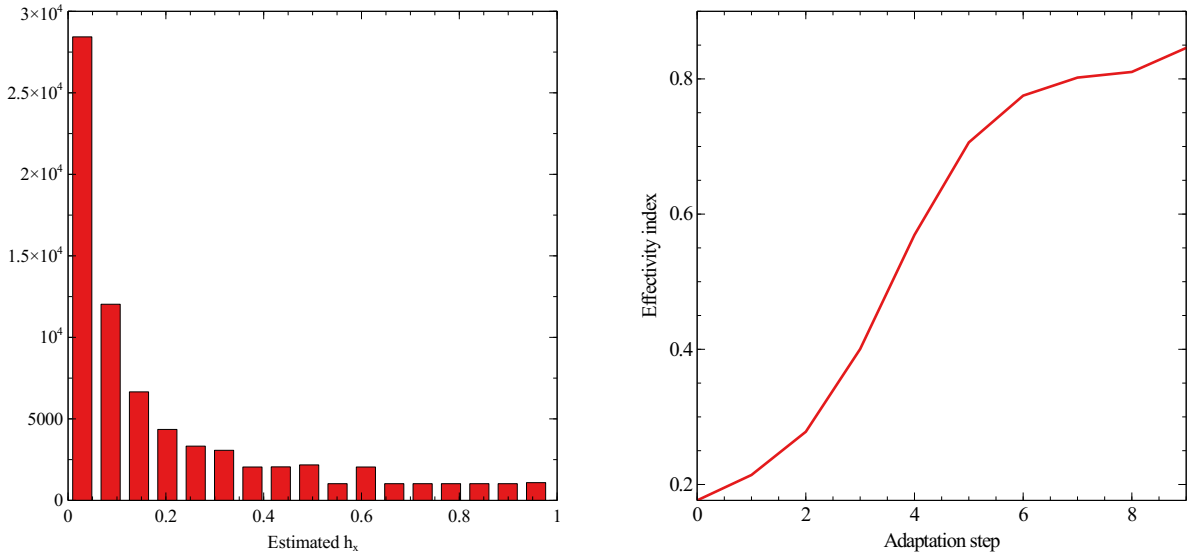
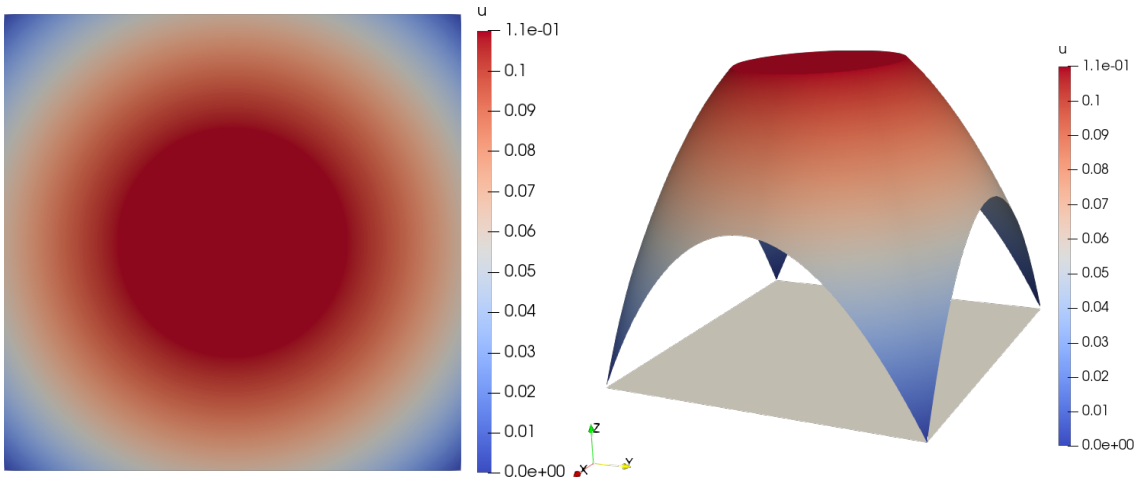


Figure 14: Test case 2 (rectilinear discontinuity). Histogram of the element size computed at the last iteration of the metric-based adaptive procedure (left); trend of the effectivity index as a function of the adaptation step (right).



(a) (b)

Figure 15: Test case 2 (circular discontinuity). 2D (left) and 3D (right) view of the solution.

