Journal of Computational and Applied Mathematics 416 (2022) 114442

Contents lists available at ScienceDirect

Journal of Computational and Applied Mathematics

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

An adaptive rectangular mesh administration and refinement technique with application in cancer invasion models

Niklas Kolbe [a], Nikolaos Sfakianakis [b,*]

[a] Institute of Geometry and Practical Mathematics, RWTH Aachen University, Germany
[b] School of Mathematics and Statistics, University of St Andrews, UK

ARTICLE INFO

Article history:
Received 7 July 2021
Received in revised form 25 April 2022

Keywords:
Mesh administration
Adaptive mesh refinement
Finite volume method
h-refinement
Cancer invasion

ABSTRACT

We present an administration technique for the bookkeeping of adaptive mesh refinement on (hyper-)rectangular meshes. Our technique is a unified approach for h-refinement on 1-, 2- and 3D domains, which is easy to use and avoids traversing the connectivity graph of the ancestry of mesh cells. Due to the employed rectangular mesh structure, the identification of the siblings and the neighbouring cells is greatly simplified. The administration technique is particularly designed for smooth meshes, where the smoothness is dynamically used in the matrix operations. It has a small memory footprint that makes it affordable for a wide range of mesh resolutions over a large class of problems. We present three applications of this technique, one of which addresses h-refinement and its benefits in a 2D tumour growth and invasion problem.

1. Introduction

Time and again *adaptive mesh refinement* (AMR) has been employed to improve the accuracy of numerical methods and to alleviate their computational burden. Typical fields of application of AMR include engineering, astrophysics, and fluid dynamics, where the associated techniques have become a vital component of the overall numerical investigation, see, e.g., [1–8]. In mathematical biology however, they have been scarcely applied yet. Some examples can be found in [9–13].

AMR, has become a standard for which particular and elaborate numerical methods have been developed. Still the mesh administration remains a complex process, which most scientific computing efforts try to avoid. The aim of the current paper is to propose a "do-it-yourself" recipe of AMR and mesh administration strategy that emphasizes its simplicity of implementation and use, while allowing for the handling of situations that cover most of the spectrum of scientific computing.

In this perspective, one of the aims of our work is to apply AMR to models that describe the *first stage in cancer metastasis* – and one of the *hallmarks of cancer* – the invasion of the *extracellular matrix* (ECM) [14]. The models we consider in this work are typically *advection–reaction–diffusion* (ARD) systems where the involved quantities are represented by their macroscopic densities. The dynamics that their solutions exhibit can be complex and their numerical treatment is challenging, and hence, AMR is seen as an important tool. A 1D application of AMR – in particular *h-refinement* – in a *Finite Volume* (FV) method for such a system has been studied in our previous work [13]. There, we demonstrated a significant improvement in accuracy and efficiency when AMR was employed. For 2D (or higher dimensional) cases however, a series of additional numerical difficulties arise. For example, the nature of these particular cancer invasion models promotes the

* Corresponding author.
E-mail addresses: kolbe@igpm.rwth-aachen.de (N. Kolbe), n.sfakianakis@st-andrews.ac.uk (N. Sfakianakis).

https://doi.org/10.1016/j.cam.2022.114442

use of FV methods over meshes with *rectangular cells*, which when combined with h-refinement techniques lead to *hanging nodes*. This complicates significantly the computation of the advection and, most notably, of the diffusion components of the system. In the literature, 2D *convection–diffusion* problems have been previously solved by FV methods augmented by AMR techniques, see e.g. [15] and the references therein.

Many technical issues related to the implementation of the AMR techniques arise as well. Some are typical: traversing the *connectivity graph* between *children*, *parents*, and *ancestor* mesh cells, as well as the identification of *sibling* and *neighbour cells* needed for the computation of the numerical fluxes and for the refinement/coarsening procedures. Other difficulties are less common e.g. particularities of the model that might pose additional conditions on the structure and the handling of the mesh. To mention but a few: the fast dynamics of the solutions and the diffusion of the model manifest themselves in the "smoothness" of the mesh, we refer e.g. to [16] for a thorough discussion on the topology of mesh generation. Furthermore, the large number of system components and the richness of phenomena to be resolved call for different (sometimes multiple) refinement and coarsening criteria; the possibility of a blow-up, which is inherent in this type of systems, see e.g. [17,18], requires a dynamic adaptation of the highest refinement level.

To address the above numerical and technical issues, we develop our own AMR and mesh administration techniques. There are several reasons for that: first, we aim for simplicity, flexibility, and portability of our algorithms. Secondly, we want complete control of all the stages of the numerical treatment of the models. This includes various components of the implementation: the numerical methods for the solution of the *partial differential equation* (PDE) systems, the AMR techniques for the refinement/coarsening process, as well as the grid administration and *bookkeeping algorithms* for the handling of the data on the mesh.

The overall effort is extensive; we have previously addressed the development of the numerical methods to solve the corresponding models, and the 1D h-refinement technique [13,19,20]. In the current paper, we focus primarily on the grid administration/handling, and secondarily on the presentation of a concrete application of h-refinement to a 2D *cancer invasion* system. The *h-refinement* technique that we employ, makes use of cell bisection in 2, 4, or 8 equivalent rectangle cells in 1-, 2-, or 3D, respectively. This approach has been widely used in the application of AMR methods, see e.g. [6,8].

Regarding the structure of the mesh data and their bookkeeping, there are several requirements that should be fulfilled by a modern mesh administration technique. We mention here only the following, and refer to [21–25] for more details: (a) *generality* with respect to the dimensions of the problems and the shape of the domain, (b) *efficiency* in the access times of the stored information during and after the reconstruction of the mesh, (c) at least affordable, if not minimal, *memory costs*, (d) *simplicity, portability*, and *transparency* of the implementation, and (e) *extensibility* to parallelizations.

The most common practice is to use *pointer-based mesh data structures*. In this approach, the basic objects considered – e.g. vertices – are explicitly represented by their physical coordinates. The edges/faces/cells are defined by reference to the vertices using a cascade of pointers/handles, see e.g. [23,26,27]. In the particular case of a 2D triangulation, these references are to the three vertices as well as the three neighbouring triangles of each triangle are stored. This is an intuitive and relatively easy to implement and use technique, but it has a significant memory footprint, and poses additional computational burden at every step of the method, cf. [22]. When a discretization cell is refined the references pointing to this cell as well as the other cells it points to, need to be adjusted accordingly — a process that can be complex especially in 3D volumes. For further information on the implementation and the applications of this technique, we refer to [25,28–32] and the references therein. Alternative approaches have been devised with main aim to increase the efficiency of the methods, see e.g. the *half-edge* and *half-facet array-based mesh data structure* implementations, see e.g. [23–25]. These techniques have been used primarily for the computer graphics representation of surface objects in 3D, see e.g. [33–35].

The approach we propose in the current paper is problem-specific and focuses on rectangular meshes in 1-, 2- and 3D. Unlike the pointer-based method, we store the full data structure of the mesh tree in the form of a matrix and refer to different mesh cells via the corresponding lines of the matrix. Every cell points to its children and parent cells. We do not store the physical coordinates, we instead store the indices of the mesh cells, i.e. the refinement level and an *intra-level* identifier of every cell instead of the physical coordinates of its vertices. In more detail, for given minimum and maximum refinement levels, we pre-compile a matrix in which we store, for each cell of the discretization tree, information on its parent and children cells. This along with an enforced *grid regularity*, allows for the effortless and efficient refinement, coarsening, and neighbour identification processes. The memory footprint of the method we propose is reasonable, even for a large cascade of refinement levels in 2- and 3D experiments without using memory compression.

When comparing with the software library `p4est` [36], our approach exhibits both commonalities and differences. On the one hand, both approaches assume a tree based cell-hierarchy that branches-off from a root cell and reaches the leaf cells. Each cell points to its parent and its children cells. On the other hand, in our approach we characterize the computational cells in the mesh tree by a single integer (in one-, two-, and three-D), where in `p4est` every 3-dimensional mesh cell is characterized by 3 (real number) coordinates x, y, z. Moreover, our representation refers to the centre of the computational cell, whereas in `p4est` the lower left corner is represented. This introduces some ambiguity as no two different cells in a binary-, quad-, or octree share the same centre but they might share the same lower left corner (as, e.g., parent and some child cells). For more information on the structure and functionality of `p4est` we refer to [36,37]. We also refer to its employment in the `deal.II` computational framework [38,39]. The comparison of our proposed technique with `alugrid`, [40] is less straight forward as the information stored in `alugrid` is more diverse and includes vertices, edges, faces, and more. Still, we refer the interested reader to [40,41] for a thorough discussion of this method and its employment within the `DUNE` computational framework [42,43].

Overall, in this work we propose a mesh data structure, which is easy to understand, implement, and use, and it can handle mesh resolutions that are sufficient for a wide range of academic investigations in 1-, 2- and 3D. We exhibit this flexibility by presenting three different applications/experiments while giving particular emphasis on a cancer invasion problem.

The rest of the paper is structured as follows: in Section 2 we present the mesh administration technique, address basic operations, such as the refinement/coarsening and the identification of neighbours and siblings, and discuss the memory usage and computational costs. In Section 3 we present three numerical experiments: a generic one exhibiting the properties of the mesh administration technique, as well as applications in a Euler system and a cancer invasion model. We discuss the basic ingredients of the numerical methods and the AMR technique we use.

2. Mesh data structure and handling

A significant burden in the development of AMR techniques is the *administration* of the discretization meshes, in particular when dealing with time dependent 2- or 3D problems. We propose in this section a particularly easy procedure to *represent*, *store*, *handle*, and in general *bookkeep dyadic rectangular meshes* in a unified way for 1-, 2-, or even 3D domains.

Main characteristic of the proposed mesh administration technique is that we encode the information of the dyadic tree inside an easy-to-use matrix. We do it in such a way that we access with ease the *parent*, *children*, and most notably the *siblings* and *neighbouring* cells of any given cell. In effect we simplify greatly the *refinement* and, most importantly, the *coarsening* steps of the mesh.

This part of the paper is structured as follows: in Section 2.1 we present the main notations and definitions, in Section 2.2 we describe the way we store the information of the computational cells and the dyadic meshes, and comment on the memory usage of the method. In Section 2.3 we present the operations needed to identify the sibling and neighbour cells, in Section 2.4 we elaborate on the refinement and coarsening procedures, and in Section 2.5 we discuss the handling of data on the mesh.

2.1. Basic definitions and notations

The proposed technique can be employed on general hyperrectangles in 1-, 2-, or 3D domains. For the sake of simplicity we will restrict the presentation to the discretization of the domain $\Omega = [0, 1]^d$ for $d = 1, 2$.

We denote by G_l^d, $l \in \mathbb{N}$ a *uniform partition* of Ω with cardinality

$$|G_l^d| = 2^{ld}. \tag{1}$$

Here, l is the *refinement level* of G_l^d assumed to be bounded by $l_{\min} \leq l \leq l_{\max}$ with $l_{\min}, l_{\max} \in \mathbb{N}$. The elements of G_l^d are called *cells* and they are either intervals in $d = 1$, or squares in $d = 2$.

For every cell $C \in \bigcup_{l=l_{\min}}^{l_{\max}} G_l^d$, we denote its refinement level, i.e. the index l for which $C \in G_l^d$, by $L(C)$, its centre by $M^p(C) \in \Omega$, and the occupied physical domain by $D(C) \subset \Omega$. Accordingly we can write,

$$M^p(C) \in \left\{ \sum_{i=1}^{d} \frac{2k_i - 1}{2^{L(C)+1}} \mathbf{e_i}, \quad 1 \leq k_i \leq 2^{L(C)}, \quad k_i \in \mathbb{N} \right\}, \tag{2a}$$

$$D(C) = \left\{ M^p(C) + \sum_{i=1}^{d} \frac{\lambda_i}{2^{L(C)}} \mathbf{e_i}, \quad -\frac{1}{2} < \lambda_i < \frac{1}{2}, \quad \lambda_i \in \mathbb{R} \right\}. \tag{2b}$$

where $\mathbf{e_i}$, $i = 1, 2$ represents the unit vector of the corresponding axis.

