
Article:
Yang, FW, Goodyer, CE, Hubbard, ME et al. (2017) An Optimally Efficient Technique for the Solution of Systems of Nonlinear Parabolic Partial Differential Equations. Advances in Engineering Software, 103. pp. 65-84. ISSN: 0965-9978

https://doi.org/10.1016/j.advengsoft.2016.06.003

An Optimally Efficient Technique for the Solution of Systems of Nonlinear Parabolic Partial Differential Equations

F.W. Yang*[1], C.E. Goodyer[2], M.E. Hubbard[3] and P.K. Jimack[4]

[1]Department of Mathematics, University of Sussex, Brighton, United Kingdom
[2]ARM, York House, Manchester, United Kingdom
[3]School of Mathematical Sciences, University of Nottingham, United Kingdom
[4]School of Computing, University of Leeds, United Kingdom

June 5, 2016

Abstract

This paper describes a new software tool that has been developed for the efficient solution of systems of linear and nonlinear partial differential equations (PDEs) of parabolic type. Specifically, the software is designed to provide optimal computational performance for multiscale problems, which require highly stable, implicit, time-stepping schemes combined with a parallel implementation of adaptivity in both space and time. By combining these implicit, adaptive discretizations with an optimally efficient nonlinear multigrid solver it is possible to obtain computational solutions to a very high resolution with relatively modest computational resources. The first half of the paper describes the numerical methods that lie behind the software, along with details of their implementation, whilst the second half of the paper illustrates the flexibility and robustness of the tool by applying it to two very different example problems. These represent models of a thin film flow of a spreading viscous droplet and a multi-phase-field model of tumour growth. We conclude with a discussion of the challenges of obtaining highly scalable parallel performance for a software tool that combines both local mesh adaptivity, requiring efficient dynamic load-balancing, and a multigrid solver, requiring careful implementation of coarse grid operations and inter-grid transfer operations in parallel.

Keywords: parallel, adaptive mesh refinement, finite difference, implicit, multigrid, thin film flow, tumour growth.

*Main correspondent: F.W.Yang@sussex.ac.uk

1 Introduction

Many problems in computational engineering and science are based upon the use of complex mathematical models and their numerical approximations. These models often consist of highly nonlinear, time-dependent and coupled PDEs. Accurate, efficient and reliable numerical algorithms (and, frequently, great computational power) are necessary in order to obtain robust computational solutions. This paper describes a new Engineering Software tool that we have developed to exploit advanced numerical methods for the efficient solution of nonlinear time-dependent systems of PDEs. Specifically, the focus of this software is nonlinear, and potentially stiff, parabolic systems. This type of system may be used to represent a plethora of different applications, ranging from solidification [4] and computational fluid dynamics [2, 5] to tumour growth [3].

The multigrid method is commonly accepted as being one of the fastest numerical methods for solving algebraic equations arising from mesh-based discretizations of PDEs. Brandt in his 1977 paper [6] systematically describes the first multigrid methods, and some of their applications. Subsequent publications, e.g. [15, 8], suggest further combinations of multigrid methods with spatial adaptivity and adaptive time-stepping, for applications in which physical effects occur at multiple length and time scales. In recent years a number of general-purpose software packages have been developed to provide adaptive multigrid solvers for broad classes of PDEs. Noteworthy examples include DEAL.II [9] and DUNE [10]. The software that we introduce in this paper is intended to complement such general-purpose packages by providing a tool that we have developed specifically for the solution of multiscale parabolic PDE systems using fully-implicit time stepping. Typically these problems are highly stiff, thus requiring strongly stable temporal discretization, which leads to large systems of (generally nonlinear) algebraic equations to be solved at each time step. Our solver is written specifically with such systems in mind, and exploits *nonlinear* geometric multigrid methods in order to advance in time with optimal efficiency.

The particular nonlinear multigrid scheme that we use is based upon a combination of the multilevel adaptive technique (MLAT) and the full approximation scheme (FAS), first introduced in [15] and [6] respectively. These are implemented in a parallel setting based upon distributed memory parallelism using the MPI library, and this paper describes our new software framework for the first time. In addition to this overview of the software, we also include a thorough evaluation of our implementation and the effectiveness of our chosen computational techniques. This is acheived using two example applications which are based upon (i) the thin film flow of a spreading droplet [2], for which we demonstrate the temporal and spatial adaptivity in detail; and (ii) a multi-phase-field model of tumour growth, which was also discussed in [11].

In Section 2, we introduce our software framework in detail, whilst in Section 3 we we present detailed results from the two selected applications that are provided for validation purposes. We conclude this paper in Section 4 with suggestions for possible future work.

2 Software Framework

In this section, we provide an introduction to our software with its key features. This is followed by a high level overview of the programme flow and then enhanced details of the the main components of our software tool. Afterwards, various implementation issues are explained, and the section ends with a description of the user's control of the software.

2.1 Introduction

The purpose of this paper is to describe a new software tool, Campfire, which we have developed in order to efficiently obtain solutions to general systems of nonlinear parabolic PDEs. The core of the software is the use of nonlinear multigrid methods to solve the algebraic systems arising from implicit temporal discretization schemes. The user is free to select their own spatial discretization scheme based upon cell-centred quadrilateral or hexahedral elements (i.e. the degrees of freedom are stored at the cell centres). Finite difference, low order discontinuous Galerkin and finite volume methods are typical examples that may provide such cell-centred discretizations, though we make use only of the former in all of the examples presented in this paper.

Distinctively, our nonlinear multigrid solver has optimal, linear, computational complexity for general systems of parabolic PDEs (illustrated in Section 3). The software is able to include spatial and temporal adaptivity, and parallel computing is implemented through geometric domain decomposition. This combination of techniques gives a huge boost to the efficiency, thus allowing complex, time-dependent systems to be solved in 3-D within reasonable and practical time.

Campfire is designed to be flexible and efficient. It only requires the user to supply the fully-discretized system (in the form of a residual function), the initial and boundary conditions of the model and prescribed parameter values. Most of the functionalities within this software can be easily adjusted by altering these parameters, though robust default values for general cases are also provided.

For clarity, we summarise the key features of our software:

- it is able to carry out the computation in a parallel environment where the parallelization comes from mesh partitions via domain decomposition;
- it is able to generate a distributed mesh hierarchy using an open source library (i.e. PARAMESH [1]), with appropriately modified mesh data structures and dynamic load-balancing procedures, tuned for enhanced parallel multigrid performance;
- it is able to dynamically adapt the spatial mesh in a hierarchical manner based upon flexible error control criteria;
- it is able to apply highly stable, fully-implicit, time stepping with step-size selection based upon flexible error control criteria (and not generally dependent

upon numerical stability constraints);

- it is able to solve the nonlinear algebraic systems arising from the adaptive spatial and temporal discretizations using nonlinear multigrid methods, [6, 15];
- it is able to store checkpoint files for possible restarts in parallel environments via HDF5;
- it is able to output solutions into standard formats (e.g. CSV and VTK) for common visualization tools such as Paraview [12].

This paper is the first description of the software tool itself, however, earlier versions of the multigrid solver, that are now part of the software, have been used in [4, 13] to solve specified problems in the solidification of metallic alloys.

2.2 Overview

In Campfire the user is able to define their own spatial discretization based upon cell-centre values in the hexahedral mesh (in the 3-D case). Combining this with an implicit temporal discretization leads to a system of algebraic equations for which there are unknown values at the centre of each hexahedron (one unknown for each dependent variable) at the end of each time step. Throughout this section we represent this system of equations as either

$$\mathcal{F}(u) = 0 \qquad \text{or} \qquad \mathcal{A}(u) = f \ . \tag{1}$$

Here u stores all of the unknown values to be determined at the end of the time step, whilst $\mathcal{F}$ and $\mathcal{A}$ are nonlinear functions related by $\mathcal{F}(u) = \mathcal{A}(u) - f$ for some known vector f.

In order to commence a new simulation in Campfire it is necessary to provide a set of parameters that define the problem and control the mesh adaptivity (see subsection 2.4.2), and a set of initial conditions for each dependent variable (see subsection 2.4.1). These are used to initialise the software (including memory allocation), to allocate an initial spatial mesh (with the support of the PARAMESH library) and to assign the initial state of the necessary variables (see subsection 2.4.1). This initialization process is indicated by the "New job" branch in Figure 1. Further details of the mesh data structure and the key parameters, as well as how to define the discrete PDE system, are provided in the next section: the purpose of Figure 1 is to describe how Campfire operates at a high level, and how the key components are linked together. Note that it is also possible to re-start a previous run from a checkpoint file, as indicated by the "Restart" branch in the same figure. The HDF5 checkpoint file allows all previous mesh and solver parameters to be picked up and the previous run to be continued from that point in time onwards.

As noted in Figure 1, the key loop within Campfire occurs at each implicit time step. Within each such step there are three main components: adaptive mesh refinement