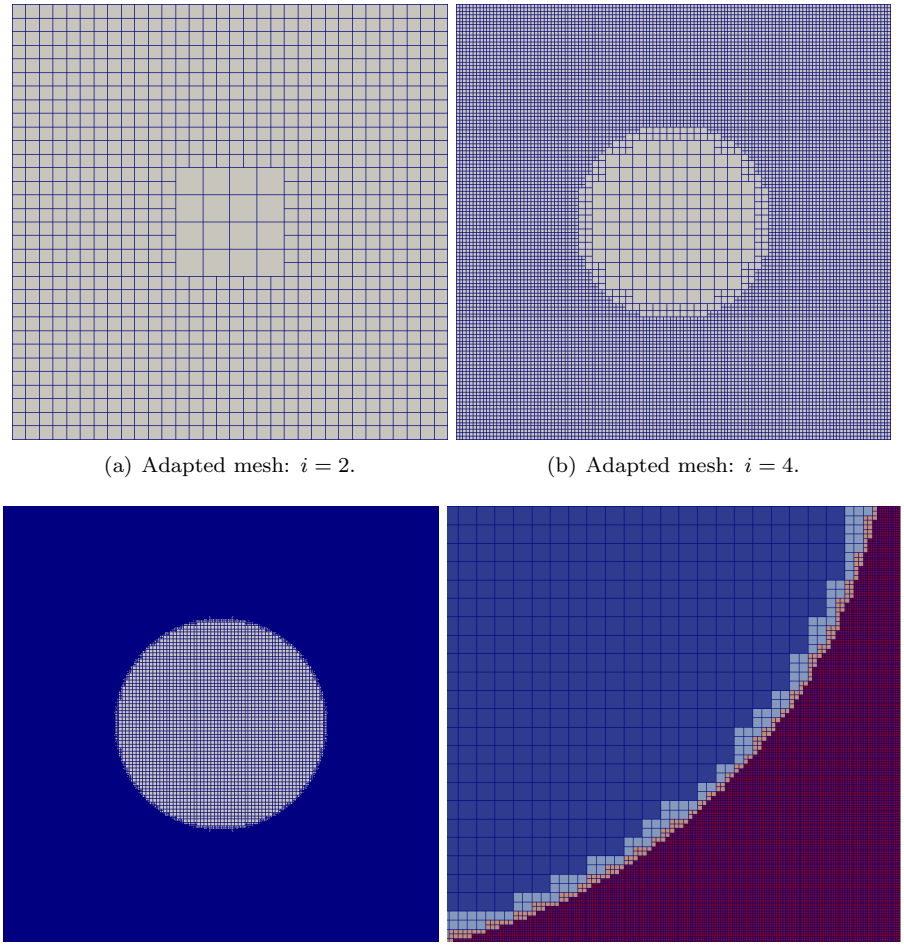
adapt. level	dofs	$h_{\min}$	$\|u-u_h\|_{L^2(\Omega)}$	$\|u-u_h\|_{L^2(\Omega)}$
7	81	0.176777	$2.82441 \cdot 10^{-3}$	$1.89502 \cdot 10^{-1}$
8	289	0.0883883	$6.82165 \cdot 10^{-4}$	$1.84833 \cdot 10^{-1}$
9	1033	0.0441942	$1.83441 \cdot 10^{-4}$	$2.81516 \cdot 10^{-1}$
10	3829	0.0220971	$5.33397 \cdot 10^{-5}$	$1.36216 \cdot 10^{-1}$
11	14245	0.0110485	$3.05214 \cdot 10^{-5}$	$2.01399 \cdot 10^{-2}$
12	55149	0.00552427	$1.16476 \cdot 10^{-5}$	$3.82096 \cdot 10^{-3}$
13	215321	0.00276214	$4.59463 \cdot 10^{-6}$	$1.02780 \cdot 10^{-3}$
14	853511	0.00138107	$2.98502 \cdot 10^{-6}$	–

Table 1: Test case 2 (circular discontinuity). Quantitative information about the metric-based algorithm (left panel); comparison between the metric-based approach proposed here and the geometric procedure proposed in [20] (fourth vs. fifth column) in terms of $L^2(\Omega)$-norm of the discretization error.

solution, so that also a variation of u_{ex} different from a steep gradient or a layer (see Fig. 15 (right)) can be responsible for a mesh refinement (the benefits lead by such a refinement are further commented below).

A more quantitative analysis of the metric-based approach is provided in Table 1, which collects the number of dofs (second column), the minimum diameter of the adapted mesh (third column) together with the $L^2(\Omega)$-norm of the discretization error (fourth column) at the last eight adaptation levels (first column). From a geometric viewpoint, the number of dofs increases by a factor varying between 3.5 and 4 throughout the iterative levels, while the minimum diameter halves iteration after iteration. Concerning the error, the reduction is more significant during the first levels before stagnating around 10^{-6}. We remark that the adaptive algorithm breaks before matching tolerance tol since the maximum number of iterations has been performed.

We have compared the output provided by Algorithm 2 with the adapted mesh proposed in [20] where the authors exploit a geometric indicator based on the standard concept of distance from the interface. The discrepancy between the two adaptive procedures is evident both from a qualitative and a quantitative viewpoint. The final adapted mesh in [20] confines the mesh refinement along the interface between G and S. The quantitative comparison has been carried out by setting a maximum level of refinement as well as an upper bound on the mesh cardinality. The last column in Table 1 is useful to cross-compare the two adaptive approaches in terms of accuracy, since gathering the value of the $L^2(\Omega)$-norm of the discretization error. It is evident that, for a fixed adaptation level, the extra-refinement in G allows to reach a higher accuracy with respect to an approach where only the discontinuity of ε is tracked, with, on average, a gain