A cell $\tilde{C}$ is termed *child cell* of the cell $C \in G_l^d$ if $\tilde{C} \in G_{l+1}^d$ and $M^p(\tilde{C}) \in D(C)$. Equivalently, the cell C is called the *parent cell* of $\tilde{C}$. Every parent cell has several (2,4, or 8, depending on the dimension d) children cells, and every child cell has a single parent cell. Children cells of the same parent cell are called *siblings*. Geometrically, the children cells are obtained by *bisection* of the parent cell.

The centre of a parent cell resides on the boundary of all its children cells, and the boundary of a parent cell is partly shared with the boundaries of its children cells. The cell $\tilde{C} \in G_{\tilde{l}}^d$ is a *descendant* of the cell $C \in G_l^d$, equivalent to saying that C is an *ancestor* of $\tilde{C}$, if there is a cascade of parent–children relations between C and $\tilde{C}$.

Two cells that share a (part of their) boundary of co-dimension 1 but no part of their physical domain are called *neighbours*. In 2D, this definition excludes cells that share a single point of their boundary. The common boundary between two neighbour cells is their *interface*.

For the rest of this work, we consider meshes that are subsets of $\bigcup_{l=l_{\min}}^{l_{\max}} G_l^d$, for particular $l_{\min}, l_{\max} \in \mathbb{N}$. We also expect them to obey a particular smoothness relation discussed in the following definition.

Definition 1 (*Regular Structured Mesh*)**.** A d-dimensional ($d = 1, 2$) mesh G is called *Regular Structured Mesh* (RSM) of minimum and maximum refinement levels $l_{\min}, l_{\max} \in \mathbb{N}$, and of *mesh regularity* $m_r \in \mathbb{N}$ if

$$G \subset \bigcup_{l=l_{\min}}^{l_{\max}} G_l^d, \tag{3}$$

Table 1
Mesh cells on uniform dyadic meshes that discretize [0,1] with refinement level 0 through $l_{\max}$. The refinement level l and the intra-level index k suffice for their complete characterization.

Level	Number of cells	Centres	Cell size
0	1	$\left\{\frac{1}{2}\right\}$	1
1	2	$\left\{\frac{1}{4}, \frac{3}{4}\right\}$	$\frac{1}{2}$
2	4	$\left\{\frac{1}{8}, \frac{3}{8}, \frac{5}{8}, \frac{7}{8}\right\}$	$\frac{1}{4}$
$\vdots$	$\vdots$	$\vdots$	$\vdots$
$l_{\max}$	$2^{l_{\max}}$	$\left\{\frac{2k-1}{2^{l_{\max}+1}},\ k = 1, \ldots, 2^{l_{\max}}\right\}$	$\frac{1}{2^{l_{\max}}}$

and if

1. For all $\mathbf{x} \in [0, 1]^d$ either $\exists!\, C \in G$ such that $\mathbf{x} \in D(C)$ or $\exists C \in G$ such that $\mathbf{x} \in \partial D(C)$.
2. For all neighbour cells $C_j, C_k \in G$, the *mesh regularity condition* holds:

$$|L(C_j) - L(C_k)| \leq m_r. \tag{4}$$

We can now introduce the basic operations of *refinement* and *coarsening*: for $G \subset \bigcup_{l=l_{\min}}^{l_{\max}} G_l^d$ a RSM of regularity m_r, and a cell $C \in G$, we set

- *Refinement* to be the process of replacing a cell C in G *by all of its children* from the level $L(C) + 1$.
- *Coarsening* to be the process of replacing a cell C *and all of its siblings* in G by their parent cell from the level $L(C) - 1$.

As we expect the mesh G to maintain the RSM properties after the refinement and coarsening, both operations are subject to additional constraints, i.e. the resulting cells have to be of refinement level l such that $l_{\min} \leq l \leq l_{\max}$, and the resulting mesh should respect the mesh regularity condition (4). See Section 2.4, for a detailed description of the refinement–coarsening procedures.

2.2. Cell representation for nonuniform meshes

The proposed mesh administration and book-keeping technique has the benefit of being a unified approach over 1-, 2-, and 3D domains, although for the sake of presentation we will discuss here only the 1- and 2D cases.

In what follows we present the technique in its constituent components: the representation of cells and meshes, and how basic operations like refinement, coarsening or accessing siblings or neighbours are performed. For ease of the presentation, we start with the 1D case.

Cell representation in 1D

We consider all possible mesh cells $C \in \bigcup_{l=l_{\min}}^{l_{\max}} G_l^1$ and identify their centres, relative positions, and sizes with respect to the levels of refinement in Table 1. We note at first that every cell C can be uniquely defined by its refinement level $L(C)$ and its centre $M^p(C)$. The centre $M^p(C)$ in turn can be uniquely characterized by the refinement level $L(C)$ and an *intra-level index* k enumerating the cells in the current level, see also Table 1.

Hence, the full sequence of dyadic meshes $\bigcup_{l=l_{\min}}^{l_{\max}} G_l$ can be described by the matrix

$$\mathcal{C} = \begin{bmatrix} \mathbf{k} \\ \mathbf{l} \\ \mathbf{p} \\ \mathbf{d_l} \\ \mathbf{d_r} \end{bmatrix}^T, \tag{5}$$

where the lines (i.e. the columns of $\mathcal{C}^T$) represent the cells $C \in \bigcup_{l=l_{\min}}^{l_{\max}} G_l^1$ and the line vectors $\mathbf{k}$, $\mathbf{l}$, and $\mathbf{p}$ include their intra-level index, their refinement level, and the line of C in which the parent of the current cell is located. Furthermore $\mathbf{d_l}$, $\mathbf{d_r}$ are the lines where the children cells (left and right respectively) of the cell are stored in terms of line numbers of $\mathcal{C}$. Every cell included in the matrix $\mathcal{C}$ can be characterized uniquely by its refinement level l and intra-level index k, or by the corresponding line of the matrix.

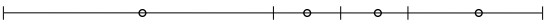

Fig. 1. The 1D grid with centres $\left\{\frac{1}{4}, \frac{9}{16}, \frac{11}{16}, \frac{7}{8}\right\}$ and sizes $\left\{\frac{1}{2}, \frac{1}{8}, \frac{1}{8}, \frac{1}{4}\right\}$ is equivalently given by the set $\{2, 12, 13, 7\}$ including the corresponding lines of the matrix C.

Accordingly, the tree-example presented in Table 1 can be written in a matrix formulation as

$$\mathcal{C} = \left[\begin{array}{c|cc|cccc|cccccccc} 1 & 1 & 2 & 1 & 2 & 3 & 4 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \\ 0 & 1 & 1 & 2 & 2 & 2 & 2 & 3 & 3 & 3 & 3 & 3 & 3 & 3 & 3 \\ - & 1 & 1 & 2 & 2 & 3 & 3 & 4 & 4 & 5 & 5 & 6 & 6 & 7 & 7 \\ 2 & 4 & 6 & 8 & 10 & 12 & 14 & - & - & - & - & - & - & - & - \\ 3 & 5 & 7 & 9 & 11 & 13 & 15 & - & - & - & - & - & - & - & - \end{array}\right]^T, \tag{6}$$

where the vertical lines separate cells of different refinement levels, with $l = 3$ being the maximum in this case. The empty component in the first column of $\mathcal{C}^T$ implies that the corresponding first cell does not posses a parent cell within $\bigcup_{l=l_{\min}}^{l_{\max}} G_l^1$. Similarly, the empty components in the fourth and fifth line of $\mathcal{C}^T$ imply that the corresponding cells do not posses any children cells in $\bigcup_{l=l_{\min}}^{l_{\max}} G_l^1$.

Well posedness. The centres of the cells of different levels cannot coincide since, otherwise, that would mean, for two cells with levels of refinement $l_1 < l_2$ and intra cell indices k_1, k_2 that:

$$\frac{2k_1 - 1}{2^{l_1+1}} = \frac{2k_2 - 1}{2^{l_2+1}} \quad \Leftrightarrow \quad \mathbb{N} \ni 2k_1 - 1 = \frac{2k_2 - 1}{2^{l_2-l_1}} \notin \mathbb{N}.$$

Since the size of every cell of level l is $\frac{1}{2^l}$, cells of different levels will not intersect, unless one of them is a descendant of the other.

Size of $\mathcal{C}$ and memory usage in 1D. As the refinement level l is upper bounded by $l_{\max}$, the size of the matrix $\mathcal{C}$ is finite. In particular, assuming a cascade of meshes starting from the coarse mesh $G^1_{l_{\min}=5}$ of 32 cells up to the fine mesh $G^1_{l_{\max}=11}$ of 2048 cells, the matrix $\mathcal{C}$ in its formulation (5) would have $(2^5 + \cdots + 2^{11}) = 4096$ lines, accounting for the levels $l = 5, \ldots, 11$, and $4096 \times 5 = 20480$ (unsigned) integer valued entries. Accounting for 4 bytes per integer, the memory needed to store $\mathcal{C}$ is 80 KB.

Mesh representation. Based on the above formulation, the computational grid can be represented by the (finite) sequence of line numbers of C corresponding to the cells of the grid. Consider, for example, the 1D mesh

$$G^1 = \left\{\left(\frac{1}{4}, \frac{1}{2}\right), \left(\frac{9}{16}, \frac{1}{8}\right), \left(\frac{11}{16}, \frac{1}{8}\right), \left(\frac{7}{8}, \frac{1}{4}\right)\right\}, \tag{7}$$

where each included cell is characterized by its centre $M^p(C)$ and its size $|D(C)|$, see also Fig. 1. The size of each cell of level l is 2^{-l}, its centre is given through the intra-level index k as $\frac{2k-1}{2^{l+1}}$. The corresponding matrix $\mathcal{C}$ is given by (6). The matrix lines that correspond to the mesh (7), and in effect represent it, are given by the set

$$G_{\mathcal{C}} = \{2, 12, 13, 7\}. \tag{8}$$

The order in which the cells appear in $G_{\mathcal{C}}$ follows the *Z-order* or Lebesgue space filling curve, namely, they are order first with respect to their refinement level l, and then to their intra-level index k. This becomes most important in 2- and higher spatial dimensions. From the implementation point of view, we use vectors of the form (8), instead of (7), to communicate the grid between different parts of the algorithm.

The following example considers the adaptation of $G_{\mathcal{C}}$ in the case of refinement: if one step of the refinement takes place and the cell $\left(\frac{1}{4}, \frac{1}{2}\right) \in G^1$ is replaced by its children cells, the mesh cell $2 \in G_{\mathcal{C}}$ needs to be replaced by its children cells, represented in the lines 4 and 5 of $\mathcal{C}$, and the grid $G_{\mathcal{C}}$ becomes

$$G_{\mathcal{C}}^{\text{new}} = \{4, 5, 12, 13, 7\}.$$

Cell representation in 2D

In dimension two, we include two additional columns in the matrix $\mathcal{C}$ to account for all the children, i.e.

$$C = \left[\begin{array}{c} \mathbf{k} \\ \mathbf{l} \\ \mathbf{p} \\ \mathbf{d_{NW}} \\ \mathbf{d_{NE}} \\ \mathbf{d_{SW}} \\ \mathbf{d_{SE}} \end{array}\right]^T, \tag{9}$$

where the intra-level index in the line vector **k** describes the cells of the same level in a lexicographic order first with respect to the x- and the y- coordinate of the centre of the cell. Here $\mathbf{d_{NW}}$, $\mathbf{d_{NE}}$, $\mathbf{d_{SW}}$ and $\mathbf{d_{SE}}$ are the lines of C^T corresponding to the four children cells located northwest (NW), northeast (NE), southwest (SW), and southeast (SE) with respect to the centre of the parent cell indicated by the matrix line. As in the 1D case, the line vectors **k**, **l** and **p** include the intra-level index of the cell, its refinement level, and the line of its parent cell. Again each cell can be uniquely identified by its refinement level and its intra-level index.