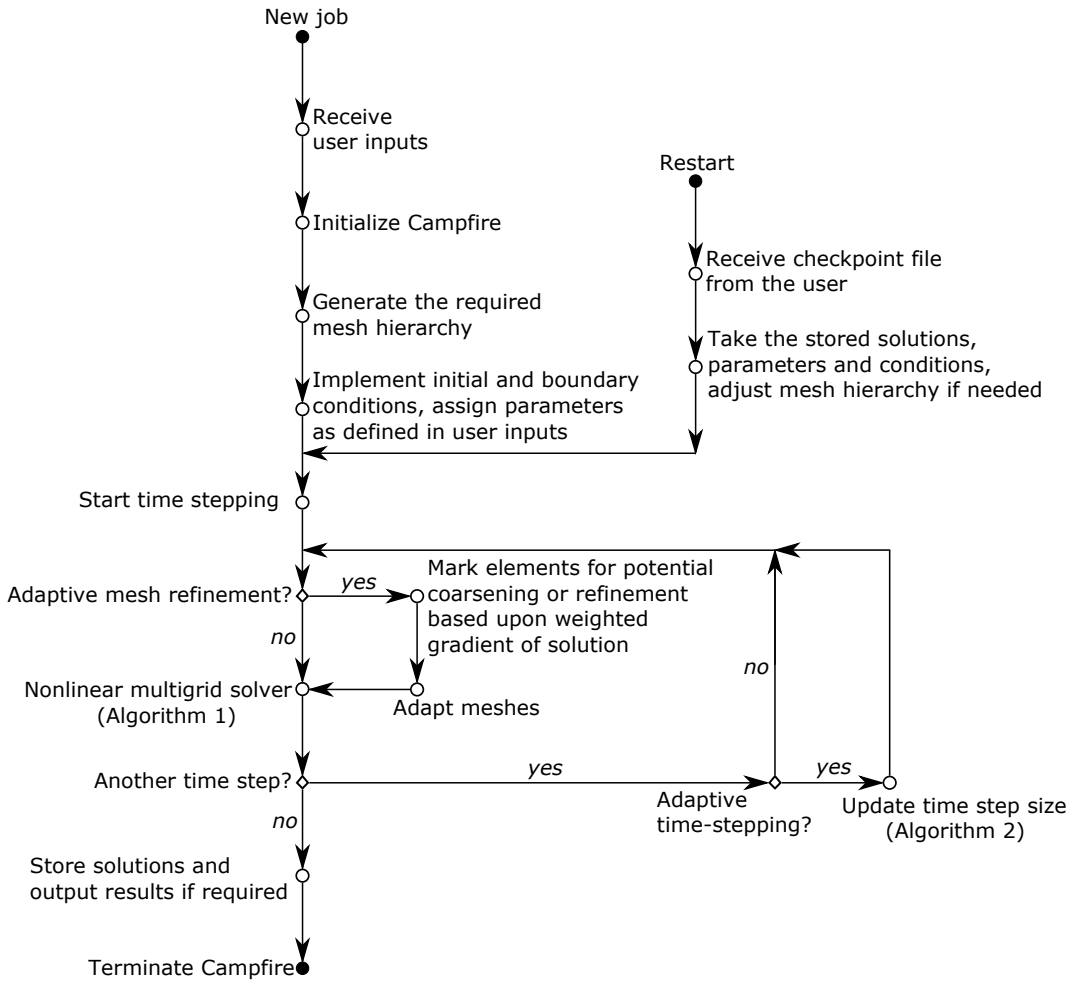

Figure 1: This flow chart illustrates how Campfire operates and how each operation is connected.

(AMR) based upon the latest solution values; the use of the nonlinear multigrid solver for the nonlinear algebraic system of equations, of the form (1), that arises at the current time step; and the selection of the step size for the next time step (adaptive time-stepping). We now describe these three components in further detail.

The AMR algorithm is fairly standard and is described in full in [1]. As explained in Section 2.3.1, it is based upon hierarchical refinement through the use of a quad-tree or an oct-tree (in two or three dimensions respectively) data structure. Each node of the tree is a block of uniform mesh (e.g. $16 \times 16 \times 16$ hexahedral elements) which may be refined into four or eight child blocks (in two or three dimensions). The algorithm is divided into two phases: the first decides which blocks should be refined or coarsened (based upon either a default error indicator or a user-supplied error indicator, along with some constraints on the mesh topology (e.g. neighbouring blocks cannot differ by more than one level of refinement)); whilst the second phase actually implements the refinement. For this second phase, not only is the tree data structure updated to reflect blocks which are refined or coarsened (for coarsening, it is effectively the inverse of a refinement operation so, in three dimensions, eight child blocks would be removed from the tree and the memory freed), but a dynamic load-balancing routine is then invoked so as to ensure that the mesh tree is equally partitioned across the available processes. In the default version of PARAMESH the tree is partitioned based upon the partition of a depth-first ordering (referred to as Morton ordering), however this is very unsuitable for multigrid solvers since the finest mesh level (where most computational work takes place (see below)) will not generally be split equally amongst the processes. Hence we use a dynamic load-balancing strategy which partitions all of the blocks that lie at the same depth of the tree independently: each of these partitions aims to provide an equal number of blocks per process, and neighbouring blocks allocated to the same process where possible (so as to minimize inter-process communication).

As already noted above, the algebraic solver required at each time step uses a nonlinear multigrid method based upon MLAT and FAS, [15, 6]. A single iteration of this scheme is described in detail in Algorithm 1. Note that the nonlinear multigrid approach differs fundamentally from linear multigrid since the coarse grid correction phase is not based upon the approximate solution of an error equation, as in the linear case [7, 8]. Instead, the coarse grid correction in nonlinear multigrid is obtained by solving an approximation to the original system on a coarser grid, but with a modified right-hand side (RHS) term which comes from adding the difference between the residual of the restricted solution and the restriction of the fine grid residual, [7, 8, 6], as shown in steps 5 and 6 of the algorithm. Otherwise the nonlinear algorithm follows a similar pattern to standard multigrid: a typical V-cycle (as shown in Algorithm 1) requires a small number of sweeps of an iterative smoother (step 1), a residual calculation (step 2), a restriction to the coarser grid (step 3), a coarse grid correction (steps 4 to 10) and further sweeps of the smoother (step 10). Indeed, when this algorithm is applied to a linear problem it may be shown to be equivalent to a standard linear multigrid V-cycle. In order to establish convergence (or otherwise) of the multigrid iterations the norm of the residual is monitored after each V-cycle: once it is reduced

below a prescribed absolute value or a prescribed relative reduction from the initial residual, convergence is assumed (and if neither criterion is met after a maximum number of iterations the time step is repeated with a smaller step size). Further details of the smoother, the grid transfer operators (I_h^{2h} and I_{2h}^h) and the default parameters used for each V-cycle are given in the following sections.

Algorithm 1 V-cycle MLAT nonlinear FAS multigrid method

The superscripts h and $2h$ denote fine and coarse grid (Ω_h and Ω_{2h}) values respectively.

Function: u^h = V-cycleMLATMG$(h, u^h, u^{2h}, f^h, f^{2h}, A^h(u^h), A^{2h}(u^{2h}))$

1. Apply p_1 iterations of the pre-smoother on $A^h(u^h) = f^h$
$u^h = \text{PRE-SMOOTH}(p_1, u^h, A^h(u^h), f^h)$

2. Compute the residual r^h on Ω_h
$r^h = f^h - A^h(u^h)$

3. Restrict the residual r^h from Ω_h to $\Omega_{2h} \cap \Omega_h$ to obtain r^{2h}
$r^{2h} = I_h^{2h} r^h$

4. Restrict the fine grid approximate solution u^h from Ω_h to Ω_{2h} to obtain w^{2h}

$$w^{2h} = \begin{cases} I_h^{2h} u^h & \text{on } \Omega_{2h} \cap \Omega_h \\ u^{2h} & \text{on the remaining part of } \Omega_{2h} \end{cases}$$

5. Compute the modified RHS

$$f^{2h} = \begin{cases} r^{2h} + A^{2h}(w^{2h}) & \text{on } \Omega_{2h} \cap \Omega_h \\ f^{2h} & \text{on the remaining part of } \Omega_{2h} \end{cases}$$

6. if Ω_{2h} = coarsest grid **then**
Perform an "exact" coarsest grid solve on $A^{2h}(u^{2h}) = f^{2h}$
else
u^{2h} = V-cycleMLATMG$(2h, w^{2h}, u^{4h}, f^{2h}, f^{4h}, A^{2h}(u^{2h}), A^{4h}(u^{4h}))$
end if

7. Compute the error approximation e^{2h} on $\Omega_{2h} \cap \Omega_h$
$e^{2h} = u^{2h} - w^{2h}$

8. Update solution on the remaining part of Ω_{2h}
$u^{2h} = u^{2h}$ latest

9. Interpolate the error approximation e^{2h} from Ω_{2h} to Ω_h to obtain e^h
$e^h = I_{2h}^h e^{2h}$

10. Perform correction
$u^h = u^h + e^h$

11. Apply p_2 iterations of the post-smoother on $A^h(u^h) = f^h$
$u^h = \text{POST-SMOOTH}(p_2, u^h, A^h(u^h), f^h)$

Finally, we describe the adaptive time-stepping procedure. There are two main options for controlling the adaptive time-step selection in Campfire: either selecting δt based upon the rate of convergence of the multigrid solver at the previous time step; or selecting δt based upon an estimate of the local error per unit step for the BDF2 scheme [17] over the previous time step (as in [18] for example). We illustrate the

former in Algorithm 2 since this is the default setting in Campfire, and is independent of the time-stepping scheme being used. As may be seen from Algorithm 2, when a converged solution is obtained at the previous time step in a low number of multigrid V-cycles the time step size will be increased (subject to a maximum value which can be set by the user). Otherwise, if convergence is not achieved at the previous time step in a prescribed maximum number of V-cycles then the step size is reduced by 25% and the previous step is *retaken*. Alternatively, if convergence was achieved at the previous time step, but at a slower rate than targeted (i.e. a high number of V-cycles were taken), then the step size is reduced by 10% for the next time step. After the algebraic system is solved or the maximum number of V-cycles is reached (i.e. slow convergence or non-convergence respectively), the number of V-cycles and the norm of the residual can be assessed. If none of the above occurred then convergence was neither too slow nor excessively fast and so δt is left unchanged for the next step. All of the parameter values for measurements in the if-statements, as well as the different ratios of changes to δt, may be altered by the user. Note we use the superscripts $\tau + 1$ and τ to indicate the next time step size and the current step size, respectively.

Algorithm 2 Adaptive time stepping in Campfire

1. Input: No.V-cycles – the number of V-cycles that are used by the multigrid solver at this time step
2. Input : r – residual
3. **if** No.V-cycles is low **and** r is acceptable **then**
4. $\quad \delta t^{\tau+1} = \frac{10}{9}\delta t^{\tau}$
5. $\quad$ **if** $\delta t^{\tau+1} \geq$ max-δt-allowed **then**
6. $\quad\quad \delta t^{\tau+1} =$ max-δt-allowed
7. $\quad$ **end if**
8. **else if** No.V-cycles reaches the maximum number **or** r is not acceptable **then**
9. $\quad$ Recompute the current time step τ with $\delta t^{\tau} = \frac{3}{4}\delta t^{\tau}$
10. **else if** No.V-cycles is high **then**
11. $\quad \delta t^{\tau+1} = \frac{9}{10}\delta t^{\tau}$
12. **end if**

The above descriptions summarise the general computational flow within Campfire as well as its essential components. In the next section, we provide details of some of the key implementation issues.

2.3 Key Algorithmic Details

In the previous section we provided a brief introduction to AMR and associated procedures, such as dynamic load-balancing, and to FAS nonlinear multigrid, with its associated components. In this section, we focus on two key algorithmic details. The first is the fundamental mesh structure used in Campfire which is described in Subsection 2.3.1. The other concerns the key components of the nonlinear multigrid solver, and we illustrate these in Subsection 2.3.2.