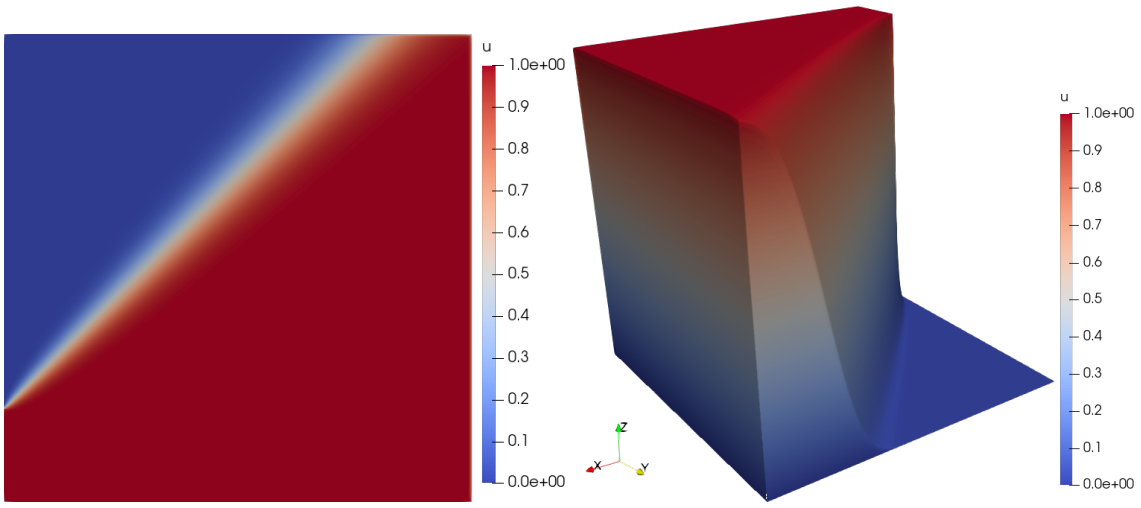
(a) Adapted mesh: $i = 2$.

(b) Adapted mesh: $i = 4$.

(c) Final mesh.

(d) Final mesh: detail of the interface.

Figure 16: Test case 2 (circular discontinuity). Metric-based adaptation.



(a) (b)

Figure 17: Test case 3. 2D (left) and 3D (right) view of the solution.

of two/three orders of accuracy.

5.3. Test case 3: an advection-diffusion problem with a diagonal advective field

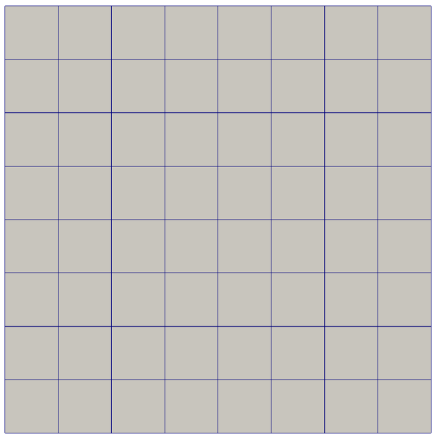
As last configuration, we consider the setting proposed in [45], where problem (1) is solved on $\Omega = (0,1)^2$ for $\varepsilon = 10^{-6}$, $\boldsymbol{\beta} = [\cos(\vartheta), \sin(\vartheta)]^T/\varepsilon$, $b = 0$ and $f = 0$. The problem is provided with Dirichlet data, namely we impose

$$u(x,y) = \begin{cases} 1 & \text{on } \Gamma = \{(0,y) \,:\, 0 < y \leq 0.2\} \cup \{(x,0) \,:\, 0 \leq x \leq 1\}, \\ 0 & \text{on } \partial\Omega \setminus \Gamma. \end{cases}$$

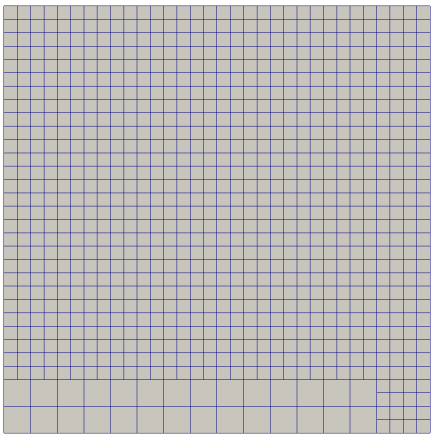
The solution identified by this choice of data exhibits an internal diagonal layer, with a slope equal to $\pi/4$, together with two very steep boundary layers, the former along the right side of Ω, the latter in correspondence with the right portion of the top side (see Fig. 17). We remark that the diagonal layer is characterized by a non-uniform thickness.