Size of C and memory usage in 2D. It is more instructive to describe the memory needed for storing the matrix $\mathcal{C}$ via an example that covers a wide class of numerical experiments. The general case follows in a similar way. We assume hence, a cascade of meshes G_l^2, with $l_{\min} \leq l \leq l_{\max}$ where $l_{\min} = 4$, i.e. 16×16 cells are included on the coarsest mesh and $l_{\max} = 10$, i.e. 1024×1024 cells are included on the finest mesh. The total number of entries of the matrix $\mathcal{C}$ is 9,786,112 and, with 4 bytes per input, the memory needed is approximately 39 MB.

Such a memory consumption is not a strong constraint, especially since it refers to a high resolution in 2D. Confer, e.g., [44] where a similar mesh resolution was used to solve the first stage of a tsunami wave. Despite that, we can even more reduce the memory usage after noting that the cells on the coarsest level $l_{\min}$ do not posses any parent cells, nor do the cells in the finest level $l_{\max}$ posses any children cells. This means that the matrix $\mathcal{C}$ will need 4,194,560 less entries, and the actual memory used is reduced to approximately 22 MB. To achieve such savings in practice the columns of $\mathcal{C}$ are separately stored in memory blocks of different sizes.

An even further reduction in memory consumption can be achieved by removing the k- and l-column from the matrix $\mathcal{C}$. The cell level and the intra-level index can be easily recomputed on the fly as follows: if i is the matrix line corresponding to a cell, its level is given by the smallest integer $l \geq l_{\min}$ such that

$$k = \sum_{j=l_{\min}}^{l-1} 2^{jd} - i$$

is positive. Then, the intra-level index is also given by k. In the example above the memory needed reduces to approximately 11 MB this way. Note that the efficient mesh administration algorithms that we propose employ an inclusion map, see (11), which requires a small amount of additional memory.

Size of C and memory usage in 3D. In the same manner we note that the memory required for a 3D case that spans from a coarse mesh of $16 \times 16 \times 16$ cells to a fine mesh of $256 \times 256 \times 256$ cells will be approximately 292 MB. These requirements can also be reduced by computing the k- and l-columns on the fly to approximately 146MB, i.e. an average of 16 or 8 bytes per mesh cell respectively. To computationally facilitate the search for neighbours, we derive an inclusion map, see (11), which has an additional burden of 36.8 MB. These memory costs are affordable, especially if it is taken into account that the resolution $256 \times 256 \times 256$ is adequate even for challenging problems, e.g. in [45] such mesh resolution was deemed sufficiently accurate for the comparative study of numerical methods solving the turbulent Rayleigh–Taylor instability. Additional memory can be saved if the refinement level l is stored in $\mathcal{C}$ as an unsigned integer of 1 byte size; this would result in a further reduction by 18 MBs.

In an even larger scale, further reduction in memory usage can be achieved by elaborate compression techniques, see e.g. [22], but they fall beyond the scope of this paper. However, we note that the above mentioned 2- and 3D mesh resolutions are memory-wise affordable and adequate for a wide range of academic and non-academic studies.

Comparison with other grid administration techniques

The reported memory consumption in `alugrid` [40] ranges between 700–800 `bytes` per element for hexahedral elements. Due to the structure and the information stored, `alugrid` cannot be directly compared to our method, but assuming a memory consumption of 146 MB, `alugrid` can store approximately the information for 2×10^5 cells whereas our method stores the information for 2×10^7 cells.

The memory consumption of `p4est` on 3D domains, as discussed in [36], is 24 bytes per octant. Our proposed method on the other hand necessitates 16 bytes per octant. However, in our case we store the information of the full tree rather than the current mesh conformation as done in `p4est`. In a more generally setting, `p4est` aims to the scalable parallelism to thousands of processors, whereas our approach emphasizes on the simplicity, ease of implementation, and portability of the code.

2.3. Siblings and neighbours

From the information stored in the matrix $\mathcal{C}$, the parent and the children cells of every cell are directly accessible. However, in practical situations further information is needed. In particular, the identification of the siblings and the

neighbours of a given cell is important in e.g. the coarsening procedure or the computation of the numerical fluxes. In the following we describe these operations, based on the information stored in the matrix $\mathcal{C}$.

Siblings

Let a cell C be represented in the line i of the matrix $\mathcal{C}$. Then, its parent cell can be found in the line $m = \mathcal{C}(i, 3)$, and its group of siblings in the lines $\mathcal{C}(m, 4)$ through $\mathcal{C}(m, 5)$ in 1D, and $\mathcal{C}(m, 4)$ through $\mathcal{C}(m, 7)$ in 2D.

To identify the relative position of a cell among its siblings, we use the ordering with respect to the (intra-level) index k. In 2D in particular, a cell is a *south*-(S), *north*-(N), *west*-(W), or *east*-(E) sibling according to the rules

$$\begin{cases} \text{W-cell}, & \text{if } k \text{ is odd} \\ \text{E-cell}, & \text{if } k \text{ is even}, \end{cases} \quad \text{and} \quad \begin{cases} \text{N-cell}, & \text{if } \left\lfloor \frac{k}{2^l} \right\rfloor \text{ is odd} \\ \text{S-cell}, & \text{if } \left\lfloor \frac{k}{2^l} \right\rfloor \text{ is even.} \end{cases} \tag{10}$$

Neighbours

To efficiently identify the neighbours of a cell, we employ the regularity of the mesh. For a RSM G of smoothness m_r, and a cell $C \in G$ of level $L(C) = l$ and intra-level index k we identify its neighbours by distinguishing the following cases:

Neighbours of the same level. The neighbours of C on the uniform mesh G_l^d can be easily found using the intra-level index k. Neighbours to the left and right in dimension one, and to the W- and E- in dimension two, have respectively indexes $k \mp 1$, while neighbours to the S- and N- in dimension two have indexes $k \mp 2^{l-1}$. Thus they are represented in the corresponding ∓ 1 and $\mp 2^{l-1}$ lines of the matrix C relative to the cell C. The *same level neighbours* however are not necessarily cells of the grid G. Nevertheless the identification of these 2 cells in 1D, or 4 cells in 2D is crucial in the neighbour finding process.

Neighbours of lower levels. If N is a same level neighbour of C, but not part of the actual grid, i.e. $N \notin G$, then an ancestor A of N could be included in the current mesh G. If there is such an ancestor $A \in G$, then A is also a neighbour of C due to the grid structure. In order to find all possible neighbours, we check m_r generations of ancestors for inclusion in the current grid. These ancestors can be identified by iteratively using the parent-cell entry m of the matrix C and switching to the corresponding line. This means that for at most $2m_r$ cells in 1D and $4m_r$ cells in 2D it needs to be examined if they are included in G.

Neighbours of higher levels. As before, let N be a same-level neighbour of the cell C. If *neither N nor any of its ancestors* is included in G, we look for neighbours of C among the descendants of N. Once again, m_r generations have to be screened. In the algorithm we propose, we exploit the relative position of N, e.g. assume that N is an E-neighbour of C in 2D and proceed as follows:

> starting with a queue containing only N we iterate through the queue by checking each entry for inclusion in G. If it is included, we have found a neighbour, otherwise we add the NE- and SE-child of the entry to the queue for the next iteration step.

This way all neighbours of C, among the descendants of N, can be found in at most $m_r + 1$ iteration steps.

For an efficient computation of all the neighbours of a cell $C \in G$, we propose to compute all same level neighbours first, and afterwards check the same level neighbours themselves, then their ancestors, and last their descendants for being included in the grid G. To allow for computationally inexpensive checks for inclusion of a cell in the current mesh, we derive an inclusion map from the grid: For $G = \{C_1, \ldots, C_N\}$ we store

$$m_G(C) = \begin{cases} k & \text{if } C = C_k \in G \\ 0 & \text{if } C \notin G \end{cases} \tag{11}$$

for every cell C decoded by the matrix $\mathcal{C}$. This map is also used for the handling of approximate functions in FV schemes. Note that the additional memory requirements are minor, e.g., in the 3D example in Section 2.2 approximately 36.8 MB are required.

2.4. Refinement/coarsening

If the mesh is used for the numerical solution of PDEs, we employ the monitor function M, which assigns a non-negative value to each cell of the grid and accordingly marks cells for refinement and coarsening. Typically, the marking is decided by two threshold values $\theta_{\text{coars}} < \theta_{\text{refin}}$. The cells for which the monitor function is below θ_{coars} are marked for coarsening, whereas the cells where the value of the monitor function is greater θ_{refin} are marked for refinement.

By $G_{\text{ref}(T)}$ we denote the mesh G where all cells in $T \subset G$ have been refined once, i.e. replaced by their children cells, similarly $G_{\text{coars}(T)}$ denotes the mesh G where all cells in $T \subset G$ have been coarsened once, i.e. replaced by their

corresponding parent cell. In what follows, we assume that after evaluation of the mesh by the monitor function, we have identified the subsets $T_r, T_c \subset G$ that include cells which are marked for refinement and coarsening, respectively.

Strong refinement and weak coarsening. Note that the meshes $G_{\text{ref}(T_r)}$, and $G_{\text{coars}(T_c)}$ might not satisfy the grid regularity condition (4). Hence T_r and T_c have to be adjusted according to the structure of G. In practice, we set higher priority on refinement and lower priority on coarsening; we aim for *strong refinement* and *weak coarsening*. That is, we possibly refine more cells than initially marked for refinement and coarsen less cells than initially marked for coarsening. In effect, the actually refined and coarsened meshes are given by $G_{\text{ref}(T_R)}$ for $T_R \supseteq T_r$ and $G_{\text{coars}(T_C)}$ for $T_C \subseteq T_c$, respectively.

For each cell C_m we denote the *dependency sets* by

$$D_r(C_m) = \{\text{smallest set } T \subseteq G : \ C_m \in T, \quad G_{\text{ref}(T)} \text{ satisfies (4)}\},$$
$$D_c(C_m) = \{\text{smallest set } T \subseteq G : \ C_m \in T, \quad G_{\text{coars}(T)} \text{ satisfies (4)}\}.$$

Given these definitions we conduct strong refinement and weak coarsening using

$$T_R = \bigcup_{C_r \in T_r} D_r(C_r), \text{ and } T_C = \{C_c \in T_c \mid D_c(C_c) \subseteq T_c\}.$$

Algorithm. We conduct both, strong refinement and weak coarsening by starting with $T_R \leftarrow T_r$, $T_C \leftarrow T_c$ and iterating through T_R and T_c from the highest to the lowest level of the cells. If a cell $C_r \in T_R$ in the refinement process has a neighbour C_N with $L(C_r) - L(C_N) = m_r$, we mark the neighbour for refinement, i.e. we add C_N to the refinement set T_R. If a cell $C_c \in T_C$ in the coarsening process has a neighbour C_N with $L(C_N) - L(C_c) = m_r$, and $C_N \notin T_C$, we remove C_c from the coarsening set T_C. We refer to the Algorithms 1 and 2 for further details.

In numerical computations, we perform a *mesh update* of a grid G as follows: we first compute the monitor function, mark cells for refinement and coarsening and deduce the sets T_r and $\tilde{T}_c$ using threshold values as described above. Then we conduct strong refinement using the effective refinement set T_R. Since the refined mesh might not include all the cells that were initially marked for coarsening, i.e. possibly $\tilde{T}_c \not\subseteq G_{\text{ref}(T_R)}$, we afterwards conduct weak coarsening using the updated set $T_c = \tilde{T}_c \setminus T_R \subseteq G_{\text{ref}(T_R)}$.