2.3.1 Parallel Adaptive Mesh Refinement

One of the fundamental building blocks in Campfire is its mesh structure, which is inherited from the open source software library PARAMESH [1]. This structure is based upon a dynamic tree of Cartesian mesh blocks of a fixed size (a quad-tree in 2-D or an oct-tree in 3-D). To illustrate, we use an simple 2-D adaptive grid with a coarse block size as an example. This example is shown in Figure 2 (a), with a user-defined block size of 2×2 (note this size choice is for the purpose of demonstration: in practice, sizes like 8×8 or 16×16 (or $16 \times 16 \times 16$ in 3-D) are usually used). As shown in this figure, despite different refinement levels, all mesh blocks have this selected size of 2×2. We include a "telescope" view of the example adaptive grid in Figure 2 (e) where we use solid dots to represent the cell-centred grid points. In (a) and (e), the heavy lines indicate the boundaries of each block, and the lighter lines indicate individual cells.

Parallelism is achieved through distributing mesh blocks to multiple MPI processes, and using MPI to communicate between individual MPI processes to exchange data. An important concept is that of guard cells. Each mesh block is surrounded by a layer of guard cells, which may be expanded to multiple layers for schemes with larger stencils. We illustrate this concept using the four mesh blocks on the finest level of our example. This is shown in Figure 2 (d), where the guard cells are marked using dashed lines and the guard cell centres are indicated by $\circ$. The guard cells at the actual domain boundary contain information which allows the specified boundary conditions to be implemented, and others are used to store values of corresponding grid points on the neighbouring blocks (the linked curves show the corresponding grid points between neighbours). Thus the parallel communication is used to update the data held by these guard cells.

The quad-tree structure corresponding to the mesh illustrated in (a) and (e) is shown in Figure 2 (b). We use four shapes (i.e. $\blacktriangle$, $\circ$, $\square$ and $\bullet$) to indicate a possible partition across four MPI processes in a parallel environment. Note this is not the original, so-called Morton ordering used in PARAMESH, but the modified version from Campfire. For completeness, the Morton ordering (if applied to this example) is shown in Figure 2 (c). From a multigrid point of view, the parallel distribution in (a) is superior since the work is partitioned equally at each level, most importantly the finest level.

2.3.2 Nonlinear Multigrid

There are two key components to the nonlinear multigrid solver described in the previous section: steps 1 and 11 of Algorithm 1 require an iterative smoother, whilst data transfer operators are needed to perform the restriction and the interpolation in steps 3, 4 and 9.

First of all, we describe our choice of the multigrid smoother. From our experience, for systems that have multiple dependent variables, it is better to update all these dependent variables simultaneously at each visited grid point. We refer to this as a point-wise, nonlinear *block* Jacobi method [16].

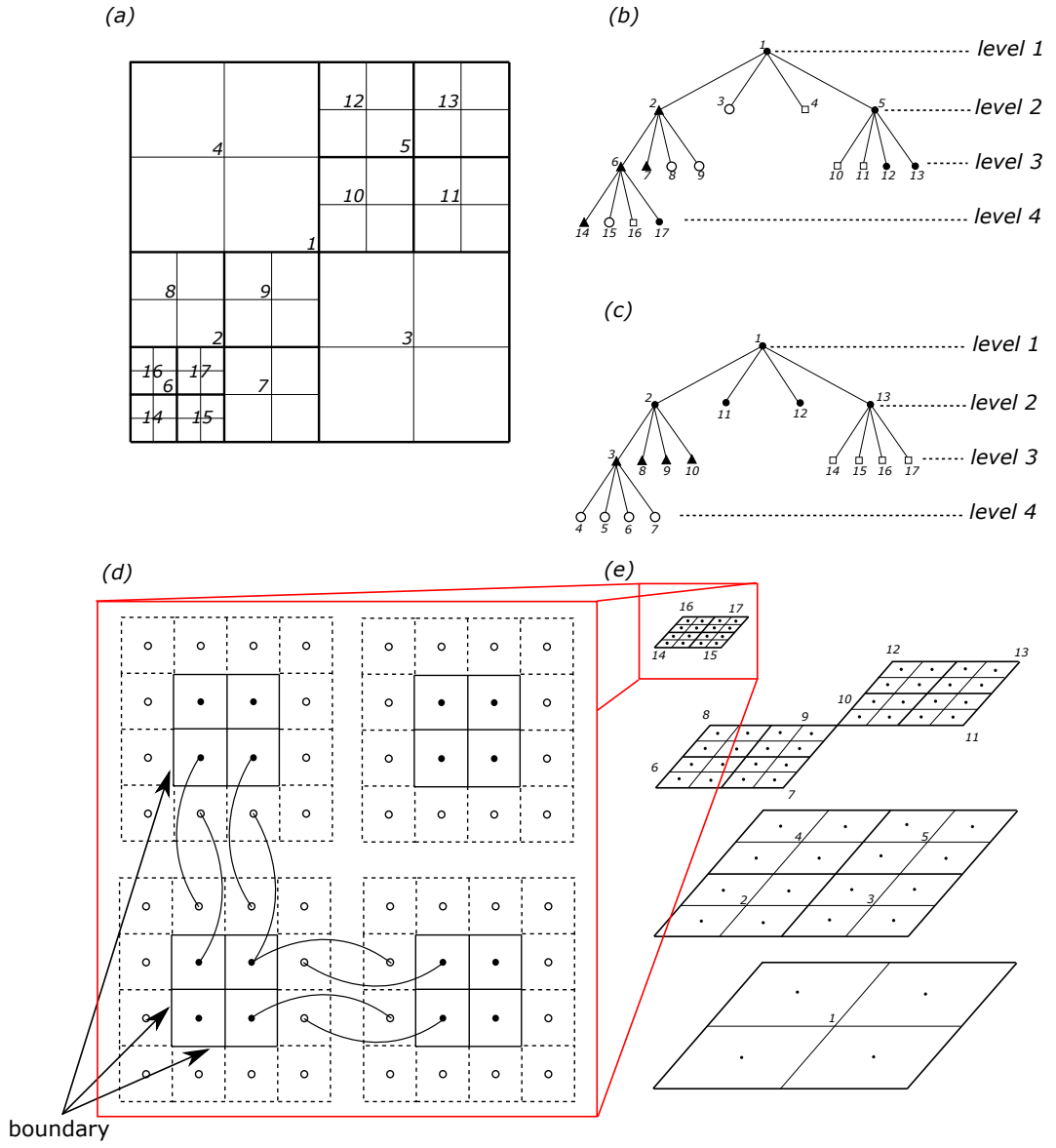

Figure 2: (a) A 2-D adaptive Cartesian mesh. (b) The corresponding quad-tree data structure of the mesh in (a). (c) For comparison, the parallel distribution based upon a depth-first ordering. (d) An illustration of the role of guard cells. (e) A "telescope" view of the mesh hierarchy.

Let us consider a cell-centred finite difference discretization at each implicit time step: $\mathcal{F}(u) = 0$, where u is a vector containing all unknown values at the end of the step. Let $u_{i,k}$ be the approximate solution on grid point i for unknown variable k, where we assume $\mathcal{K}$ unknowns at each grid point. The system $\mathcal{F}(u) = 0$ is made up of $N \times \mathcal{K}$ coupled nonlinear algebraic equations (where N is the number of internal, cell-centred grid points), each of the form

$$\mathcal{F}_{i,k}(u) = 0, \tag{2}$$

where $i = 1, \ldots, N$ and $k = 1, \ldots, \mathcal{K}$. To clarify the notation $u_{i,k}$ is the k^{th} component of $u_i \in \mathbb{R}^{\mathcal{K}}$ and $\mathcal{F}_{i,k}$ is the k^{th} component of $\mathcal{F}_i \in \mathbb{R}^{\mathcal{K}}$. On one grid point i, all $\mathcal{K}$ variables may be updated simultaneously as

$$u_i^{\ell+1} = u_i^{\ell} - C_i^{-1}\mathcal{F}_i(u^{\ell}), \tag{3}$$

where ℓ is the iteration index and C_i^{-1} is the inverse of the $\mathcal{K} \times \mathcal{K}$ local Jacobian matrix C_i, which is given as

$$C_i = \begin{pmatrix} \frac{\partial \mathcal{F}_{i,1}}{\partial u_{i,1}} & \frac{\partial \mathcal{F}_{i,1}}{\partial u_{i,2}} & \cdots & \frac{\partial \mathcal{F}_{i,1}}{\partial u_{i,\mathcal{K}}} \\ \frac{\partial \mathcal{F}_{i,2}}{\partial u_{i,1}} & \frac{\partial \mathcal{F}_{i,2}}{\partial u_{i,2}} & \cdots & \frac{\partial \mathcal{F}_{i,2}}{\partial u_{i,\mathcal{K}}} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial \mathcal{F}_{i,\mathcal{K}}}{\partial u_{i,1}} & \frac{\partial \mathcal{F}_{i,\mathcal{K}}}{\partial u_{i,2}} & \cdots & \frac{\partial \mathcal{F}_{i,\mathcal{K}}}{\partial u_{i,\mathcal{K}}} \end{pmatrix}. \tag{4}$$

This iterative method is also used as the iterative solver on the coarsest grid (i.e. at step 6 of Algorithm 1), by sweeping through the grid multiple times.

The other key multigrid component introduced in the previous section is the grid transfer operators, used to move data between grid levels. Considering we are using cell-centred grids, a cell-averaging restriction and a bilinear interpolation are employed [8]. We illustrate these two operators on a simple 2-D cell-centred grid in Figure 3. A 2-D version of the restriction can be written as

$$\begin{aligned} I_h^{2h} u_h(x,y) = \\ \frac{1}{4}\Bigg[& u_h\left(x - \frac{h}{2}, y - \frac{h}{2}\right) + u_h\left(x - \frac{h}{2}, y + \frac{h}{2}\right) \\ & u_h\left(x + \frac{h}{2}, y - \frac{h}{2}\right) + u_h\left(x + \frac{h}{2}, y + \frac{h}{2}\right) \Bigg], \end{aligned} \tag{5}$$