Figures 18 and 19 show the meshes generated at certain iterations of Algorithms 1 and 2, respectively. Both the adaptive procedures are run starting from an initial structured mesh characterized by 4 subdivisions along the two axes and by setting tol $= 10^{-6}$ and $i_{\max} = 10$. The internal and the boundary layers are detected by both the procedures, and vary the level of refinement consistently with the different thickness of the layers. In particular, the boundary layers which are extremely thin are very sharply captured, while it is evident the non-uniform as well as greater thickness of the diagonal layer. As for the previous test cases, the metric-based approach converges in a fewer number of iterations when compared with the marking-based method. However, in contrast to Test cases 1 and 2 where the metric-based approach detects layers more sharply, here the refined areas exhibit a similar amplitude for both the procedures, although the way adopted to cluster the elements is different in the two cases (more confined and finer around the diagonal boundary for Algorithm 1, wider and coarser for Algorithm 2). In terms of efficiency, the metric-based approach outperforms the marking-based algorithm, the number of dofs being considerably lower (150233 in Figure 19(b) versus 1371972 in Figure 18(d)).

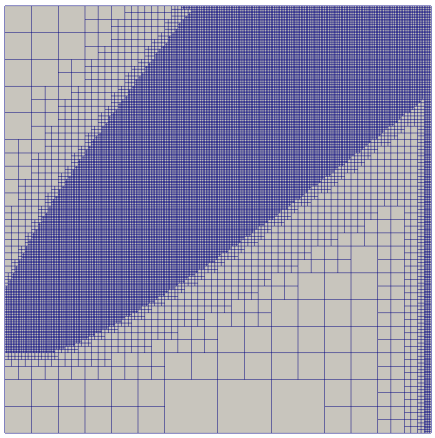
We exploit this test case to investigate the scalability of the metric-based adaptation procedure, by mimicking the analysis carried out in Figure 9. Figure 20 gathers the associated results. A cross-comparison between the two figures highlights a similar trend, although for this test case the bottleneck is represented by the adaptation procedure (which stagnates at about 32 cores). On the contrary, the gradient recovery exhibits a better speedup, approximately linear up to 512 cores. The deterioration of the mesh adaptation



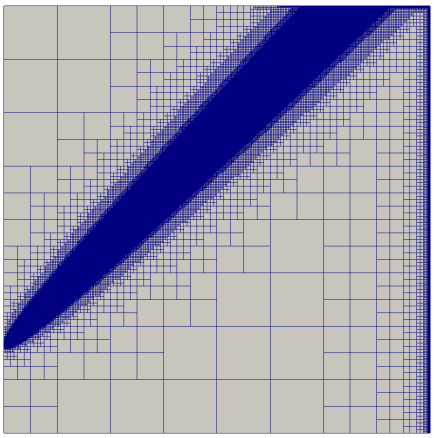
(a) Adapted mesh: $i = 1$.



(b) Adapted mesh: $i = 3$.

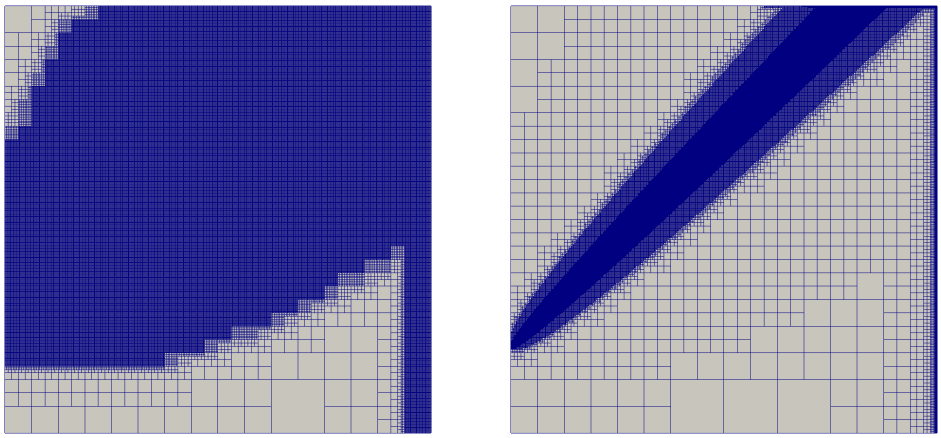


(c) Adapted mesh: $i = 6$.



(d) Adapted mesh: $i = 10$

Figure 18: Test case 3. Marking-based adaptation.



(a) Adapted mesh: $i = 2$.

(b) Adapted mesh: $i = 4$.

Figure 19: Test case 3. Metric-based adaptation.

speedup finds a justification in the continuous change of the computational load predicted by 22 for each cell as well as of the mesh partitioning among processes. Moreover, concerning the computational effort, running the whole algorithm requires a very contained time, ranging from half a second to few seconds.