Algorithm 1 Strong refinement. Mesh cells are marked for refinement using an indicator function and then sorted by their refinement level in $T_{l_{\min}}, \ldots, T_{l_{\max}}$. In the following iterations sorted by refinement level, conflicts, due to condition (4), are resolved by additionally refining neighbours causing these conflicts.

Input: the current mesh G^{old} and the numerical solution U^{old}
Initialize the new mesh $G^{\text{new}} \leftarrow G^{\text{old}}$
Initialize $T_{l_{\min}}, \ldots, T_{l_{\max}} \leftarrow \emptyset$
for $C \in G^{\text{old}}$ **do**
  if $M(C) > \theta_{\text{refin}}$ and $L(C) < l_{\max}$ **then**
    $T_{L(C)} \leftarrow T_{L(C)} \cup C$
  end if
end for
for $l = l_{\max} - 1, \ldots, l_{\min} + m_r$ **do**
  for $C \in T_l$ **do**
    find the set of neighbours $N(C)$ of C
    for $C_N \in N(C)$ **do**
      if $l - L(C_N) = m_r$ and $C_N \notin T_{l-m_r}$ **then**
        $T_{l-m_r} \leftarrow T_{l-m_r} \cup \{C_N\}$
      end if
    end for
  end for
end for
for $C \in \bigcup_{l=l_{\min}}^{l_{\max}-1} T_l$ **do**
  identify the set of children $\mathcal{C}(C)$ of C
  $G^{\text{new}} \leftarrow G^{\text{new}} \setminus \{C\}$
  $G^{\text{new}} \leftarrow G^{\text{new}} \cup \mathcal{C}(C)$
  project the numerical solution from $U^{\text{old}}\big|_C$ to $U^{\text{new}}\big|_{\mathcal{C}(C)}$
end for
Output: the refined mesh G^{new} and the updated numerical solution U^{new}

Algorithm 2 Weak coarsening. Mesh cells are marked for coarsening using an indicator function and then sorted by their refinement level in $T_{l_{\min}}, \dots, T_{l_{\max}}$. In the following iterations from high to low refinement levels, coarsening of a mesh cell only takes place if all sibling cells are on the same refinement level and marked for coarsening and no conflicts due to (4) are detected.

1: **Input:** the current mesh G^{old} and the numerical solution U^{old}
2: Initialize the new mesh $G^{\text{new}} \leftarrow G^{\text{old}}$
3: Initialize the new numerical solution $U^{\text{new}} \leftarrow U^{\text{old}}$
4: Initialize $T_{l_{\min}}, \dots, T_{l_{\max}} \leftarrow \emptyset$
5: **for** $C \in G^{\text{old}}$ **do**
6: **if** $M(C) < \theta_{\text{coar}}$ and $L(C) > l_{\min}$ **then**
7: $T_{L(C)} \leftarrow T_{L(C)} \cup \{C\}$
8: **end if**
9: **end for**
10: **for** $l = l_{\max}, \dots, l_{\min} + 1$ **do**
11: **for** $C \in T_l$ **do**
12: identify the set of siblings $S(C)$ of C
13: **if** $S(C) \not\subset T_l$ **then**
14: $T_l \leftarrow T_l \setminus \{S(C)\}$
15: **continue** for loop in line 11
16: **end if**
17: **if** $l < l_{\max} - m_r$ **then**
18: compute the set $\mathcal{N}(C) = \{\text{neighbour cells of all cell in } S(C)\}$
19: **for** $C_N \in \mathcal{N}(C)$ **do**
20: **if** $L(C_N) - l = m_r$ and $C_N \notin T_{l+m_r}$ **then**
21: $T_l \leftarrow T_l \setminus \{S(C)\}$
22: **continue** for loop in line 11
23: **end if**
24: **end for**
25: **end if**
26: identify the parent cell C_p of C
27: $G^{\text{new}} \leftarrow G^{\text{new}} \setminus \{S(C)\}$
28: $G^{\text{new}} \leftarrow G^{\text{new}} \cup \{C_p\}$
29: project the numerical solution $U^{\text{old}}\big|_{S(C)}$ to $U^{\text{new}}\big|_{C_p}$
30: **end for**
31: **end for**
32: **Output:** the coarsened mesh G^{new} and the updated numerical solution U^{new}

2.5. Data handling and projection in FV

In practice a RSM is used to handle piecewise defined functions. Let us consider a measurable function $u : (0, 1)^d \to \mathbb{R}$ and its piecewise constant representation on a RSM G,

$$u_G(x) = \sum_{C \in G} U_C \chi_{D(C)}(x),$$

where

$$U_C = |C|^{-1} \int_C u(x)\, dx \tag{12}$$

and where χ_D is the characteristic function of the set $D \subset \Omega$. If we assume the representation of the grid as a vector $G = \{C_1, C_2, \dots, C_N\}$, such a function can be simply stored in a corresponding vector $U_G = \{U_{C_1}, U_{C_2}, \dots, U_{C_N}\}$.

In our implementation the information of both G and U_G are stored in arrays of the same size that in general allow for more entries as cells are included in the grid. Initially, we allocate arrays of a generous size, which we later extend in large increments if further memory for refinement is required. The array including the information in G is not ordered and allows for holes since array entries for cells removed by coarsening are "deleted" and set to zero. To account for newly added cells by refinement the first free memory positions being zero are used. The cell average U_C in the array for U_G is stored at the same index where the corresponding cell C is stored in the array for G. To identify a cell and associated data in memory we use the inclusion map (11). We compute cell-wise fluxes in our numerical simulations for which this approach has been efficient. However, we note that the data needs to be reordered to efficiently use common linear algebra packages such as LAPACK [46].

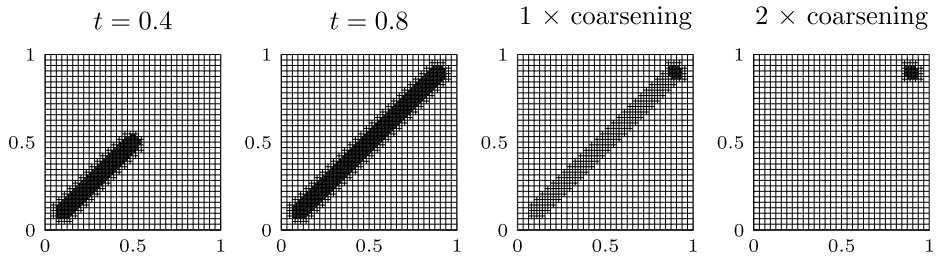

Fig. 2. Diagonal movement of the monitor function (15) with refinement at intermediate and final time and after one and two subsequent coarsening steps.

Projection to lower levels. Let G^{low} be a RSM derived by a series of subsequent coarsening operations on G. For each $C^l \in G^{\text{low}}$ we define $\mathcal{L}_G(C_l)$ to be the set that includes either C^l if $C^l \in G$ or otherwise all descendants of C^l that are included in G. The elements in $\mathcal{L}_G(C_l)$ can be easily identified using the matrix $\mathcal{C}$. We employ the formula

$$U_{C^l} = \sum_{C \in \mathcal{L}_G(C^l)} \left(2^{L(C^l)-L(C)}\right)^d U_C, \tag{13}$$

to define the projected function $u_{G^{\text{low}}}$. This function satisfies (12) for all $C \in G^{\text{low}}$, hence no accuracy is lost.

We apply this projection after the weak coarsening procedure to update the FV solution. Further, it can be used to compute the difference (e.g. in the discrete L^1 norm) of two functions on different RSMs. In this case the function on the finer grid is first projected to the coarser grid before the difference is computed.

Projection to upper levels. This projection is needed to update piecewise constant functions after strong refinement. Let us consider a RSM G^{up}, which has been derived by a series of subsequent refinement operations on G. For each cell $C^u \in G^{\text{up}}$ we define by $\mathcal{U}(C^u)$ either C^u itself if $C^u \in G$ or otherwise the ancestor of C^u which is included in G. Once again, $\mathcal{U}(C^u)$ can be easily identified using $\mathcal{C}$. A simple projection for the definition of $u_{G^{\text{up}}}$ would be the choice

$$U_{C^u} = U_{\mathcal{U}(C^u)}. \tag{14}$$

Higher order projections, which are not considered in this work, make use of reconstructions that take into account also the neighbour cells of $\mathcal{U}(C^u)$, these include minmod, CWENO, and more [8,47,48].

3. Numerical experiments

We present three numerical applications of the mesh administration technique and the AMR method. The first one is a generic experiment without any physical or biological interpretation. In this, we demonstrate the properties of the mesh administration and the AMR techniques using predefined monitor functions to drive the grid reconstruction. The second one is a 2D explosion experiment using the 2D Euler system showing that our methods can efficiently capture the more important regions of the phenomenon. The third one is a biological application; a 2D cancer invasion model exhibiting highly dynamic and complex solutions. The computer programs employed are implemented in Fortran 2008 (mesh administration and numerical schemes) and Python 3 (visualization and error computation) and are available upon request.

3.1. Generic experiment

In a first test, we consider a 2D RSM with refinement levels set to $l_{\min} = 5$ and $l_{\max} = 7$ and mesh regularity $m_r = 1$. We start with a uniform mesh on the lowest level, i.e. $G = G_5^2$, and consider the time dependent monitor function

$$M_1(\mathbf{x}, t) = e^{-100 \left\| \mathbf{x} - (0.1+t)(1,\ 1)^T \right\|_2^2}. \tag{15}$$

Using the small time step $\Delta t = 0.005$ and starting at $t = 0$, we advance in time and perform strong refinement (without coarsening) once at every time step with threshold $\theta_{\text{ref}} = 0.8$ until the time $t = 0.8$ is reached.

The resulting meshes at $t = 0.4$ and at the final time are shown in Fig. 2. The movement of the Gaussian $M_1(\mathbf{x}, t)$ has left a trace on the grid. On the trace of the Gaussian, the cells have been refined to the maximal level 7. Cells of level 6 are only visible on the transition between the finer and the coarser grid and are a result of the mesh regularity. We then coarsen the mesh twice using $\theta_{\text{coars}} = 0.8$ without evolving the monitor function further in time. Fig. 2 exhibits the mesh after performing each weak coarsening procedure. The trace of the movement is completely coarsened after the second coarsening.

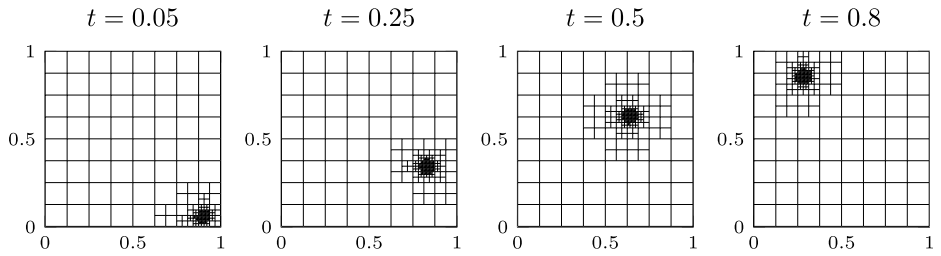

Fig. 3. Circular movement of the monitor function (16). Refinement and coarsening take place with the time evolution.