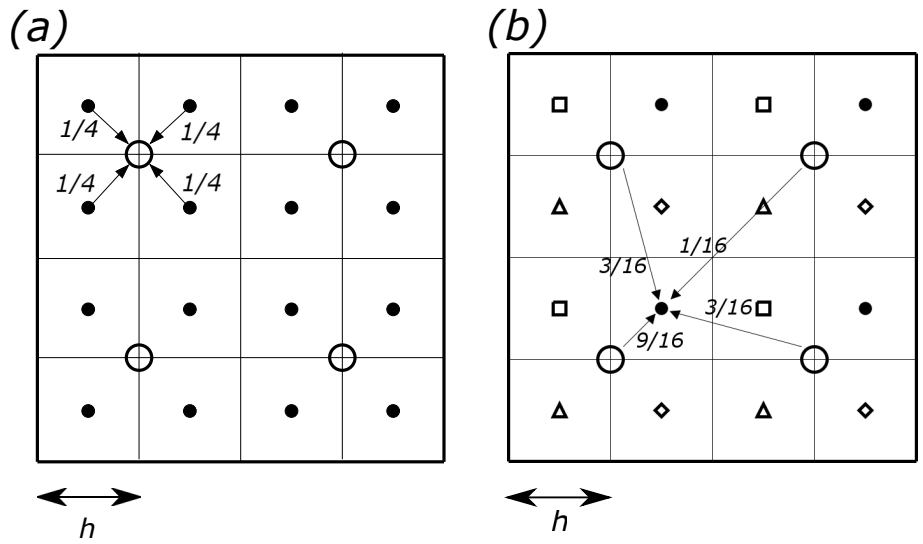

Figure 3: (a) The cell-averaging restriction operator in Equation (5): arrows indicate an example of this process from points (marked as •) on the fine mesh level to a point (marked as ○) on the coarse mesh level. (b) The bilinear interpolation operator in Equation (6): arrows indicate an example of this process from points (marked as ○) on the coarse mesh level to a point (marked as •) on the fine mesh level.

and a 2-D version of the interpolation is given as

$$I_{2h}^{h}u_{2h}(x,y) = \begin{cases} \frac{1}{16}\Big[9u_{2h}\left(x-\frac{h}{2},y-\frac{h}{2}\right)+3u_{2h}\left(x-\frac{h}{2},y+\frac{3h}{2}\right) & \\ +3u_{2h}\left(x+\frac{3h}{2},y-\frac{h}{2}\right)+u_{2h}\left(x+\frac{3h}{2},y+\frac{3h}{2}\right)\Big] & \text{for } \bullet; \\ \frac{1}{16}\Big[3u_{2h}\left(x-\frac{3h}{2},y-\frac{h}{2}\right)+u_{2h}\left(x-\frac{3h}{2},y+\frac{3h}{2}\right) & \\ +9u_{2h}\left(x+\frac{h}{2},y-\frac{h}{2}\right)+3u_{2h}\left(x+\frac{h}{2},y+\frac{3h}{2}\right)\Big] & \text{for } \square; \\ \frac{1}{16}\Big[3u_{2h}\left(x-\frac{h}{2},y-\frac{3h}{2}\right)+9u_{2h}\left(x-\frac{h}{2},y+\frac{h}{2}\right) & \\ +u_{2h}\left(x+\frac{3h}{2},y-\frac{3h}{2}\right)+3u_{2h}\left(x+\frac{3h}{2},y+\frac{h}{2}\right)\Big] & \text{for } \diamond; \\ \frac{1}{16}\Big[u_{2h}\left(x-\frac{3h}{2},y-\frac{3h}{2}\right)+3u_{2h}\left(x-\frac{3h}{2},y+\frac{h}{2}\right) & \\ +3u_{2h}\left(x+\frac{h}{2},y-\frac{3h}{2}\right)+9u_{2h}\left(x+\frac{h}{2},y+\frac{h}{2}\right)\Big] & \text{for } \triangle, \end{cases} \tag{6}$$

where the array u stores values at the grid points, (x, y) are the Cartesian coordinates, $h, 2h$ are the grid spacings on fine and coarse grids respectively and the geometric symbols are indicated in Figure 3 (b). The 3-D versions of these two operators (i.e. 3-D cell averaging and trilinear interpolation) are straightforward extensions.

2.4 User Experience

In this section, we provide an overview of how a user of the software can define the problem being solved and control the way in which the solver progresses. As indicated above, the key user-defined functions are the residual function, which allows the discretized PDE system to be defined (see Equation (2)), and the local Jacobian matrix which contains the derivative terms shown in Equation (4). In the first subsection below we describe the key data structure that must be used in order to define these. In the second subsection, we then provide an overview of the key parameters that may be selected or defined by the user in order to control the model, the domain, the multigrid solver, etc.

2.4.1 User-Defined Functions

Following the approach of PARAMESH [1], within Campfire the key data structure is the so-called "unk" array. This is a multi-dimensional data structure that is synchronised to the mesh blocks, making it highly parallelizable. This data structure is also suitable for multiple dependent variables. It is given by

$$\text{unk}(var, i, j, k, lb), \tag{7}$$

where var is described below, i, j, k represent each grid point (i.e. cell) in the current mesh block and lb denotes the mesh block. If it is in 2-D, k stays as 1.

In Campfire's default option, there are five arrays associated with each dependent variable. For example, the first variable has the following data structure:

- $\text{unk}(1, i, j, k, lb)$, stores the latest solution (or the initial condition);
- $\text{unk}(2, i, j, k, lb)$, stores the modified RHS (or is zero on the finest grid);
- $\text{unk}(3, i, j, k, lb)$, stores the computed residual value (shown in Equation (2));
- $\text{unk}(4, i, j, k, lb)$, stores the solution from the previous time step;
- $\text{unk}(5, i, j, k, lb)$, stores the solution from the one before previous time step.

For a system of PDEs, the second dependent variable then starts with $\text{unk}(6, i, j, k, lb)$ through to $\text{unk}(10, i, j, k, lb)$. For example, the thin film model presented in Section 3.1 has two dependent variables, thus 10 "unk" arrays. Furthermore, the tumour growth model presented in Section 3.2 has five dependent variables, hence 25 "unk" arrays are employed. It is worth noting that the i, j, k indices also include the guard cells on the current mesh block. Note that $\text{unk}(5, ., ., ., .)$ is only used when a multi-step discretization is employed in time, such as the adaptive BDF2 scheme [17].

For each variable, the corresponding 3^{rd} array of the "unk" arrays is assigned by a user-supplied subroutine which is used to define the residual of the discrete system. Specification of this subroutine is the main programming task that must be undertaken once the user has selected their spatial and temporal discretizations. This subroutine is used by the smoother, as well as to compute the residual required in step 2 of Algorithm 1.

Another user-defined term is the initial condition, which must be assigned to the corresponding 1^{st} array of the "unk" arrays initially. Later on the data in these arrays will be replaced by the most recent solutions for each variable in the system.

The final user-defined subroutine is required to compute the local Jacobian matrix which contains the derivative terms (see Equation (4)). This matrix is stored in an $N \times N$ array where N is the number of coupled equations. Each entity specified in Equation (4) must be assigned by the user and, at each grid-point visit in the smoother, this local matrix is re-computed using the corresponding information related to the current grid point.

2.4.2 Software Parameters

In total there are seven categories of user-controlled parameters. In the following paragraphs we provide a high level summary of the key ones, with descriptions and some default values.

1. Mesh setup: first of all, to set up the mesh hierarchy, a mesh block size is to be defined: *nxb*, *nyb* and *nzb* specify the number of cell-centred grid points in each axis direction. It is worth noting there is a trade-off, since a smaller size will create too many guard cells relative to the block size, hence burdening the memory and parallel communication, however a larger size will deteriorate the flexibility of the dynamic load balancing and the adaptive mesh refinement. Referring back to the example used in Figure 2, for clarity, we illustrated using a 2×2 block size. However, by default, Campfire suggests a size of 8×8 for simulations which would have up to 4 million grid points at the finest level if we were to measure the grid as uniform (i.e. up to 2048×2048). For larger simulations, especially in 3-D, we suggest to use 16×16 in 2-D and $16 \times 16 \times 16$ in 3-D as the block size.

Related parameters include: *nguard* which defines the number of layers (depends on the choice of discretization stencils) of guard cells and is initially set to be 1; *maxblocks*, which is the maximum mesh blocks allowed for each MPI process; *nBx*, *nBy* and *nBz* which set the number of mesh blocks on the coarsest level for each axis direction, and generally are set to be 1; *lrefine_max* governs the maximum number of levels in the mesh hierarchy.

2. Problem setup: we begin with defining the domain size by adjusting *grid_min* and *grid_max*. For example, *grid_min* $= 0$ and *grid_max* $= 1$ defines a square domain with size $[0, 1] \times [0, 1]$, or a cube domain with size $[0, 1] \times [0, 1] \times [0, 1]$. It is possible to define domains that each axis has a different size. In addition, *total_vars* is the number of dependent variables, *dt* is the initial time step size and *simulation_time* is the ending time T.

3. Multigrid setup: p_1 and p_2 are the number of iterations of the smoother (see Algorithm 1) to be carried out on each grid and they generally take values from the

range of 1 to 4; *solve_count_max* is the maximum number of iterations of the smoother to be carried out as a solver on the coarsest grid; *max_v_cycle_count* is the maximum number of V-cycles allowed in each time step (see step 8 in Algorithm 2); *mg_min_lvl* is the coarsest level for the multigrid solver, in case the root level is not desired. Furthermore, *lrefine_max* defines the finest grid of the solver, *absolute_tol* is the absolute tolerance for the stopping criterion and *relative_tol* is the relative tolerance for the stopping criterion. If the infinity norm of the residual ($||r||_\infty$) drops by this amount (comparing against the norm from the first V-cycle of this time step), or it falls below the absolute tolerance, then we consider the solution to be converged at that time step.

4. **Output setup:** there are two parameters for output, *verbose* takes an integer value between 1 and 4 for the amount of terminal output, where 4 is the most detailed output and is for debugging; *output_rate* defines the number of time steps between generation of checkpoint files.

5. **Model-specific:** model-specific parameters can be included in a file to allow all of the model-related components to be grouped together. Tables 1 and 7, in Section 3, show the parameters required by the thin film model and the tumour growth model, respectively.

6. **AMR setup:** *local_adaptation*, which takes 0 or 1 as the switch for this functionality; and *ctore* and *ctode* which are the thresholds for mesh refinement and coarsening, respectively.

7. **Adaptive time-stepping setup:** parameters described here control the adaptive time-stepping, *adaptive_TS* is the 0 or 1 switch (i.e. a fixed time step size is used when *adaptive_TS* $= 0$); *low_vcycle_count* (see step 3 in Algorithm 2) and *high_vcycle_count* (see step 10 in Algorithm 2) are two integer values used to indicate whether we should increase *dt* for efficiency or reduce it for accuracy/stability, based upon the rate of convergence of multigrid at the end of each time step.

Having provided a brief overview of the software that we have developed, in the following section we validate this software by demonstrating its successful application to two very different examples, for which we are able to contrast our results with others published elsewhere, [2, 3].

3 Applications

In this section, two illustrative applications are solved using our software framework. The first is a thin film flow model of droplet spreading from [2], whilst the second is a multi-phase-field model of tumour growth from [3]. Note that we have chosen to follow notation which is consistent with the papers [2] and [3] from which our models

are taken. Hence, there are some minor notational differences, however we clearly state every variable and notation within each subsection. For example, h is generally used as the distance between two adjacent grid points, but in Section 3.1, it means the height of the thin film, and dx is employed instead to represent the distance between grid points in the simulations.