Finally, for comparison purposes, we have applied to the configuration at hand also the adaptive procedure proposed in [46]. The authors compute the area-averaged L^2-norm of the discrete gradient over each mesh element,

$$\gamma_k = \frac{\|\nabla u_h\|_{L^2(\Omega^{(k)})}}{\left|\Omega^{(k)}\right|},$$

and then mark elements for refinement or coarsening according to condition

$$\gamma_k \geq \frac{c_1}{N_{\text{el}}} \sum_{k=1}^{N_{\text{el}}} \gamma_k \quad \text{or} \quad \gamma_k \leq \frac{c_2}{N_{\text{el}}} \sum_{k=1}^{N_{\text{el}}} \gamma_k,$$

respectively with c_1, c_2 user-defined parameters to be empirically tuned to confine the refinement/coarsening to the most problematic squares. Starting from the same initial mesh adopted in Figures 18-19, after 10 iterations we obtain the mesh in Fig. 21 which shows the final grid together with three preliminary adapted meshes. In particular, the final tessellation is fully comparable with the one yielded by the marking-based algorithm in Fig. 18, the main features of the solution being identified with a similar refinement pattern and after the same number (10) of iterations. The discrepancy between the two approaches is in terms of number of dofs, since having 1371972 nodes in Figure 18(d) to be compared with 1628325 nodes of Figure 21(d).

6. Conclusions and perspectives

This work formalizes a new mesh adaptation strategy for Cartesian quadtree meshes. The adaptation process is driven by the metric associated with a recovery based estimator for the $L^2(\Omega)$-norm of the dicretization error. Suitable finite difference formulas are employed to yield a higher-order approximation for the solution gradient to be, successively, employed in order to define a recovered solution. Finally, the discrepancy between the numerical and the recovered solution is used to settle the error estimator behind the mesh adaptation process.

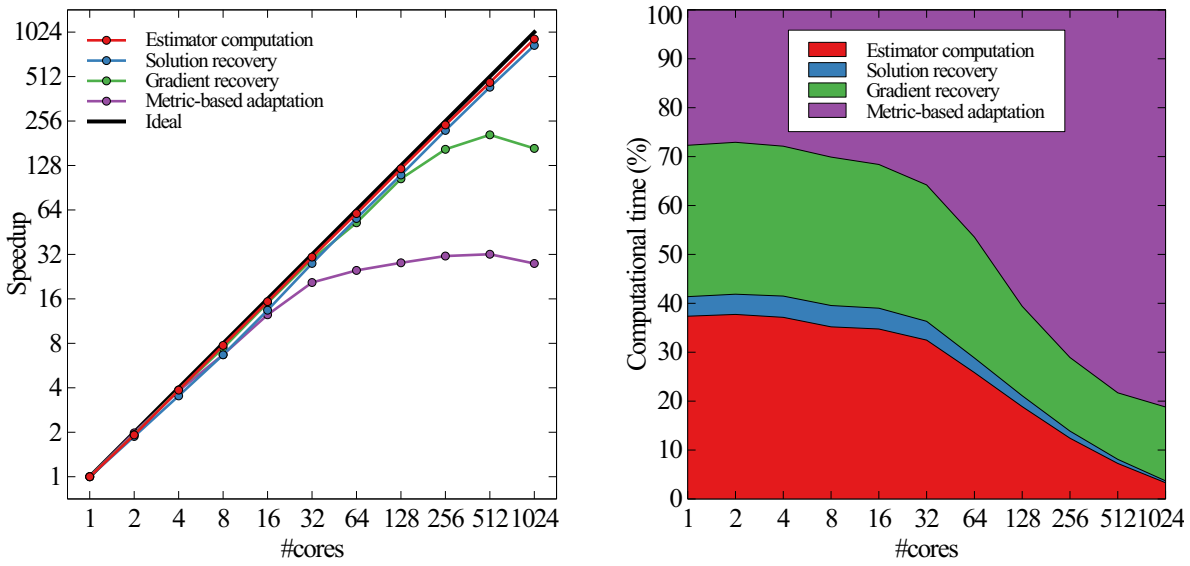


Figure 20: Test case 3. Scalabilty analysis of the different phases of the metric-based adaptation procedure: parallel speedup (left) and stacked barplot of the time percentage associated with the different phases of the algorithm (right) as a function of the number of parallel processes.

The massive numerical verification in Section 5 confirms the reliability of the proposed algorithm when applied to advection-diffusion-reaction problems, also in the presence of problem discontinuities not necessarily aligned with the Cartesian directions. Moreover, we have assessed the higher efficiency of the proposed metric-based approach when compared with a well-established estimate–mark–refine method and with other mesh adaptation strategies available in the literature, since guaranteeing, in general, a reduction of the number of estimation–adaptation steps and of dofs. We have also assessed the scalability of the metric-based algorithm. We recognize a remarkable performance in terms of parallel speedup, which is approximately linear for most of the phases involved in the adaptation process, as well as of computational time, which remains of the order of few seconds also in the worst cases.

The estimator η turns out to be fully general in terms of differential problem to be solved, dimensionality of the problem, and adopted discretization. As a consequence, future developments could involve the extension of the proposed algorithm to more complex three-dimensional problems, such as large scale scenarios or time-dependent applications. As a more challenging task, we aim to generalize the a posteriori setting here settled to an anisotropic context, starting from the papers already available in the literature which provide an anisotropic counterpart of the Zienkiewicz-Zhu error estimator [47, 48, 49, 32].