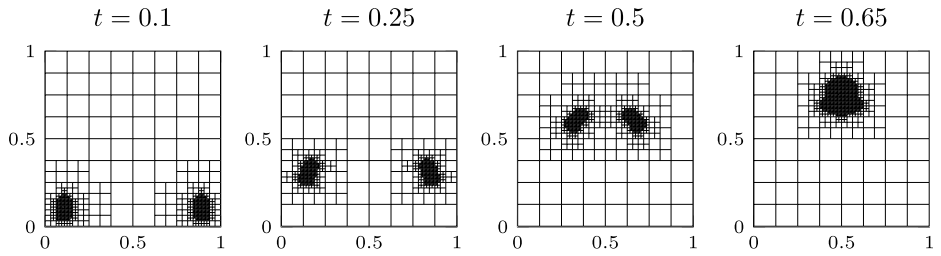

Fig. 4. Diagonally moving monitor functions meet. Note their "tails" obtained by setting the refinement and coarsening parameters θ_{refin}, and θ_{coars} to different values.

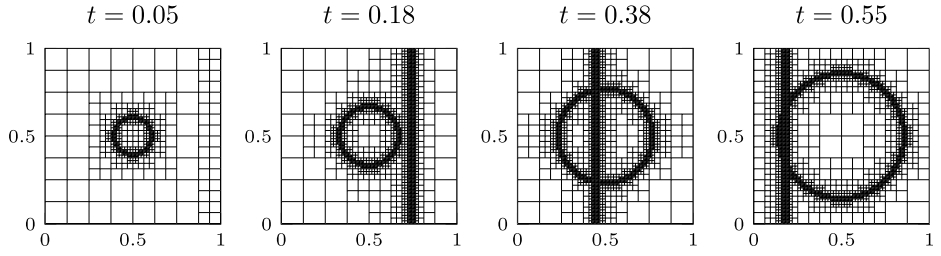

Fig. 5. Ring of increasing diameter crossed by a plane wave used as monitor function.

In a second test we employ another 2D RSM with $l_{\min} = 3$ and $l_{\max} = 7$ and mesh regularity $m_r = 1$. We start again with a uniform mesh on the lowest level, i.e. $G = G_3^2$, and consider the time dependent monitor functions

$$M_2(\mathbf{x}, t) = e^{-100 \left\| \mathbf{x} - 0.9(\cos(0.5\pi t),\ \sin(0.5\pi t))^T \right\|_2^2}, \tag{16}$$

$$M_3(\mathbf{x}, t) = e^{-100 \left\| \mathbf{x} - 0.9(\cos(\pi(1-0.5t)),\ \sin(\pi(1-0.5t)))^T \right\|_2^2} + M_2(t), \tag{17}$$

$$M_4(\mathbf{x}, t) = \begin{cases} 1, & 0.07 < \|x\|_2 - 0.5\,t < 0.1 \text{ or } 0.98 < x_1 + 1.5\,t < 1.02, \\ 0, & \text{otherwise.} \end{cases} \tag{18}$$

Starting at $t = 0$ and using the time step $\Delta t = 0.005$, we evolve in time and perform strong refinement and weak coarsening once at every time step.

In case of the monitor function M_2, and $\theta_{\text{ref}} = \theta_{\text{coars}} = 0.8$, we see in Fig. 3 that the circular movement of the Gaussian is not memorized by the mesh. We can observe a symmetric stepwise decrease of the cell levels when moving away from the centre of the Gaussian, which is refined to the maximal level.

Fig. 4 shows the same experiment using the monitor M_3 and the distinct thresholds $\theta_{\text{ref}} = 0.8$, $\theta_{\text{coars}} = 0.3$. A particular memory effect of the mesh structure can be observed in the form of "tails" following two travelling Gaussians. The appearance of the tails in this experiment is due to the different refinement and coarsening thresholds and the particular monitor function (cf. Fig. 3). Depending on the problem under consideration, e.g. in a combined shock and rarefaction wave resolved by a FV, such property might be beneficial for the numerical investigations.

In Fig. 5 we chose as monitor function a ring of increasing diameter intersected by a plane wave given in and $\theta_{\text{ref}} = \theta_{\text{coars}} = 0.5$. As the horizontally moving wave intersects the ring, cells in the tail are marked for coarsening but due to the strategy we employ and the presence of the ring some of these cells are not coarsened.

3.2. Euler Equations

In this section we apply our mesh administration and refinement technique in a FV method to solve the Euler System

$$\mathbf{U}_t + \nabla \cdot \mathbf{F}(\mathbf{U}) = 0 \quad \text{in } \Omega, \quad \mathbf{U} = (\rho, \mathbf{u}, E)^T, \quad \mathbf{u} = (u_1, u_2)^T, \tag{19}$$

$$\mathbf{U}(\cdot, 0) = \mathbf{U}_0(\cdot, t) \quad \text{in } \Omega \tag{20}$$

on the square $\Omega = (0, 1)^2$, where ρ, $\mathbf{u}$, and E represent the *density*, *velocity*, and *energy* of the fluid per unit volume. The system is equipped with transmissive boundary conditions [49], as well as with flux functions and pressure

$$\mathbf{F}(\mathbf{U}) = \begin{pmatrix} \rho\mathbf{u} \\ \rho\mathbf{u} \otimes \mathbf{u} + pI \\ (\rho E + p)\mathbf{u} \end{pmatrix}, \quad \frac{p}{\gamma - 1} = E - \frac{\rho}{2}(u^2 + v^2) \tag{21}$$

with an assumed *specific heat* ratio of $\gamma = 1.4$.

To denote the FV scheme we enumerate the cells of the RSM by $G^n = \{C_i^n,\ i = 1, \dots, N^n\}$. The mesh G^n as well as the number of included cells N^n changes throughout the computation due to the mesh adaptation that we employ. Further, we denote the set of neighbours of a cell C in G by $N(C)$. Given a time grid $t^0 = 0,\ t^n < t^{n+1} = t^n + \Delta t^n,\ n = 0, 1, 2 \dots$, we consider the cell averages

$$\tilde{\mathbf{U}}_i^n = \frac{1}{|C_i^n|} \int_{C_i^n} \mathbf{U}(\mathbf{x}, t^n)\, dx, \quad i = 1, \dots, N,$$

for which the exact evolution reads

$$\tilde{\mathbf{U}}_i^{n+1} = \tilde{\mathbf{U}}_i^n - |C_i^n|^{-1} \sum_{C_j^n \in N(C_i^n)} \int_{t^n}^{t^{n+1}} \int_{\partial C_{ij}^n} \mathbf{F}(\mathbf{U}(\mathbf{x}, t))\mathbf{n}_{ij}^n\, d\mathbf{x}\, dt \tag{22}$$

for $i = 1, \dots, N^n$. Here, ∂C_{ij}^n denotes the edge between the cells C_i^n and C_j^n and $\mathbf{n}_{i,j}^n$ is the outer normal vector of C_i^n pointing towards C_j^n. By abuse of notation we assume in (22) that $\tilde{U}_i^n$ already refers to the average after projection to C_i^n and hence $\tilde{U}_i^n$ and $\tilde{U}_i^{n+1}$ are averages over the same cell. The same is assumed for approximations in the following description. Employing approximate averages, $\mathbf{U}_i^n \approx \tilde{\mathbf{U}}_i^n$, and a numerical flux function $\mathbf{H}$, (22) transitions into the FV scheme

$$\mathbf{U}_i^{n+1} = \mathbf{U}_i^n - \Delta t^n \sum_{C_j^n \in N(C_i^n)} \frac{|\,\partial C_{ij}^n\,|}{|C_i^n|} \mathbf{H}(\mathbf{U}_i^n, \mathbf{U}_j^n, \mathbf{n}_{i,j}^n), \tag{23}$$

for $i = 1, \dots, N^n$. A time update of the FV scheme requires the identification of neighbours as well as an evaluation of the numerical flux function for each pair of neighbours. The RSM that we use allows us to compute the relation between neighbouring cell sizes and interfaces by

$$\frac{|\,\partial C_{ij}^n\,|}{|C_i|} = 2^{2L(C_i^n) - \max\{L(C_i^n), L(C_j^n)\}}.$$

We use a common vector-splitting approach, see [50], to compute both a numerical flux function $\mathbf{H}(\mathbf{U}_i^n, \mathbf{U}_j^n, \mathbf{n}_{ij}^n)$ as an approximation to the mean value of the actual flux $\mathbf{F}$ through C_{ij}^n, i.e. of

$$\frac{1}{\Delta t^n\, |\partial C_{ij}^n|} \int_{t^n}^{t^{n+1}} \int_{\partial C_{ij}^n} \mathbf{F}(\mathbf{U}(\mathbf{x}, t))\mathbf{n}\, d\mathbf{x}\, dt$$

and propagation velocities a_{ij}^n at each cell interface ∂C_{ij}^n. The latter allow us to update the time increment of our explicit scheme by the local CFL condition

$$\Delta t^n \le \text{CFL} \min_{1 \le i,j \le N^n} \frac{|\partial C_{ij}^n|}{|a_{ij}^n|}.$$

A CFL number of 0.5 is used in our simulations.

We consider the explosion experiment (cf. [51]) with initial condition

$$U^0(\mathbf{x}) = \begin{cases} (1, 0, 2.5)^T, & \text{if } \mathbf{x} \in K = \{\mathbf{x} \in \mathbb{R}^2,\ \|\mathbf{x} - (0.5, 0.5)^T\|_2 < 0.12\}, \\ (0.125, 0, 0.25)^T, & \text{otherwise.} \end{cases}$$

Using a RSM with $l_{\min} = 7,\ l_{\max} = 9$ and starting on the coarsest grid, i.e. $G^0 = G_7^2$, we perform strong refinement and subsequent weak coarsening once after each time step of scheme (23). Therefore we use the density gradient monitor (cf.

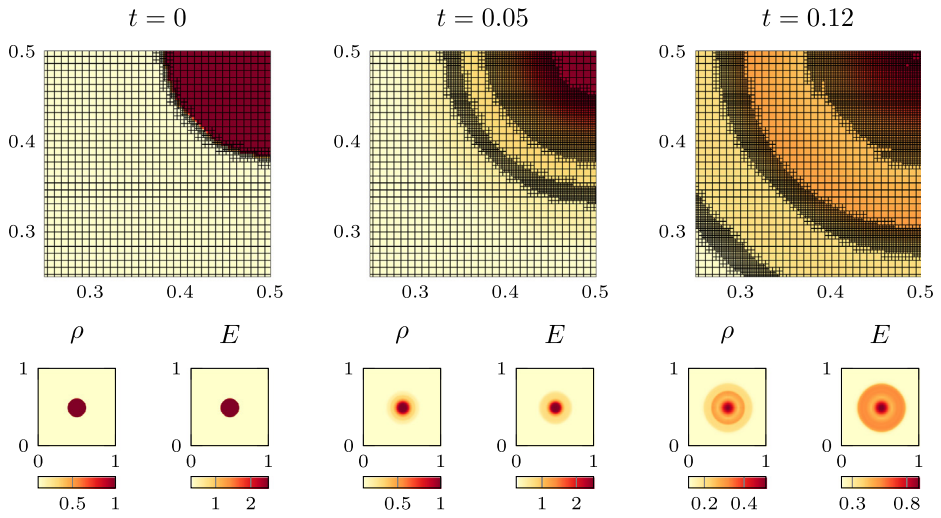

Fig. 6. Euler explosion in 2D with adaptive mesh refinement. Numerical solutions at the time instances $t = 0$, $t = 0.05$ and $t = 0.12$. Top: density and mesh in $[0.25, 0.5]^2$. Bottom: density and energy on the full domain. A shock, a contact discontinuity, and a rarefaction wave are formed and resolved on the non-uniform grid. The numerical solution employs 16,144 coarse cells ($l = 7$) and 612 ($l = 8$) + 1392 ($l = 9$) refined cells at time $t = 0$, 15,136 coarse cells ($l = 7$) and 1208 ($l = 8$) + 15,136 ($l = 9$) refined cells at time $t = 0.05$ and 13,504 coarse cells ($l = 7$) and 2864 ($l = 8$) + 34,624 ($l = 9$) refined cells at time $t = 0.12$.

also [52]) defined on each cell by

$$g_i^n = \max_{C_j^n \in N(C_i^n)} \frac{|\rho_j^n - \rho_i^n|}{\|M^p(C_i^n) - M^p(C_j^n)\|_2}, \quad M_i^n = \frac{g_i^n}{\max_{\{0 \le i \le N^n\}} g_i^n},$$

and the threshold values $\theta_{\text{ref}} = \theta_{\text{coars}} = 0.4$. We apply the projections (13) and (14) to update the numerical solutions.