The thin film model, described in Section 3.1, is used to validate the software against existing numerical results, to demonstrate optimal convergence of the nonlinear multigrid interation and second order convergence of the numerical results in space and time, and to show the improvements in efficiency obtained by applying adaptivity in space and time. In Section 3.2, a tumour growth model is used to provide further evidence that optimal multigrid convergence and second order convergence of the numerical results are retained when adaptivity mesh refinement is applied, then it is used to highlight parallel performance issues for a larger, three-dimensional, test case. For simplicity, in each of our simulations we consider a domain which, in each coordinate direction, is the same length and divided into the same number of cells (N), so dx is the same in each direction.

Note all the computations were carried out on the HPC service provided at the University of Leeds. Each compute node consists of two 8-core Intel E5-2670 2.6GHz processors with 16GB of shared memory per processor (i.e. 2GB per core). The computational nodes are connected with "Infiniband" interconnects.

3.1 A Thin Film Flow Model of Droplet Spreading

The physical phenomenon of a liquid droplet spreading on a substrate has been studied in many scientific fields. In each case, a common demand is to obtain a relatively accurate numerical model for which the solution represents a good approximation to real-world experiments [19]. Many such models are suggested in the review paper [20]. The model presented here is very close to the work of Schwartz, Bertozzi and their co-workers in [21, 22, 23, 24], and is derived from the Navier-Stokes equations through the use of the lubrication approximation. The precise model, based upon a precursor film, has been previously described by Gaskell et al. in [2] using a vertex-centred finite difference approximation, and solving the resulting system using the FAS nonlinear multigrid with uniform grids. Here we present solutions that are generated by using our parallel, adaptive multigrid solver in two space dimensions (but representing a 3-dimensional flow of a thin film) and compare them with the results in [2].

Following the lubrication approximation, the resulting non-dimensional model consists of two dependent variables: $h(x, y, t)$ and $p(x, y, t)$. The former measures the droplet thickness, and the latter represents the pressure field of the droplet. The Reynolds equation for droplet spreading is given as

$$\frac{\partial h}{\partial t} = \frac{\partial}{\partial x}\left[\frac{h^3}{3}\left(\frac{\partial p}{\partial x} - \frac{B_o}{\epsilon}\sin\alpha\right)\right] + \frac{\partial}{\partial y}\left[\frac{h^3}{3}\left(\frac{\partial p}{\partial y}\right)\right], \tag{8}$$

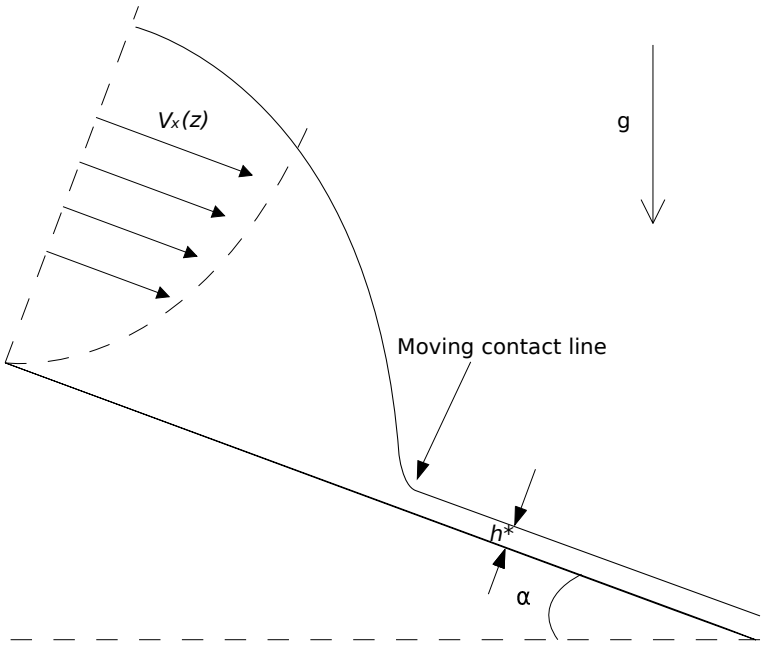

Figure 4: Sketch of precursor film model on an inclined substrate at angle α to the horizontal and the parabolic velocity $v_x(z)$ in the droplet liquid. Note that h^* represents a true thin film ahead of droplet, and the velocity is zero at the substrate.

where B_o is the non-dimensional Bond number, measuring the relative importance of gravitational force relative to surface tension [22], ϵ is the ratio between the characteristic droplet thickness and the extent its footprint on the substrate, and is assumed to be small by the lubrication theory. The main computational challenge is to accurately capture the moving contact line between the thin film liquid and the solid substrate (see Figure 4). Equation (8) is based on the assumption of a no-slip condition at the substrate, however a non-zero velocity is required at the moving contact line (the interface between the air, the drop and the substrate) in order to permit spreading. Figure 4 shows how this "paradox" may be resolved, illustrating a cross-section of the droplet on a substrate inclined at an angle α to the horizontal, as well as a precursor film of thickness h^* to overcome the no-slip condition at the moving contact line. The physical phenomenon of a thin precursor film has been detected in the real-world experiments presented in [25, 26].

The pressure field $p(x, y, t)$ appears in Equation (8) but also needs to be determined. In this model the associated pressure equation is as follows:

$$p = -\triangle(h + s) - \Pi(h) + B_o(h + s - z)\cos\alpha, \tag{9}$$

where possible substrate topographies may be included through $s(x, y)$ (though not considered here for simplicity), and $\Pi(h)$ is a disjoining pressure term which is defined in [22, 23]. This term is used to alleviate the singularity at the moving contact line,

and is given by

$$\Pi(h) = \frac{(n-1)(m-1)(1-\cos\Theta_e)}{h^*(n-m)\epsilon^2}\left[\left(\frac{h^*}{h}\right)^n - \left(\frac{h^*}{h}\right)^m\right], \tag{10}$$

where n and m are the exponents of interaction potential and Θ_e is the equilibrium contact angle. The space derivatives in Equations (8) and (9) are both approximated using standard, second order, centred differences.

The computational domain Ω is chosen to be rectangular with non-dimensional Cartesian coordinates $(x, y) \in \Omega = (0, 1) \times (0, 1)$. It is further assumed that the droplet is far away from the boundary. Therefore zero Neumann boundary conditions are applied for both h and p:

$$\frac{\partial h}{\partial \nu} = \frac{\partial p}{\partial \nu} = 0 \ \text{ on } \ \partial\Omega, \tag{11}$$

where ν denotes the outward-pointing normal to the boundary $\partial\Omega$.

For the purpose of validation, we compare results from using our software to selected results presented in [2]. First of all, the values of parameters that are used in the droplet spreading model (Equations (8) to (10)) are presented in Table 1. These values were used by Gaskell et al. in [2].

Parameters	Values	Parameters	Values
B_o	0	ϵ	0.005
Θ_e	1.53°	h^*	0.04
n	3	m	2
α	0°		

Table 1: The parameters of the droplet spreading model that were used by Gaskell et al. in [2].

The initial condition for the variable of droplet thickness $h(x, y, t = 0)$ is given as

$$h^{t=0}(r) = \max\left(5\left(1 - \frac{320}{9}r^2\right), h^*\right), \tag{12}$$

where $r^2 = x^2 + y^2$. Having obtained $h(x, y, t = 0)$, the initial condition for pressure p on all internal grid points i, j $(i, j = 1, \ldots, n)$ may be defined as

$$\begin{aligned} p_{i,j}^{t=0} = {} & \frac{1}{dx^2}\left\{h_{i+1,j}^{t=0} + h_{i-1,j}^{t=0} + h_{i,j+1}^{t=0} + h_{i,j-1}^{t=0} - 4h_{i,j}^{t=0}\right\} \\ & + \frac{(n-1)(m-1)(1-\cos\Theta_e)}{h^*(n-m)\epsilon^2}\left[\left(\frac{h^*}{h_{i,j}^{t=0}}\right)^n - \left(\frac{h^*}{h_{i,j}^{t=0}}\right)^m\right] \\ & - B_o h_{i,j}^{t=0}\cos\alpha, \end{aligned} \tag{13}$$

in which it is assumed that $s = z = 0$.

We choose Figure $5(b)$ from [2] to validate against. This figure shows the evolution of the maximum height of the droplet during simulations. All grids are uniform and the non-dimensional time duration is $[0, 10^{-5}]$. In Figure 5, the left-hand side shows a copy of Figure $5(b)$ from [2]. The right-hand side figure shows the results using our multigrid solver. The maximum height of the droplet is initially 5.0, as implied by the initial condition in Equation (12). Figure 5 shows a good agreement with the results from grid hierarchies [1] $16^2 - 512^2$ and $16^2 - 1024^2$. For the coarser grid hierarchies (i.e. $16^2 - 64^2$, $16^2 - 128^2$ and $16^2 - 256^2$), our results appear to be more accurate than the ones from [2]. This may be caused by the use of adaptive time stepping in [2], in which the size of the time step is based upon local error estimation, as opposed to our choice of a fixed time step size, systematically reduced in proportion to dx on the finest grid.

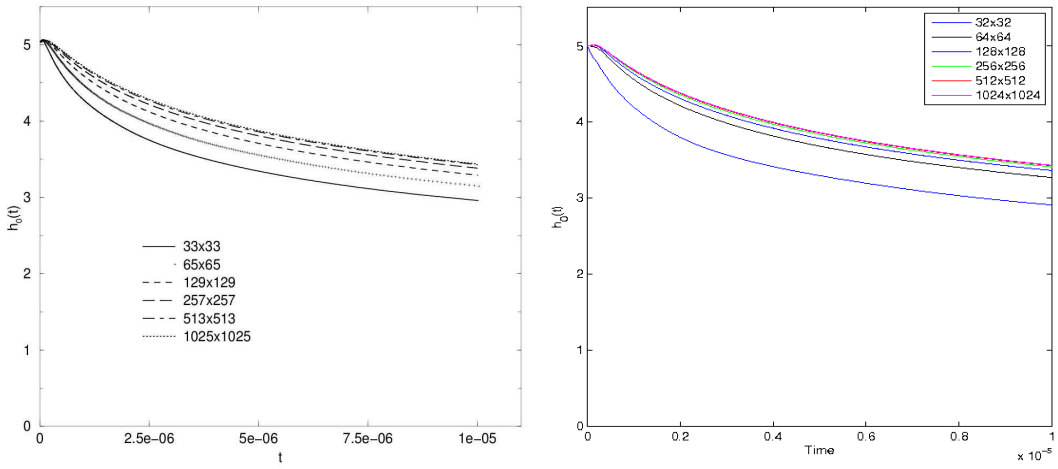

Figure 5: The evolution of the maximum height of the droplet during simulations, on the left-hand side is Figure $5(b)$ from [2] and on the right-hand side, we show results from our multigrid solver. Parameters used to generate these results are shown in Table 1. Legends in these figures indicate the finest resolutions of grids that are used for each simulation. Note that in the left-hand figure the grid resolution is indicated by numbers of nodes, while in the right-hand figure it is indicated by numbers of cell.