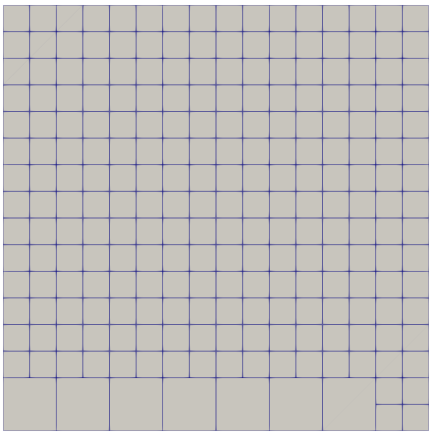
Acknowledgements

This work was supported by Accordo Quadro ASI-POLIMI “Attività di Ricerca e Innovazione” n. 2018-5-HH.0, collaboration agreement between the Italian Space Agency and Politecnico di Milano. Simona Perotto thanks the PRIN research grant n.20204LN5N5 Advanced Polyhedral Discretisations of Heterogeneous PDEs for Multiphysics Problems and the INdAM-GNCS 2022 Project Metodi di riduzione computazionale per le scienze applicate: focus su sistemi complessi.

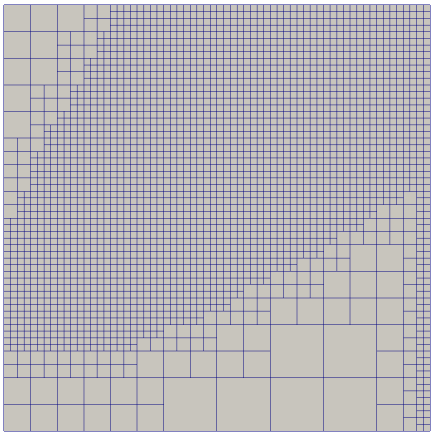
References

[1] H. Samet, Using quadtrees to represent spatial data, in: H. Freeman, G. G. Pieroni (Eds.), Computer Architectures for Spatially Distributed Data, Springer Berlin Heidelberg, 1985, pp. 229–247.

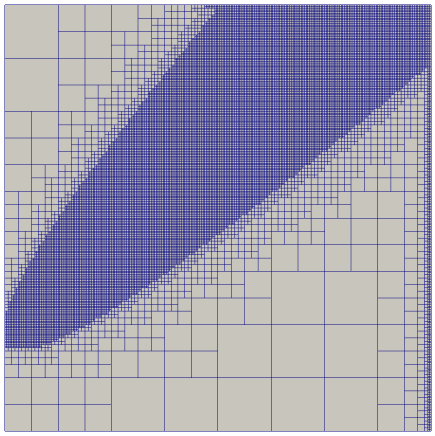
[2] G. M. Morton, A computer oriented geodetic data base and a new technique in file sequencing, Tech. rep., International Business Machines Company, New York (1966).



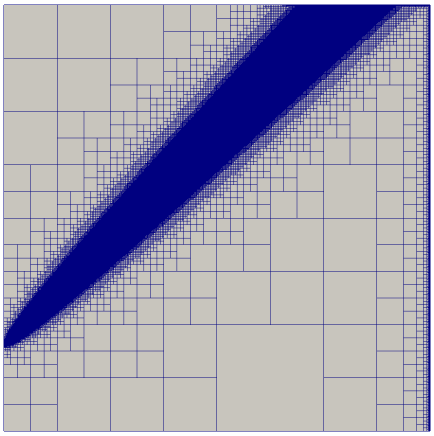
(a) Adapted mesh: $i = 2$.



(b) Adapted mesh: $i = 4$.



(c) Adapted mesh: $i = 6$.



(d) Adapted mesh: $i = 10$.

Figure 21: Test case 3. Adaptation based on [46].