In Fig. 6 we show the results of the simulation. We observe the formation of a circular shock wave travelling in an outward direction, followed by a contact discontinuity that also moves outwards, and by a rarefaction wave that moves inwards. We see that all three waves are properly resolved by the mesh, most notably the contact discontinuity, despite the lower order of accuracy of the numerical scheme that we have adopted. Note also that the transition areas between the waves are resolved by the highest refinement level.

3.3. Tumour growth model

Cancer metastasis starts with the invasion of the local *extracellular matrix* (ECM) by the cancer cells of the primary tumour. Then, it evolves as follows: the cancer cells proliferate and migrate until they find neighbouring blood vessels. They then *intravasate* (enter the blood stream) and travel with the blood flow. At a secondary location of the organism they *extravasate* (exit from the blood stream) to a new organ and engender new tumours. For that reason, the invasion of the ECM is considered to be one of the "*hallmarks of cancer*" [14].

The model we address in this work is a 2D, macroscopic, deterministic, ARD system that describes the densities of the cancer cells c as the primer unknown variable, the density v of the *collagen* on which cancer cells adhere and move as the main component of the *extracellular matrix* (ECM), and the density of a generic enzyme m of the *matrix metalloproteinases* (MMPs) family that is secreted by the cancer cells and is responsible for the degradation of the ECM.

There is a wide variety of cancer invasion models of this type that has been proposed in the literature e.g. [19,20,53–59]; we employ here a model similar to [60] due primarily to its simplicity. This model reads

$$\mathbf{U}_t + \nabla \cdot \mathbf{F}(\mathbf{U}, \nabla \mathbf{U}) = \mathbf{S}(\mathbf{U}), \quad \mathbf{U} = (c, v, m)^T \quad \text{in } \Omega, \tag{24}$$

$$\mathbf{U}(\cdot, 0) = \mathbf{U}_0(\cdot, t) \quad \text{in } \Omega, \quad \frac{\partial \mathbf{U}}{\partial \mathbf{n}} = 0 \quad \text{on } \partial\Omega, \tag{25}$$

where we have assumed homogeneous Neumann boundary conditions on the square $\Omega = (0, 1)^2$ and flux function and source term given by

$$\mathbf{F}(\mathbf{U}, \nabla \mathbf{U}) = \begin{pmatrix} \chi c \nabla v - D_c \nabla c \\ 0 \\ -D_m \nabla m \end{pmatrix}, \quad \mathbf{S}(\mathbf{U}) = \begin{pmatrix} \mu c(1 - c) \\ -\delta v m \\ \alpha c - \beta m \end{pmatrix}. \tag{26}$$

One of the reasons for choosing to work with this system is the different motility properties that each component exhibits. The cancer cells, represented by c, move using their motility apparatus on the ECM v in a particular manner:

they exhibit a preferred direction towards higher densities of the ECM (*haptotaxis* part of the model), but still a part of their motion is random and is understood as cellular diffusion. The ECM v does not move or otherwise translocate. The MMPs m diffuse freely in the *interstitial fluid* without possessing any motility mechanism (molecular diffusion).

In addition to the above motility properties, the cancer cells *proliferate* and produce MMPs with a constant rate. The MMPs attach on the ECM which they instantly dissolve (at least compared to the invasion time scale of the model), and disassemble to their constituents due to chemical degradation, see e.g. [61,62].

For the needs of this paper we perform two numerical experiments in which we use the parameters

$$\chi = 2 \times 10^{-2},\ D_c = 2 \times 10^{-4},\ D_m = 10^{-3},\ \mu = 0.5,\ \delta = 4,\ \alpha = 0.5,\ \beta = 0.3. \tag{27}$$

For the numerical treatment we proceed in a similar way as with the Euler equations in Section 3.2. We solve the system (24)–(26) using a FV approach, and consider a time grid $t^0 = 0,\ t^{n+1} = t^n + \Delta t^n$ for $n = 0, 1, \ldots$. We consider at every time t^n a RSM $G^n = \left\{C_i^n,\ i = 1, \ldots, N^n\right\}$ where the number of computational cells N^n is adapted during the computation. The numerical solution itself is denoted at time t^n by $\mathbf{U}_i^n$ for $i = 1, \ldots, N^n$ and is obtained through the numerical scheme

$$\mathbf{U}_i^{n+1} = \mathbf{U}_i^n + \Delta t^n \mathbf{S}(\mathbf{U}_i^n) - \Delta t^n \sum_{C_j^n \in N(C_i^n)} \frac{|\partial C_{ij}^n|}{|C_i^n|} \mathbf{H}(\mathbf{U}_i^n, \mathbf{U}_j^n, \mathbf{n}_{ij}^n), \tag{28}$$

for $i = 1, \ldots, N^n$, where ∂C_{ij}^n denotes the edge between the cells C_i^n and C_j^n and $\mathbf{n}_{i,j}^n$ the outer normal vector of C_i^n pointing towards C_j^n. By abuse of notation we assume in (28) that U_i^n already refers to the approximate solution after projection to C_i^n and hence both U_i^n and U_i^{n+1} are considered on the same cell. The combined diffusion and haptotaxis flux is approximated by the numerical flux function

$$\mathbf{H}(\mathbf{U}_i^n, \mathbf{U}_j^n, \pm\mathbf{e}_k) = \pm \begin{pmatrix} -D_c\, \nabla_{i,j}^h c^n & +\chi(\nabla_{i,j}^h v^n)^+ c_i - \chi(\nabla_{i,j}^h v^n)^- c_j \\ 0 & \\ -D_m\, \nabla_{i,j}^h m^n & \end{pmatrix}, \tag{29}$$

where $\mathbf{e}_k \in \{\mathbf{e}_1, \mathbf{e}_2\}$ refers to an unit vector in $\mathbb{R}^2$ and we use the notations

$$\mathbf{U}_i^n = \begin{pmatrix} c_i^n \\ v_i^n \\ m_i^n \end{pmatrix}, \quad \begin{pmatrix} \nabla_{i,j}^h c^n \\ \nabla_{i,j}^h v^n \\ \nabla_{i,j}^h m^n \end{pmatrix} = \frac{\mathbf{U}_j^n - \mathbf{U}_i^n}{|\,\partial C_{ij}^n\,|}$$

for $1 \le i, j \le N^n$. Moreover, the positive and negative part are defined by

$$f^+ = \max\{0,\ f\}, \quad f^- = -\min\{0,\ f\}.$$

In (29) we employ for simplicity classical finite differences to discretize diffusion, which are not considered consistent in case of hanging nodes as they appear in our AMR scheme, clarify [63]. Using this discretization an additional error is introduced, which depends on the area of all cells that have a hanging node at their boundary [63,64]. While for this reason in general a more specialized scheme for diffusion on nonconforming meshes such as [65] is preferable, the finite difference discretization in combination with AMR can still improve the accuracy of the full scheme over a uniform method, when the number of hanging nodes is controlled for example by intervention in the scheme. We verify in our first tumour growth experiments that the number of hanging nodes does not become too large even without modification of the scheme and that convergence can be observed experimentally (see Table 4).

We employ time steps Δt^n according to the condition

$$\Delta t^n \le \mathrm{CFL} \min_{1 \le i,j \le N^n} \min \left\{ \frac{|\partial C_{ij}^n|}{\chi\, |\nabla_{ij}^h v^n|}, |(\partial C_{ij}^n)|^2\, D_m^{-1} \right\} \tag{30}$$

using the CFL number 0.5. Due to the explicit scheme we employ, the quadratic term in (30) is required for numerical stability. However, due to the low diffusivity this term does not cause a significant computational burden. We discretize the domain using a RSM with $l_{\min} = 5$ and $l_{\max} = 7$, start the simulation on the grid $G^0 = G_5^2$ and perform both strong refinement and weak coarsening once after each time step of scheme (28). Similar as in Section 3.2 we consider the gradient of the cancer density

$$g_i^n = \max_{C_j^n \in N(C_i^n)} \frac{|c_j^n - c_i^n|}{\|M^p(C_i^n) - M^p(C_j^n)\|_2}, \quad M_i^n = \frac{g_i^n}{\max_{\{0 \le i \le N^n\}} g_i^n}$$

and use the threshold values $\theta_{\mathrm{ref}} = 0.2,\ \theta_{\mathrm{coars}} = 0.1$. We project the solution between the cells as discussed in Section 2.5. The domain

$$\Omega_{\mathrm{top}} = \left\{ \mathbf{x} \in \Omega,\ x_2 \ge \sin\left(\frac{x_1^3}{125} + \frac{2x_1 + 26}{25} + \frac{1}{20} \right) \right\},$$

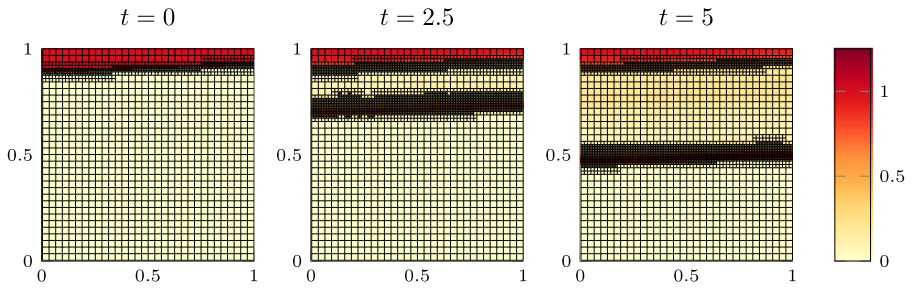

Fig. 7. Numerical simulation of cancer invasion on a 2D domain with adaptive mesh refinement (time instances $t = 0$, $t = 2.5$ and $t = 5$ of the cancer cell density). The cancer cells invade the ECM in the form of waves and the mesh is refined at the regions of interest.

Table 2
L^1 errors and relative CPU times of the scheme in the tumour growth experiment at $t = 2.5$ in case $v_0 = 1$. The solution computed on an uniform mesh on level $l = 8$ was used as reference in the error computation. The adaptive grid solution, which uses 789 coarse cells ($l = 5$) and 396 ($l = 6$) + 2176 ($l = 7$) refined cells, outperforms the uniform grid solutions on level 5 and level 6 in terms of error while taking less CPU time than the uniform grid solution on level 7. More specifically, in order to obtain approximately half the error of the adaptive mesh solution, the uniform one on level 7 should be employed, but it implies approximately 5 times more mesh cells and approximately 6 times more CPU time.

Grid	Mesh cells	L^1 error	Relative CPU time
Uniform $l = 5$	1024	4.092×10^{-2}	4.92%
Uniform $l = 6$	4096	1.743×10^{-2}	44.81%
Uniform $l = 7$	16384	7.147×10^{-3}	615.69%
Adaptive $l \in \{5, 6, 7\}$	3361	1.338×10^{-2}	100.00%

Table 3
CPU time breakdown of the AMR method in the tumour growth experiment. The computation was run in case of the uniform initial ECM density $v_0 = 1$ until final time $T = 5$ with AMR allowing for mesh cells from level 5 to level 7 (compare also Table 2). The relative times were computed from average CPU times over 10 runs. CPU time was approximately spent half for flux computations and half for mesh updates. Within the mesh updates most CPU time was spent for the computation of the monitor function while mesh refinement and coarsening required similar CPU times. An additional test in the setting of Table 2 showed that reducing the mesh updates to one every second time step decreases the total CPU time to 73.2713% of the previous run time and the relative time for mesh updates to 35.5212%, while the error was only increased by 11%.