In order to generate these results, we use a 16^2 grid as the coarsest grid (also as the block size). There are in total 4 smoothing iterations on each grid level, i.e. $p_1 = p_2 = 2$ in Algorithm 1, and 60 iterations of the smoother are used for the coarsest grid solver. The time step size for grid hierarchy $16^2 - 32^2$ is $\delta t = 3.2 \times 10^{-7}$. Each time the finest grid is refined, the time step size is halved.

At each time-step, convergence of the multigrid iteration is checked after each V-cycle and the iteration is stopped if either $||r||_\infty < 10^{-6}$ or $||r||_\infty / ||r_1||_\infty < 10^{-5}$, in

[1] The notation we use here which identifies the grid hierarchy is showing the coarsest and the finest grid resolutions. For example, $16^2 - 512^2$ indicates that there are 6 grids: 16^2, 32^2, 64^2, 128^2, 256^2 and 512^2.

which r is the residual (for h or p) and r_1 is the residual after the first V-cycle of the current time-step.

Since the solver from [2] also performs a nonlinear multigrid iteration with FAS, we validate the performance of our multigrid solver against the one used by Gaskell et al. More specifically, we validate the convergence rate of each multigrid V-cycle for a typical time step, based upon the infinity norm of residuals. This is shown in Figure $4(b)$ from [2]. In Figure 6, the left-hand side shows the performance of the solver used in [2]. On the right-hand side is the performance of our multigrid solver. For both solvers, a total number of 10 V-cycles within this particular time step are performed. From this figure, the results suggest that both solvers perform similarly. It is worth noting there is one significant difference. In the results from [2], the convergence rate deteriorates significantly from the 9^{th} V-cycle to the 10^{th} V-cycle. However, the results from using our multigrid solver remain robust in this situation. This may be due to the use of a different spatial discretization in [2] to the cell-centred scheme used in this work. Overall these tests provide excellent validation.

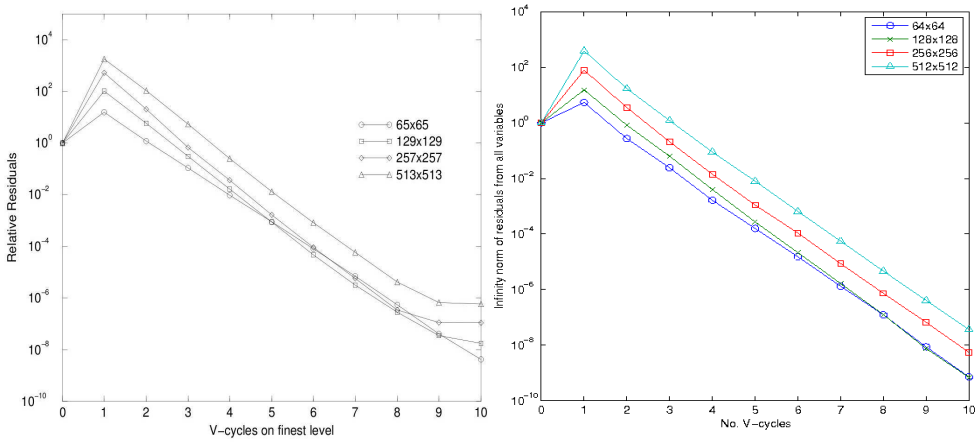

Figure 6: Convergence rate of a typical multigrid V-cycle for a single time step. On the left-hand side is Figure $4(b)$ from [2] and on the right-hand side are the results of our multigrid solver. Four different finest grid resolutions are used, as shown in the legends. Parameters that are used to generate these results are shown in Table 1.

From the right-hand side of Figure 6, we also see that all the curves are nearly parallel, which implies the reduction in the residual is independent from the sizes of the grids. This optimal convergence rate is the goal of multigrid methods, and indicates that the complexity of our multigrid solver is linear. We summarise the CPU time costs from five simulations in Figure 7, with finest grid sizes of 128^2, 256^2, 512^2, 1024^2 and 2048^2, respectively. As we quadruple the number of points on the finest grid, we also halve the time-step size, and all simulations finish at the same end time $T = 1 \times 10^{-5}$. We use a log-log plot to illustrate the relation between the number of grid points and the average CPU time per time step from these five simulations in Figure 7. We conclude that our multigrid solver does indeed have a linear complexity.

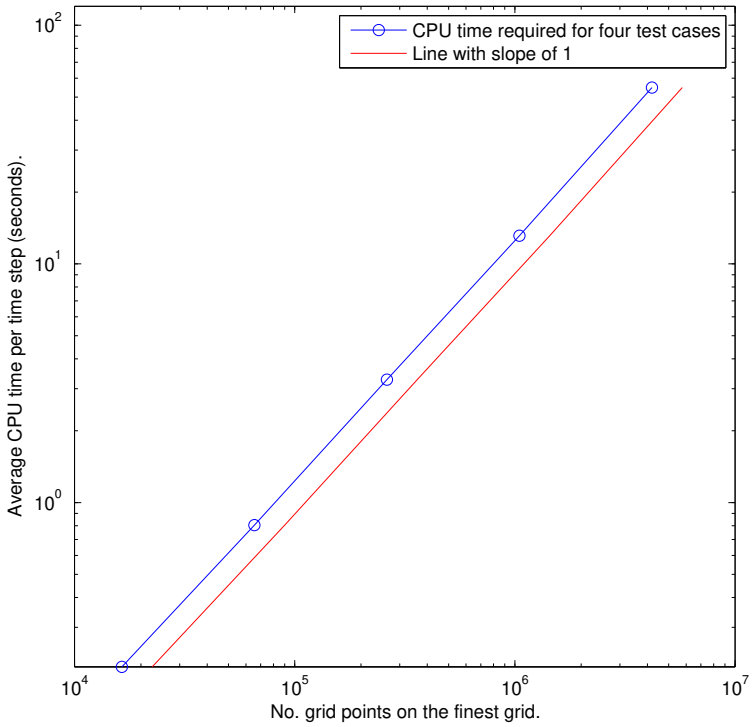

Figure 7: A log-log plot of the CPU time per time step against the total number of grid points from the finest grid. For comparison, a line with slope of 1 is also shown in the figure.

So far the tests presented have only used uniform grids and fixed time step sizes. In order to investigate the effectiveness of our adaptivity techniques, we first consider the use of adaptive time-stepping. As mentioned previously, this is achieved through using the adaptive BDF2 method [17]. Here we present results from two different grid hierarchies, for which the finest grids have 512^2 and 1024^2 grid points respectively. The equivalent simulations using fixed time step sizes are already presented in the right-hand side of Figure 5. For the adaptive time stepping we use the same initial step size as in the non-adaptive cases: initial time step sizes for these two cases are 2×10^{-8} and 1×10^{-8} respectively. It is now possible to contrast adaptive time step selection against the equivalent simulations (from the right-hand side of Figure 5) that are undertaken using fixed time step sizes. For the 512^2 case, 500 fixed time steps are required with a step size of 2×10^{-8}. For 1024^2 case, 1000 time steps are required with a step size of 1×10^{-8}. Since the very small time steps are only actually required at very early times, the use of our adaptive time stepping approach reduces the number of time steps required to 39 and 45, respectively.

The detailed comparison between the use of fixed time step size and adaptive time stepping is presented in Table 2. In the 512^2 case, adaptive time stepping takes just 9.6% of the time taken when fixed time steps are used. This percentage becomes 5.2% for the simulation in the 1024^2 case. This is despite the increased average number of

V-cycles required per time step in the adaptive case. Note however that the number of V-cycles needed is still independent of the grid size.

Case	No. TSs fixed δt	Avg. V-cycle per TS	CPU time (seconds)	No. TSs ATS	Avg. V-cycle per TS	CPU time (seconds)
512^2	500	5.0	2095.3	39	5.9	201.5
1024^2	1000	5.0	16721.3	45	5.8	874.4

Table 2: Comparisons between the use of fixed time step size and the adaptive time stepping for two test cases. The total number of time steps, the average V-cycles required per time step and the CPU time are used for the comparisons. Due to the limit of space, abbreviations are used, where TS means "time step", Avg. means average and ATS is short for adaptive time stepping.

For completeness, two questions are worth asking. Firstly, are our choices of the time step size too small for the fixed time step approach? In other words, could the fixed time step approach take a larger step size and be more competitive? Additional tests show that for the 512^2 case, increasing the initial time step size by a factor of 5 causes the multigrid solver to converge more slowly as, within each time step, about three more V-cycles are needed. The computation fails to converge if the initial time step size is increased by a factor of 10. Thus, the adaptive time-stepping outperforms the fixed time-stepping by a large margin for this specific problem.

The second question is: when using the adaptive time stepping, how accurate are these solutions? Since the exact solution to this problem is not known we choose to base our assessment of the accuracy upon a comparison of the height of the simulated droplets at the centre of the domain (the maximum height of the droplet) as a function of time, as shown in Figure 5. From this figure, it can be seen that by using adaptive time-stepping, the overall evolutions of the height of the droplet are very close to the ones using the original approach with fixed time step sizes. To give a further indication of how accurate the solutions are at the end of simulation, a zoom-in is shown in the right-hand side of Figure 8. The results shown indicate that our adaptive time stepping approach deteriorates the accuracy by only a very small amount. More specifically, for the 512^2 case, the values of the maximum heights of the droplet at $T = 10^{-5}$ are 3.406 from the use of fixed time step size, and 3.403 from the use of adaptive time stepping. For the 1024^2 case, the value is 3.423 from the use of fixed time step, and is 3.419 from the use of adaptive time stepping.

This droplet spreading test case is one which is likely to benefit from employing AMR, since the problem features a distinct radial moving contact line which must be accurately resolved, while elsewhere the solution is relatively smooth. To illustrate our AMR strategy, we choose three test cases for the purpose of demonstration. Their finest grids, if refined everywhere, would have resolutions of 256^2, 512^2 and 1024^2.