[3] R. A. Finkel, J. L. Bentley, Quad trees a data structure for retrieval on composite keys, Acta Informatica 4 (1) (1974) 1–9.
[4] H. Samet, The quadtree and related hierarchical data structures, ACM Computing Surveys (CSUR) 16 (2) (1984) 187–260.
[5] J. J. Hasbestan, I. Senocak, A parallel adaptive mesh refinement software for complex geometry flow simulations, in: 23rd AIAA Computational Fluid Dynamics Conference, 2017, p. 3301.
[6] J. D. Towers, A source term method for poisson problems on irregular domains, Journal of Computational Physics 361 (2018) 424–441.
[7] A. Lee, W. Geng, S. Zhao, Regularization methods for the poisson-boltzmann equation: Comparison and accuracy recovery, Journal of Computational Physics 426 (2021) 109958.
[8] E. Burman, S. Claus, P. Hansbo, M. G. Larson, A. Massing, Cutfem: Discretizing geometry and partial differential equations, International Journal for Numerical Methods in Engineering 104 (7) (2015) 472–501.
[9] C. Gürkan, S. Sticko, A. Massing, Stabilized cut discontinuous galerkin methods for advection-reaction problems, SIAM Journal on Scientific Computing 42 (5) (2020) A2620–A2654.
[10] C. Burstedde, L. C. Wilcox, O. Ghattas, `p4est`: Scalable algorithms for parallel adaptive mesh refinement on forests of octrees, SIAM Journal on Scientific Computing 33 (3) (2011) 1103–1133.
[11] T. Isaac, C. Burstedde, L. C. Wilcox, O. Ghattas, Recursive algorithms for distributed forests of octrees, SIAM Journal on Scientific Computing 37 (5) (2015) C497–C531. `doi:10.1137/140970963`.
[12] C. Burstedde, Parallel tree algorithms for AMR and non-standard data access, ACM Transactions on Mathematical Software 46 (32) (2020) 1–31. `doi:10.1145/3401990`.
[13] M. A. de Souza Lourenço, E. L. M. Padilla, An octree structured finite volume based solver, Applied Mathematics and Computation 365 (2020) 124721.
[14] [link].
URL `http://www.optimad.it/products/bitpit/`
[15] [link].
URL `http://p4est.github.io/`
[16] [link].
URL `http://basilisk.fr/`
[17] V. Laurmaa, M. Picasso, G. Steiner, An octree-based adaptive semi-lagrangian vof approach for simulating the displacement of free surfaces, Computers & Fluids 131 (2016) 190 – 204.
[18] Q. Liang, A. G. Borthwick, Adaptive quadtree simulation of shallow flows with wet–dry fronts over complex topography, Computers & Fluids 38 (2) (2009) 221 – 234.
[19] R.-C. Chen, J.-L. Liu, Monotone iterative methods for the adaptive finite element solution of semiconductor equations, Journal of Computational and Applied Mathematics 159 (2) (2003) 341 – 364.
[20] A. Raeli, M. Bergmann, A. Iollo, A finite-difference method for the variable coefficient Poisson equation on hierarchical Cartesian meshes, Journal of Computational Physics 355 (2018) 59–77.
[21] Parallel simulations for fast-moving landslides: space-time mesh adaptation and sharp tracking of the wetting front, International Journal For Numerical Methods in Fluids (2023).
[22] A. Coco, G. Russo, Second order finite-difference ghost-point multigrid methods for elliptic problems with discontinuous coefficients on an arbitrary interface, Journal of Computational Physics 361 (2018) 299–330.
[23] O. C. Zienkiewicz, J. Z. Zhu, A simple error estimator and adaptive procedure for practical engineerng analysis, International Journal for Numerical Methods in Engineering 24 (2) (1987) 337–357.
[24] O. C. Zienkiewicz, J. Z. Zhu, The superconvergent patch recovery and a posteriori error estimates. I: The recovery technique, International Journal for Numerical Methods in Engineering 33 (1992) 1331–1364.
[25] O. C. Zienkiewicz, J. Z. Zhu, The superconvergent patch recovery and a posteriori error estimates. II: Error estimates and adaptivity, International Journal for Numerical Methods in Engineering 33 (1992) 1365–1382.
[26] N. Yan, A posteriori error estimators of gradient recovery type for elliptic obstacle problems, Advances in Computational Mathematics 15 (1-4) (2001) 333–362.
[27] G. M. Porta, S. Perotto, F. Ballio, Anisotropic mesh adaptation driven by a recovery-based error estimator for shallow water flow modeling, International Journal for Numerical Methods in Fluids 70 (3) (2012) 269–299.
[28] L. Mu, R. Jari, A recovery-based error estimate for nonconforming finite volume methods of interface problems, Applied Mathematics and Computation 220 (2013) 63–74.
[29] S. Micheletti, S. Perotto, L. Soli, Topology optimization driven by anisotropic mesh adaptation: towards a free-form design, Computers & Structures 214 (2019) 60–72.
[30] X. D. Li, N.-E. Wiberg, A posteriori error estimate by element patch post-processing, adaptive analysis in energy and L_2 norms, Computers & Structures 53 (4) (1994) 907–919.
[31] G. Maisano, S. Micheletti, S. Perotto, C. Bottasso, On some new recovery-based a posteriori error estimators, Computer Methods in Applied Mechanics and Engineering 195 (37) (2006) 4794 – 4815.
[32] N. Ferro, S. Perotto, A. Cangiani, An anisotropic recovery-based error estimator for adaptive discontinuous Galerkin methods, Journal of Scientific Computing 90 (1) (2022) Paper No. 45, 24.
[33] R. Verfurth, A Review of A Posteriori Error Estimation and Adaptive Mesh-Refinement Techniques, Advances in Numerical Mathematics, Vieweg-Teubner Verlag Wiesbaden, 1996.
[34] R. E. Bank, W. Coughran Jr, L. C. Cowsar, The finite volume Scharfetter-Gummel method for steady convection diffusion equations, Computing and Visualization in Science 1 (3) (1998) 123–136.
[35] D. N. De G. Allen, R. V. Southwell, Relaxation methods applied to determine the motion, in two dimensions, of a viscous fluid past a fixed cylinder, The Quarterly Journal of Mechanics and Applied Mathematics 8 (2) (1955) 129–145.