Flux computation	47.3346%		
Mesh update	52.6654%	Monitor computation	32.4572%
		Mesh refinement	10.1746%
		Mesh coarsening	9.4059%

determines our considered initial condition

$$\mathbf{U}_0(\mathbf{x}, t) = \begin{cases} (1,\ 0,\ 0.3)^T, & \mathbf{x} \in \Omega_{\text{top}} \\ (0,\ v_0,\ 0)^T, & \mathbf{x} \in \Omega \backslash \Omega_{\text{top}}. \end{cases} \tag{31}$$

In a first tumour growth experiment, visualized in Fig. 7, we consider the initial condition (31) for a constant $v_0 = 1$. We see the creation and the preliminary phase of cancer invasion in the form of waves. These cancer invasion waves emanate from the main body of the tumour and invade the ECM. They are followed by a smooth part of the solution with lower cancer cell density. Possible reconstruction of the ECM (not accounted for in the current model) would lead to a secondary wave of cancer cells invading the ECM in a similar way. We see that the mesh is refined in the area of the first cancer wave as well as at the front of the main body of the tumour. We also note that despite the first order of accuracy of the numerical method, there is still a gain in accuracy and efficiency, by using AMR methods. This is shown in Table 2, where the L^1 error (using a reference solution on a uniform grid on level 8) show that the adaptive case, where the mesh cells vary between level 5, 6 and 7, outperforms the uniform case on levels 5 and 6 at the time $t = 2.5$. The L^1 errors have been computed using the projection to lower levels described in Section 2.5. Moreover, the adaptive numerical solution is more affordable in terms of CPU time[1] than the uniform one on level 7. A breakdown of CPU time shown in Table 3 revealed that roughly half the computation time was spent for flux computations and half for the mesh

[1] Computations were run on a Laptop with 2.4 GHz Quad-Core Intel Core i5 CPU and 16 GB of RAM running macOS 11.4.

Table 4
Refinement levels, final number of mesh cells, maximum number of hanging nodes through all time steps, L^1-errors and experimental convergence rates of the AMR scheme for increased adaptive mesh resolutions in the tumour growth experiment. The case of a uniform initial ECM density $v_0 = 1$ and final time $T = 2.5$ was considered. CFL was chosen 0.24 and θ_{ref} was reduced to 0.1 to achieve similar refinement/coarsening dynamics on all adaptive meshes. For the error computation we employed a uniform solution on refinement level 9. AMR computations were restricted to a range of two or three successive refinement levels, which we increased in each run. (On the coarsest grid two refinement levels were only used as the initial cancer density could not be represented on refinement level 2.) The error in all three components of the model decreases as finer adaptive meshes were employed indicating convergence of the AMR scheme. We further computed the experimental order of convergence (EOC) for each component by the formula EOC $= \log_2(E_{k-1}/E_k)$ with E_k denoting an error in line k of the table assuming for simplicity that the mesh fineness halves when going from one line in the table to the next one. The computed EOCs suggest first order of convergence of the AMR scheme.

Levels	Mesh cells	Hanging nodes	c	EOC	v	EOC	m	EOC
3–4	154	9	9.362×10^{-2}		5.767×10^{-2}		5.723×10^{-2}	
3–5	448	28	4.598×10^{-2}	1.03	3.050×10^{-2}	0.92	2.517×10^{-2}	1.19
4–6	1543	92	2.189×10^{-2}	1.07	1.421×10^{-2}	1.10	1.049×10^{-2}	1.26
5–7	5377	327	1.334×10^{-2}	0.71	6.301×10^{-3}	1.17	4.663×10^{-3}	1.17
6–8	18952	924	6.983×10^{-3}	0.93	2.448×10^{-3}	1.36	2.318×10^{-3}	1.01

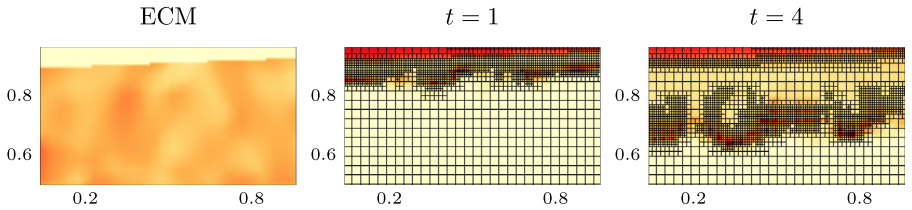

Fig. 8. Numerical simulation of cancer invasion with adaptive mesh refinement on a non uniform ECM. Left panel: initial ECM density. Middle and right panel: Cancer cells density at time instances $t = 1$ and $t = 4$. The same colourmap as in Fig. 7 is used. The migration of the cancer cells is strongly influenced by the non-uniformity of the matrix. They concentrate and invade the ECM in the form of cancer cell "islands". The refinement of the grid follows the dynamics of the cancer cells.

updates, out of which the computation of the monitor function was the most expensive part. Comparing AMR solutions on various adaptive meshes we demonstrate in Table 4 that the computed L^1-errors decrease as the mesh resolution is globally increased and thus the AMR scheme converges experimentally.

In the second experiment, shown in Fig. 8, we investigate the effect that a non-uniform ECM has in the invasion of the cancer cells by imposing a spatial structure on v_0 in (31) shown in the left panel of Fig. 8. We see that, despite the simple structure of the model, the dynamics of the solution are complex. As in the previous experiment, a propagating wave is formed. This time however, due to the non-uniformity of the ECM, the cancer cells concentrate in isolated "islands" as they move through the matrix. We can again notice the separation of these invading islands from the main body of the tumour; a behaviour that is consistent with the biomedical understanding of tumour spread, [66]. The mesh follows the dynamics of the solution and finely resolves the front of the main body of the tumour and the areas where the invasion takes place. The different refinement and coarsening thresholds θ_{ref} and θ_{coars} employed, cause a "memory effect" in the mesh, which maintains a higher resolution in previous locations of the cancer islands. These are the areas where the reconstruction of the ECM (not included in the current model) mostly takes place and a higher resolution can be very useful; another benefit of the AMR method that we employ.

4. Conclusions

The current work is motivated by our wider investigation of cancer invasion models. The special nature of these problems and, in particular, the highly dynamic behaviour of the solutions necessitate the development of specialized numerical methods and techniques. These methods can become expensive, especially in the multidimensional cases, and so AMR is sought as a way to decrease computational costs. The application of AMR methods in these problems, gives rise to a series of difficulties that need to be addressed.

We have presented here a newly developed mesh structure data administration technique used as machinery for our AMR (h-refinement) methods. When compared to existing methods in the literature, our technique exhibits similarities to pointer-based mesh data structure techniques, and exploits the rectangular structure of the mesh and its refinement by bisection.

We introduced an easy to use technique that avoids the traversing of the connectivity graph for the cell ancestry and, due to the structure of the mesh, it greatly simplifies the identification of neighbouring cells. It can be easily implemented and employed in a wide range of problems in 1-, 2-, and higher dimensional spaces. It is particularly designed for smooth meshes, and uses their smoothness dynamically in the matrix operations. The memory footprint of the method makes it affordable on coarse to very fine mesh resolutions. Additionally, our technique allows for adaptive minimum and maximum refinement levels as well as for a free choice of monitor functions and threshold parameters. Although these

properties are not investigated in the current paper, they are still potentially useful in cases where the solutions exhibit multiple dynamical phenomena or blow-ups.

We have moreover presented the components of the mesh administration technique in detail and its connection to the physical discretization of the domain. We have discussed the operations for traversing the mesh, for the identification of the sibling and neighbour cells, as well as for the local refinement and coarsening of the mesh.

Finally, we have endowed this technique with an AMR method and presented its capabilities and its flexibility in three applications. The first is a generic experiment in the absence of physical or biological laws where the mesh refinement is dictated by synthetic monitor functions. The second is a physical application of the technique and the AMR in the classical case of the Euler equation. The discussion of particular numerical solvers is beyond the scope of this paper, so we have used a common vector splitting scheme. The third application is a biological problem: a 2D tumour growth and invasion of the ECM model. This model (like other cancer invasion models) exhibits highly dynamic solutions that constitute a challenge for typical numerical methods. Even for the low order of accuracy of the numerical method that we have used we have seen a gain in efficiency and accuracy obtained by the AMR technique.

Future steps along this direction should focus mostly on the development of higher order and – for the diffusive part of the problems – implicit numerical solvers to be used in the AMR method. The development of such methods, however, necessitates extensive analysis, algorithm implementation, and numerical experimentation, which falls beyond the scope of the current paper, and is therefore postponed for a subsequent work.

Acknowledgements

NK was supported by the Postdoctoral Fellowships for Research in Japan (Standard) of the Japan Society for the Promotion of Science. NS was partly funded from the German Science Foundation (DFG) under the grant SFB 873: “Maintenance and Differentiation of Stem Cells in Development and Disease”.