Previously in Section 2, we note that the quality of the AMR is controlled by problem-specific refinement and coarsening criteria. For this droplet spreading model,

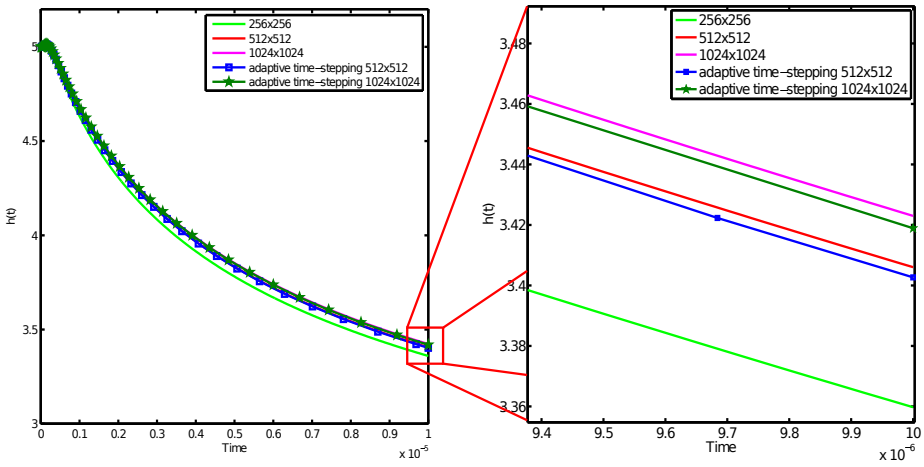

Figure 8: The left-hand side figure shows the evolution of the maximum height of the droplet in selected cases. Results with the finest grids 256^2, 512^2 and 1024^2 have been previously presented in Figure 5, where they are obtained using the fixed time step size. The right-hand side figure shows a zoom-in for the end of the graphs that are presented in the left-hand side figure.

our adaptive refinement strategy is based upon a discrete approximation to the second derivative of the droplet thickness h (i.e. $|\nabla^2 h|$). Within each mesh block, at every grid point, (i, j), the adaptive assessment is computed via:

$$\text{adaptive assessment}_{i,j} = |h_{i+1,j} + h_{i-1,j} + h_{i,j+1} + h_{i,j-1} - 4h_{i,j}| \approx dx^2 |\nabla^2 h|_{i,j}. \quad (14)$$

Each block is then flagged for refinement or coarsening based on the maximum value of adaptive assessment within the block. In this work:

- if adaptive assessment is greater than the threshold: $>$ *ctore* $= 0.01$, mark the block for refinement.
- if adaptive assessment is less than the threshold: $<$ *ctode* $= 0.001$, mark the block for coarsening.

This choice of thresholds is quite aggressive, so most of the computation is around the moving contact line.

Here we evaluate the AMR on its own, i.e. without using adaptive time-stepping. In Table 3, details of the three different test cases are presented. They are AMR 256^2, AMR 512^2 and AMR 1024^2. For comparison, we also include the CPU times for these test cases when uniform grids and fixed time step size are employed. From this table, the efficiency gained from using AMR is demonstrated. For instance, in AMR 1024^2 case, the CPU time is only 19.3% of the corresponding time using uniform grids, and the average number of V-cycles per time step increases by only half a cycle.

Cases	δt	Time steps	Avg. V-cycles per time step	CPU time (seconds)	CPU time (seconds) from uniform grids
AMR 256^2	4×10^{-8}	250	5.0	175.1	303.2
AMR 512^2	2×10^{-8}	500	5.0	645.6	2345.7
AMR 1024^2	1×10^{-8}	1000	5.5	3578.9	18521.1

Table 3: Details of three test cases using aggressive AMR with fixed time steps. CPU times when only using uniform grids are also included for comparison.

Having presented the CPU time, we further compare the number of grid points on the finest grids used in the uniform cases to the number of leaf grid points that are used in the adaptive cases. The leaf grid points are those grid points that are on the finest refinement level present in their local region. Since refining and coarsening are carried out dynamically, these numbers of leaf points are the maximum numbers that occurred throughout each of the simulations. In Table 4, this comparison is summarised. From this table, the computational workload saved by using the AMR compared to the use of uniform grids is seen to be substantial. For example, in the AMR 1024^2 case, the number of leaf points is less than 1.0% of the number of points on the finest uniform grid.

Cases	Maximum No. leaf grid points	Total No. grid points from uniform grids	Ratio between AMR and uniform grids
AMR 256^2	$2,048$	$65,536$	0.0313
AMR 512^2	$6,400$	$262,144$	0.0244
AMR 1024^2	$10,240$	$1,048,576$	0.0098

Table 4: Comparison of the maximum number of leaf grid points used in adaptive test cases and the total number of grid points in uniform test cases. A ratio between the number of leaf points with AMR and the number of grid points with uniform grids is also presented.

We have demonstrated that the use of AMR significantly improves the efficiency of the computation. However, it can be seen from Tables 3 and 4 that the CPU costs do not reduce with the same rate as the number of grid points. This is because, even when the grid points are significantly decreased by using AMR, the number of grid visits is still the same in our multigrid solver. Additionally, overheads occur when we dynamically maintain the parent-children relations between coarsening and refining. Furthermore, since this software is written with parallelization in mind, other extra overheads also exist in the implementation to deal with issues arising from parallel situations, and those overheads also affect the performance when only one CPU is employed, e.g. keeping a well parallelizable ordering of all mesh blocks dynamically during the simulation.

Having addressed the effectiveness of our AMR, it leads to the inevitable question:

how accurate are the solutions from using the AMR? Once again we use the discrepency in the maximum height of the droplet between different simulated solutions as a proxy for the error, comparing AMR results against those obtained using uniform grids. In the left-hand side of Figure 9, the evolutions of the maximum height of the droplet from the three test cases (i.e. AMR 256^2, AMR 512^2 and AMR 1024^2) with the use of AMR are presented. Results from using uniform grids (previously shown in Figure 5) are also presented for comparison. From this figure, we see that the use of the AMR produces almost identical results to the ones from using uniform grids.

Furthermore, to assess this in more detail, a zoom-in is shown on the right-hand side of Figure 9 which focusses on the solution at the end of the simulation, and height values are given in Table 5. Using adaptive grids generally compromises the accuracy, as expected given the enormous reduction in degrees of freedom, and these results demonstrate this. It is important to note however that the accuracy of the 1024^2 solution with AMR is much better than the one from the 512^2 case with uniform grid (i.e. the maximum height of the droplet is much closer to that computed from the 1024^2 case with a uniform grid). It may also be seen that the solution of the AMR 256^2 is almost identical to the one from the 256^2 case with uniform grid.

Cases	Maximum height	Cases	Maximum height
AMR 256^2	3.35971	256^2	3.35974
AMR 512^2	3.405	512^2	3.406
AMR 1024^2	3.418	1024^2	3.423

Table 5: Comparison of maximum droplet height values at $t = 1 \times 10^{-5}$ for adapted and uniform meshes.

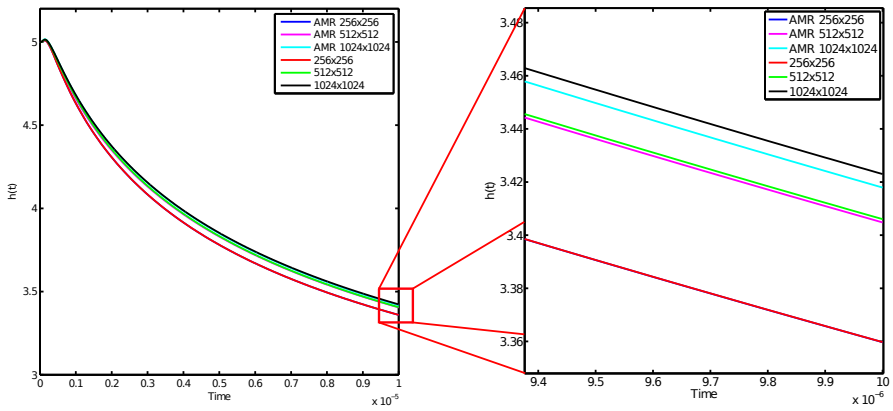

Figure 9: In the left-hand side figure, the evolution of the maximum height of the droplet is plotted from using both AMR and uniform grids. On the right-hand side, a zoom-in is included to assess the accuracy near the final time.

The AMR implemented in Campfire aims to dynamically adapt the mesh according

to the evolution of the solution. Here we present snapshots of the evolution of the mesh refinement during typical simulations. These are shown in Figure 10, where different colours are used to identify different levels of mesh refinement. Results shown in this figure demonstrate the dynamic evolution of AMR during a typical simulation.

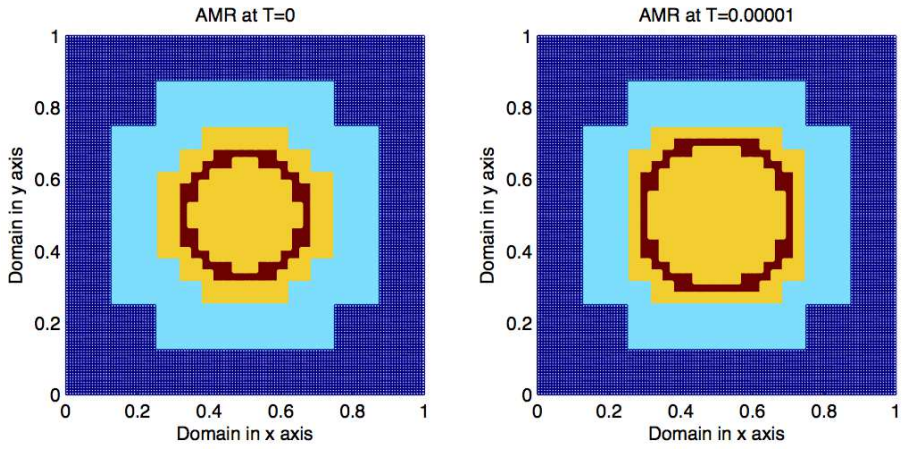

Figure 10: Snapshots of the evolution of AMR during a typical simulation. Left-hand side is the AMR at $t = 0$ and right-hand side shows the AMR at $t = 1 \times 10^{-5}$. Mesh refinement levels: 128^2 (dark blue); 256^2 (light blue); 512^2 (yellow); 1024^2 (red).