[36] M. A. Zlámal, Finite element solution of the fundamental equations of semiconductor devices. I, Mathematics of Computation 46 (173) (1986) 27–43.
[37] P. A. Markowich, M. A. Zlámal, Inverse-average-type finite element discretizations of self-adjoint second-order elliptic problems, Mathematics of Computation 51 (184) (1988) 431–449.
[38] F. Brezzi, L. D. Marini, P. Pietra, Two-dimensional exponential fitting and applications to drift-diffusion models, SIAM Journal on Numerical Analysis 26 (6) (1989) 1342–1355.
[39] W. Bangerth, C. Burstedde, T. Heister, M. Kronbichler, Algorithms and data structures for massively parallel generic adaptive finite element codes, ACM Transactions on Mathematical Software 38 (2) (2012) 1–28.
[40] T. Isaac, C. Burstedde, O. Ghattas, Low-cost parallel algorithms for 2: 1 octree balance, in: 2012 IEEE 26th International Parallel and Distributed Processing Symposium, 2012, pp. 426–437.
[41] W. Bangerth, O. Kayser-Herold, Data structures and requirements for hp finite element software, ACM Transactions on Mathematical Software 36 (1) (2009) 4/1–4/31.
[42] J.-C. Nédélec, Mixed finite elements in $\mathbb{R}^3$, Numerische Mathematik 35 (3) (1980) 315–341.
[43] P. J. Frey, P.-L. George, Mesh Generation: Application to Finite Elements, 2nd Edition, ISTE, London; John Wiley & Sons, Inc., Hoboken, NJ, 2008.
[44] C. de Falco, E. O'Riordan, Interior layers in a reaction-diffusion equation with a discontinuous diffusion coefficient, Int. J. Numer. Anal. Model. 7 (3) (2010) 444–461.
[45] A. Brooks, T. Hughes, Streamline upwind/Petrov-Galerkin formulations for convection dominated flows with particular emphasis on the incompressible navier-stokes equations, Computer Methods in Applied Mechanics and Engineering 32 (1982) 199–259.
[46] D. Schillinger, L. Dede', M. A. Scott, J. A. Evans, M. J. Borden, E. Rank, T. J. Hughes, An isogeometric design-through-analysis methodology based on adaptive hierarchical refinement of NURBS, immersed boundary methods, and T-spline CAD surfaces, Computer Methods in Applied Mechanics and Engineering 249 (2012) 116–150.
[47] S. Micheletti, S. Perotto, Anisotropic adaptation via a Zienkiewicz–Zhu error estimator for 2D elliptic problems, in: G. Kreiss, P. Lötstedt, M. N. A. Målqvist (Eds.), Numerical Mathematics and Advanced Applications, Vol. 645-653, Springer-Verlag Berlin Heidelberg, 2010.
[48] P. E. Farrell, S. Micheletti, S. Perotto, An anisotropic Zienkiewicz-Zhu-type error estimator for 3D applications, International Journal for Numerical Methods in Engineering 85 (6) (2011) 671–692.
[49] S. Micheletti, S. Perotto, P. E. Farrell, A recovery-based error estimator for anisotropic mesh adaptation in CFD, Boletin de la Sociedad Espanola de Matematica Aplicada SeMA (50) (2010) 115–137.

MOX Technical Reports, last issues

Dipartimento di Matematica
Politecnico di Milano, Via Bonardi 9 - 20133 Milano (Italy)

01/2023 Zingaro, A.; Bucelli, M.; Piersanti, R.; Regazzoni, F.; Dede', L.; Quarteroni, A.
An electromechanics-driven fluid dynamics model for the simulation of the whole human heart

02/2023 Boon, W. M.; Fumagalli, A.; Scotti, A.
Mixed and multipoint finite element methods for rotation-based poroelasticity

82/2022 Ciaramella, G.; Gander, M.; Van Criekingen, S.; Vanzan, T.
A PETSc Parallel Implementation of Substructured One- and Two-level Schwarz Methods

81/2022 Bonizzoni, F.; Hauck, M.; Peterseim, D.
A reduced basis super-localized orthogonal decomposition for reaction-convection-diffusion problems

84/2022 Ciaramella, G.; Gambarini, M.; Miglio, E.
A preconditioner for free-surface hydrodynamics BEM

85/2022 Lurani Cernuschi , A.; Masci, C.; Corso, F.; Muccini, C.; Ceccarelli, D.; San Raffaele Hospital Galli, L.; Ieva, F.; Paganoni, A.M.; Castagna, A.
A neural network approach to survival analysis for modelling time to cardiovascular diseases in HIV patients with longitudinal observations

83/2022 Ciaramella, G.; Gander, M.; Mazzieri, I.
Unmapped tent pitching schemes by waveform relaxation

80/2022 Balduzzi, G.; Bonizzoni, F.; Tamellini, L.
Uncertainty quantification in timber-like beams using sparse grids: theory and examples with off-the-shelf software utilization

78/2022 Bucelli, M.; Gabriel, M. G.; Gigante, G.; Quarteroni, A.; Vergara, C.
A stable loosely-coupled scheme for cardiac electro-fluid-structure interaction

79/2022 Antonietti, P. F.; Farenga, N.; Manuzzi, E.; Martinelli, G.; Saverio, L.
Agglomeration of Polygonal Grids using Graph Neural Networks with applications to Multigrid solvers