References

[1] M.J. Berger, J. Oliger, Adaptive mesh refinement for hyperbolic partial differential equations, J. Comput. Phys. 53 (1984) 484–512.
[2] M.J. Berger, P. Colella, Local adaptive mesh refinement for shock hydrodynamics, J. Comput. Phys. 82 (1989) 64–84.
[3] I. Babuvska, W.C. Rheinboldt, Error estimates for adaptive finite element computations, SIAM J. Numer. Anal. 15 (2011) 736–754.
[4] R. Verfürth, A posteriori error estimation and adaptive mesh-refinement techniques, J. Comput. Appl. Math. 50 (1) (1994) 67–83.
[5] R. Teyssier, Cosmological hydrodynamics with adaptive mesh refinement, Astron. Astrophys. 385 (1) (2002) 337–364.
[6] G. Puppo, M. Semplice, Numerical entropy and adaptivity for finite volume schemes, Commun. Comput. Phys. 10 (5) (2011) 1132–1160.
[7] C. Tenaud, M. Duarte, Tutorials on adaptive multiresolution for mesh refinement applied to fluid dynamics and reactive media problems, in: ESAIM: Proceedings, Vol. 34, EDP Sciences, 2011, pp. 184–239.
[8] M. Semplice, A. Coco, G. Russo, Adaptive mesh refinement for hyperbolic systems based on third-order compact WENO reconstruction, J. Sci. Comput. 66 (2) (2016) 692–724.
[9] N. Dudley Ward, S. Falle, M.S. Olson, Modeling chemotactic waves in saturated porous media using adaptive mesh refinement, Transp. Porous Media 89 (3) (2011) 487.
[10] L. Botti, M. Piccinelli, B. Ene-Iordache, A. Remuzzi, L. Antiga, An adaptive mesh refinement solver for large-scale simulation of biological flows, Int. J. Numer. Methods Biomed. Eng. 26 (1) (2010) 86–100.
[11] S.M. Wise, J.S. Lowengrub, H.B. Frieboes, V. Cristini, Three-dimensional multispecies nonlinear tumor growth – I: Model and numerical method, J. Theoret. Biol. 253 (3) (2008) 524–543.
[12] H.B. Frieboes, X. Zheng, C.H. Sun, B. Tromberg, R. Gatenby, V. Christini, An integrated computational/experimental model of tumor invasion, Cancer Res. 66 (3) (2006) 1597–1604.
[13] N. Kolbe, J. Kat'uchová, N. Sfakianakis, N. Hellmann, M. Lukáčová-Medvid'ová, A study on time discretization and adaptive mesh refinement methods for the simulation of cancer invasion : The urokinase model, Appl. Math. Comput. 273 (2016) 353–376.
[14] D. Hanahan, R.A. Weinberg, Hallmarks of cancer: the next generation, Cell 144 (5) (2011) 646–674.
[15] J. MacKenzie, W.R. Mewki, An unconditionally stable second-order accurate ALE–FEM scheme for two-dimensional convection–diffusion problems, IMA J. Numer. Anal. 32 (2012) 888–905.
[16] H. Edelsbrunner, Geometry and Topology for Mesh Generation, Cambridge University Press, 2001.
[17] E. Espejo, K. Vilches, C. Conca, A simultaneous blow-up problem arising in tumor modeling, J. Math. Biol. 79 (2019) 1357–1399.
[18] Y. Tao, M. Winkler, Critical mass for infinite-time blow-up in a haptotaxis system with nonlinear zero-order interaction, Discrete Contin. Dyn. Syst. 41 (1) (2021) 439–454.
[19] N. Hellmann, N. Kolbe, N. Sfakianakis, A mathematical insight in the epithelial-mesenchymal-like transition in cancer cells and its effect in the invasion of the extracellular matrix, Bull. Braz. Math. Soc. 47 (1) (2016) 397–412.
[20] N. Sfakianakis, N. Kolbe, N Hellmann, M. Lukáčová-Medvid'ová, A multiscale approach to the migration of cancer stem cells: Mathematical modelling and simulations, B. Math. Biol. (2016) in press.
[21] D.P. Dobkin, M.J. Laszlo, Primitives for the manipulation of three-dimensional subdivisions, Algorithmica 4 (1989) 3–32.
[22] D.K. Blandford, E.B. Guy, D.E. Cardoze, C. Kadow, Compact representations of simplicial meshes in two and three dimensions, Int. J. Comput. Geom. AP 15 (1) (2005) 3–24.
[23] T.J. Alumbaugh, X. Jiao, Compact array-based mesh data structures, in: Byron W. Hanks (Ed.), Proceedings of the 14th International Meshing Roundtable, Springer Berlin Heidelberg, Berlin, Heidelberg, 2005, pp. 485–503.
[24] D. Canino, L. De Floriani, K. Weiss, IA An adjacency-based representation for non-manifold simplicial shapes in arbitrary dimensions, Comput. Graph. 35 (3) (2011) 747–753.
[25] V. Dyedov, N. Ray, D. Einstein, X. Jiao, T.J. Tautges, AHF: array-based half-facet data structure for mixed-dimensional and non-manifold meshes, Eng. Comput. 31 (3) (2015) 389–404.
[26] R.S. Sampath, S.S. Adavani, H. Sundar, I. Lashuk, G. Biros, Dendro: Parallel algorithms for multigrid and AMR methods on 2:1 balanced octrees, in: SC '08: Proceedings of the 2008 ACM/IEEE Conference on Supercomputing, 2008, pp. 1–12.
[27] M. Kremer, D. Bommes, L.P. Kobbelt, Openvolumemesh - A versatile index-based data structure for 3D polytopal complexes, in: IMR, 2012.

[28] B.S. Kirk, J.W. Peterson, R.H. Stogner, G.F. Carey, Libmesh : A C++ library for parallel adaptive mesh refinement/coarsening simulations, Eng. Comput. 22 (3) (2006) 237–254.
[29] M.W Beall, M.S. Shephard, A general topology-based mesh data structure, Internat. J. Numer. Methods Engrg. 40 (1997) 1573–1596.
[30] G.F. Carey, M. Sharma, K.C. Wang, A class of data structures for 2-d and 3-d adaptive mesh refinement, Internat. J. Numer. Methods Engrg. 26 (1988) 2607–2622.
[31] E.B. Becker, G.F. Carey, J.T. Oden, Finite Elements, in: The Texas Finite Element Series, Prentice-Hall, Englewood Cliffs, N.J, 1981.
[32] M. Kremer, D. Bommes, L. Kobbelt, Openvolumemesh – A versatile index-based data structure for 3D polytopal complexes, in: X. Jiao, J.-Chr. Weill (Eds.), Proceedings of the 21st International Meshing Roundtable, Springer Berlin Heidelberg, Berlin, Heidelberg, 2013, pp. 531–548.
[33] A. Fabri, G.J. Giezeman, L. Kettner, S. Schirra, On the design of CGAL, A computational geometry algorithms library, Softw. - Pract. Exp. 30 (2000) 1167–1202.
[34] W. Schroeder, K. Martin, B. Lorensen, The Visualization Toolkit: An Object-Oriented Approach To 3D Graphics ; Visualize Data in 3D - Medical, Engineering Or Scientific ; Build Your Own Applications with C++, Tcl, Java Or Python ; Includes Source Code for VTK (Supports Unix, Windows and Mac), 4. ed, Kitware, Inc, Clifton Park, NY, 2006.
[35] V. Dyedov, D.R. Einstein, Jiao X., A.P. Kuprat, J.P. Carson, F. del Pin, Variational generation of prismatic boundary-layer meshes for biomedical computing, Int. J. Numer. Methods Eng. 79 (2009) 907–945.
[36] C. Burstedde, L.C. Wilcox, O. Ghattas, p4est: Scalable algorithms for parallel adaptive mesh refinement on forests of octrees, SIAM J. Sci. Comput. 33 (2011) 1103–1133.
[37] I. Tobin, B. Carsten, L.C. Wilcox, O. Ghattas, Recursive algorithms for distributed forests of octrees, SIAM J. Sci. Comput. 37 (5) (2015) C497–C531.
[38] W. Bangerth, R. Hartmann, G. Kanschat, Deal.II—A general-purpose object-oriented finite element library, ACM Trans. Math. Software 33 (4) (2007).
[39] B. Bangerth, C. Burstedde, T. Heister, M. Kronbichler, Algorithms and data structures for massively parallel generic adaptive finite element codes, ACM Trans. Math. Software 38 (2) (2011) 14:1–14:28.
[40] M. Alkämper, A. Dedner, R. Klöfkorn, M. Nolte, The DUNE-ALUGrid module, Arch. Numer. Softw. 4 (1) (2016) 1–28.
[41] P. Bastian, M. Blatt, A. Dedner, N.-A. Dreier, Chr. Engwer, R. Fritze, C. Gräser, Chr. Grüninger, D. Kempf, R. Klöfkorn, M. Ohlberger, O. Sander, The dune framework: Basic concepts and recent developments, Comput. Math. Appl. 81 (2021) 75–112.
[42] P. Bastian, M. Blatt, A. Dedner, C. Engwer, R. Klöfkorn, M. Ohlberger, O. Sander, A generic grid interface for parallel and adaptive scientific computing. Part I: Abstract framework, Computing 82 (2008) 103–119.
[43] P. Bastian, M. Blatt, A. Dedner, C. Engwer, R. Klöfkorn, M. Ohlberger, O. Sander, A generic grid interface for parallel and adaptive scientific computing. Part II: Implementation and tests in DUNE, Computing 82 (2008) 121–138.
[44] G. Ma, J.T. Kirb, F. Shib, Numerical simulation of tsunami waves generated by deformable submarine landslides, Ocean Model 69 (2013) 146–165.
[45] G. Dimonte, D.L. Youngs, A. Dimits, S. Weber, M. Marinak, C. Wunsch, A. Robinson, M.J. Andrews, P. Ramaprabhu, A.C. Calder, B. Fryxell, J. Biello, L. Dursi, P. MacNeice, Olson K., P. Ricker, R. Rosner, F. Timmes, H. Tufo, Y.-N. Young, M. Zingale, A comparative study of the turbulent Rayleigh–Taylor instability using high-resolution three-dimensional numerical simulations: The alpha-group collaboration, Phys. Fluids 16 (5) (2004) 1668–1693.
[46] E. Anderson, Z. Bai, C. Bischof, S. Blackford, J. Demmel, J. Dongarra, J. Du Croz, A. Greenbaum, S. Hammarling, A. McKenney, D. Sorensen, LAPACK Users' Guide, Third, Society for Industrial and Applied Mathematics, Philadelphia, PA, 1999.
[47] A. Harten, B. Engquist, S. Osher, S.R. Chakravarthy, Uniformly high order accurate essentially non-oscillatory schemes, III, in: Upwind and High-Resolution Schemes, Springer, 1987, pp. 218–290.
[48] C.W. Shu, S. Osher, Efficient implementation of essentially non-oscillatory shock-capturing schemes, J. Comput. Phys. 77 (2) (1988) 439–471.
[49] M. Giles, Non-Reflecting Boundary Conditions for the Euler Equations, Computational Fluid Dynamics Laboratory, Department of Aeronautics and Astronautics, Massachusetts Institute of Technology, 1988.
[50] J.L. Steger, R.F. Warming, Flux vector splitting of the inviscid gasdynamic equations with application to finite-difference methods, J. Comput. Phys. 40 (2) (1981) 263–293.
[51] E. Toro, Riemann Solvers and Numerical Methods for Fluid Dynamics, Springer, 2009.
[52] M. Feistauer, J. Felcman, I. Straškraba, Mathematical and Computational Methods for Compressible Flow, Oxford University Press on Demand, 2003.
[53] M.A.J. Chaplain, G. Lolas, Mathematical modelling of cancer cell invasion of tissue. The role of the urokinase plasminogen activation system, Math. Models Methods Appl. Sci. 15 (11) (2005) 1685–1734.
[54] N. Bellomo, N.K. Li, P.K. Maini, The foundations of cancer modelling: selected topics, speculations,& perspectives, Math. Mod. Meth. Appl. 253 (2008) 593–646.
[55] A. Arduino, L. Preziosi, A multiphase model of tumour segregation in situ by a heterogeneous extracellular matrix, Int. J. Non-Linear Mech. 75 (2015) 22–30.
[56] L. Preziosi, Cancer Modelling and Simulation, CRC Press, 2003.
[57] Chr. Stinner, Chr. Surulescu, A. Uatay, Global existence for a go-or-grow multiscale model for tumor invasion with therapy, Math. Models Methods Appl. Sci. 26 (11) (2016) 2163–2201.
[58] A. Marciniak-Czochra, T. Stiehl, Mathematical modelling of leukemogenesis and cancer stem cell dynamics, Math. Mod. Nat. Phen. 7 (2012) 166–202.
[59] M.D. Johnston, P.K. Maini, S. Jonathan-Chapman, C.M. Edwards, W.F. Bodmer, On the proportion of cancer stem cells in a tumour, J. Theoret. Biol. 266 (4) (2010) 708–711.
[60] A.R.A. Anderson, M.A.J. Chaplain, E.L. Newman, R.J.C. Steele, A.M. Thompson, Mathematical modelling of tumour invasion and metastasis, Comput. Math. Methods Med. 2 (2) (2000) 129–154.
[61] E.T. Roussos, J.S Condeelis, A. Patsialou, Chemotaxis in cancer, Nat. Rev. Cancer 11 (8) (2011) 573–587.
[62] J.S. Rao, Molecular mechanisms of glioma invasiveness: the role of proteases, Nat. Rev. Cancer 3 (7) (2003) 489–501.
[63] R. Eymard, T. Gallouët, R. Herbin, Finite volume methods, in: Handbook of Numerical Analysis, Vol. 7, Elsevier, 2000, pp. 713–1018.
[64] R. Belmouhoub, Modélisation Tridimensionnelle de La Genèse Des Bassins Sédimentaires, (Ph.D. thesis), Ecole Nationale Supérieure des Mines de Paris, 1996.
[65] Y. Coudière, Ph. Villedieu, Convergence rate of a finite volume scheme for the linear convection-diffusion equation on locally refined meshes, ESAIM Math. Model. Numer. Anal. 34 (6) (2000) 1123–1149.
[66] Japanese Gastric Cancer Association, Japanese classification of gastric carcinoma: 3rd english edition, Gastric Cancer 14 (2) (2011) 101–112.