The spatial and the temporal discretization schemes that are used in this work are both second-order accurate. Therefore, we expect the overall convergence rate to be second order, i.e. the error behaves as $O(\delta t^2, dx^2)$ as the mesh is refined. This means that halving the time step size and doubling the number of grid points in each direction should reduce the error by a factor of four. Although we do not have an exact solution to compare with, we can still look at differences between solutions at successive levels of refinement: if these also reduce by a factor of four each time δt and dx are halved then this indicates second order convergence to a solution. In order to illustrate this, we conduct our convergence tests based upon solution restriction. For example, consider three grid hierarchies using 2-D grids: $8^2 - 16^2$, $8^2 - 32^2$ and $8^2 - 64^2$. Each grid is associated with a δt: $\delta t_{(16^2)}$, $\delta t_{(32^2)} = \frac{\delta t_{(16^2)}}{2}$ and $\delta t_{(64^2)} = \frac{\delta t_{(32^2)}}{2}$, respectively. Solutions are obtained by solving the same problem on these three finest grids separately, with their corresponding δt, and with the assumption that the ending time T is exactly the same for all runs. To make a comparison between two solutions we restrict the fine grid solution to the coarse grid by using a restriction operator (e.g. four-point averaging shown in Equation (5)). Thus, the solution which is restricted from grid hierarchy $8^2 - 32^2$ can be compared to the solution from grid hierarchy $8^2 - 16^2$. Similarly, the restricted solution from hierarchy $8^2 - 64^2$ can be compared to the original solution from hierarchy $8^2 - 32^2$.

The norms of the differences between the coarser grid solution and the restriction of the finer (possibly adapted) solution restricted to the coarser grid which we consider

are

$$||e||_\infty := max(|u_{i,j}^{restricted} - u_{i,j}|), \qquad ||e||_2 := \sqrt{\frac{\sum_{i=1}^{N}\sum_{j=1}^{N}(u_{i,j}^{restricted} - u_{i,j})^2}{N \times N}}, \quad (15)$$

where $u^{restricted}$ is the restricted solution from the finer grid hierarchy, u is the solution from the coarser grid hierarchy, $i, j = 1, \ldots, N$ and N is the number of internal grid points in each axis direction on the finest level of the coarser grid hierarchy. When adaptive grids are used, at the final stage $t = T$, we transfer the solutions to the uniform meshes that are needed in order to carry out these comparisons.

In Table 6, we present convergence results that are generated using adaptive time-stepping, AMR and parallel computing from four different simulations. Their highest levels of mesh refinement are equivalent to uniform grid resolutions of 512^2, 1024^2, 2048^2 and 4096^2 respectively. A 16^2 coarsest grid is used for all cases. The end time is chosen to be $T = 2.0 \times 10^{-2}$, which is longer than the previous ones, and constitutes a "full" simulation. Results shown in this table clearly demonstrate second order convergence for both variables h and p.

For variable h						
Cases	Starting δt	Time steps	Infinity norm	Ratio	Two norm	Ratio
AMR 512^2	1×10^{-8}	1418	-	-	-	-
AMR 1024^2	5×10^{-9}	2011	2.480×10^{-5}	-	1.117×10^{-5}	-
AMR 2048^2	2.5×10^{-9}	25184	6.174×10^{-6}	4.02	2.737×10^{-6}	4.08
AMR 4096^2	1.25×10^{-9}	472368	1.544×10^{-6}	3.99	6.864×10^{-7}	3.99
For variable p						
AMR 512^2	1×10^{-8}	1418	-	-	-	-
AMR 1024^2	5×10^{-9}	2011	5.321×10^{-2}	-	2.007×10^{-2}	-
AMR 2048^2	2.5×10^{-9}	25184	1.324×10^{-2}	4.02	4.873×10^{-3}	4.12
AMR 4096^2	1.25×10^{-9}	472368	3.309×10^{-3}	3.99	1.208×10^{-3}	4.04

Table 6: Results show the differences in consecutive solutions measured in the stated norm, followed by the ratio of consecutive differences from the droplet spreading model with the initial condition given in Equation (12). These results are generated with solutions at $T = 2.0 \times 10^{-2}$ using spatial and temporal adaptivity.

3.2 A Multi-Phase-Field Model of Tumour Growth

In the previous section we illustrated the performance of our software for a 2-dimensional mathematical model of thin film flow. In this section we consider an even more complex test problem, with a greater number of nonlinear PDEs and in both two and three space dimensions.

To study the complex procedures of tumour growth and its interactions with the host, a continuum modelling technique which consists of a set of PDEs can be used to

model the morphology of tumours. The review papers [27, 28, 29] describe a number of examples of how such models can be derived. In the work presented here, a multi-phase-field model of tumour growth is considered, from Wise et al. [3]

There are, in total, four independent phase-field variables in this model, namely ϕ_W, ϕ_H, ϕ_V and ϕ_D, which represent volume fractions of extracellular fluid, healthy cells, viable tumour cells and dead tumour cells, respectively. In addition, there are three assumptions applied to these volume fractions.

1. The extracellular fluid volume fraction is everywhere constant, i.e. $\phi_W(x, y, z, t) = \phi_{W,0} =$ constant.

2. Cells are assumed to be close-packed, so $\phi_H + \phi_V + \phi_D = 1$, and the range of values of these phase-field variables is from 0 to 1.

3. Inside the tumour there are only two types of cells: viable and dead. This indicates the total tumour cell volume fraction is $\phi_T = \phi_V + \phi_D$.

Based upon these three assumptions, there are only two phase-field variables that are required to be solved, and they are ϕ_T and ϕ_D. Once these two variables are obtained, other variables may be derived from the assumptions made.

The component ϕ_T is assumed to obey the following Cahn-Hilliard-type advection-reaction-diffusion equations:

$$\frac{\partial \phi_T}{\partial t} = M\nabla \cdot (\phi_T \nabla \mu) + S_T - \nabla \cdot (\underline{u}_S \phi_T), \tag{16}$$

$$\mu = f'(\phi_T) - \epsilon^2 \nabla^2 \phi_T, \tag{17}$$

where $M > 0$ is the mobility constant, $f(\phi_T) = \phi_T^2(1 - \phi_T)^2/4$ is the quartic double-well potential, $\underline{u}_S$ is the tissue velocity (which is substituted for using Equation (23)), and $\epsilon > 0$ is an interface thickness parameter between healthy and tumour tissue. S_T is the net source of tumour cells, and is given as

$$S_T = nG(\phi_T)\phi_V - \lambda_L \phi_D, \tag{18}$$

where n is the concentration of nutrient, which is specified in Equation (26), $\phi_V = \phi_T - \phi_D$, and $\lambda_L \geq 0$ is the rate of tumour cell proliferation. $G(\phi_T)$ is a continuous cut-off function defined as

$$G(\phi_T) = \begin{cases} 1 & \text{if } \frac{3\epsilon}{2} \leq \phi_T \\ \frac{\phi_T}{\epsilon} - \frac{1}{2} & \text{if } \frac{\epsilon}{2} \leq \phi_T < \frac{3\epsilon}{2} \\ 0 & \text{if } \phi_T < \frac{\epsilon}{2}. \end{cases} \tag{19}$$

A similar dynamical equation for predicting the volume fraction of dead tumour cells ϕ_D is used:

$$\frac{\partial \phi_D}{\partial t} = M\nabla \cdot (\phi_D \nabla \mu) + S_D - \nabla \cdot (\underline{u}_S \phi_D), \tag{20}$$

where S_D is the net source of dead tumour cells, defined as

$$S_D = (\lambda_A + \lambda_N \mathcal{H}(n_N - n))\,\phi_V - \lambda_L \phi_D, \tag{21}$$

where λ_A is the death rate of tumour cells from apoptosis, λ_N is the death rate of tumour cells from necrosis, n_N is the necrotic limit (necrosis only occurs when the nutrient value is below this limit), and $\mathcal{H}$ is a Heaviside function. This Heaviside function is discontinuous and thus could prevent us from obtaining a higher order convergence rate, so we instead use the smoother approximation given by

$$\mathcal{H}(n_N - n) = \begin{cases} 1 & \text{if } n_N - n \geq \epsilon^s \\ -\frac{1}{4(\epsilon^s)^3}(n_N - n)^3 + \frac{3}{4\epsilon^s}(n_N - n) + \frac{1}{2} & \text{if } -\epsilon^s \leq n_N - n \leq \epsilon^s \\ 0 & \text{if } n_N - n < -\epsilon^s, \end{cases} \tag{22}$$

where ϵ^s controls the steepness of the smooth transition between 0 and 1.

The tissue velocity $\underline{u}_S$ is assumed to obey Darcy's law, and is defined as

$$\underline{u}_S = -\kappa(\phi_T, \phi_D)(\nabla p - \frac{\gamma}{\epsilon}\mu \nabla \phi_T), \tag{23}$$

where $\kappa(\phi_T, \phi_D) > 0$ is the tissue motility function and $\gamma \geq 0$ is a measure of the excess adhesion. An additional assumption made by Wise et al. [3] is that there is no proliferation or death of the host tissue, thus the velocity is constrained to satisfy

$$\nabla \cdot \underline{u}_S = S_T. \tag{24}$$

Instead of solving for the tissue velocity, Equations (23) and (24) are combined together, and a Poisson-like equation for the cell pressure p can be constructed:

$$-\nabla \cdot (\kappa(\phi_T, \phi_D)\nabla p) = S_T - \nabla \cdot (\kappa(\phi_T, \phi_D)\frac{\gamma}{\epsilon}\mu \nabla \phi_T). \tag{25}$$

A quasi-steady equation is given for the nutrient concentration through diffusion:

$$0 = \nabla \cdot (D(\phi_T)\nabla n) + T_c(\phi_T, n) - n(\phi_T - \phi_D), \tag{26}$$

where

$$D(\phi_T) = D_H(1 - Q(\phi_T)) + Q(\phi_T) \tag{27}$$

is the diffusion coefficient, D_H is the nutrient diffusivity in the healthy tissue, $Q(\phi_T)$ is an interpolation function, given by

$$Q(\phi_T) = \begin{cases} 1 & \text{if } 1 \leq \phi_T \\ 3\phi_T^2 - 2\phi_T^3 & \text{if } 0 < \phi_T < 1 \\ 0 & \text{if } \phi_T \leq 0. \end{cases} \tag{28}$$

and

$$T_c(\phi_T, n) = (v_P^H(1 - Q(\phi_T)) + v_P^T Q(\phi_T))(n_C - n) \tag{29}$$

is the nutrient capillary source term. Furthermore, $v_P^H \geq 0$ and $v_P^T \geq 0$ are constants specifying the degree of pre-existing uniform vascularization, and $n_C \geq 0$ is the nutrient level in capillaries.

To sum up, this multi-phase-field model of tumour growth consists of a coupled system made up of Equations (16), (17), (20), (25) and (26). There are five dependent variables in total in this system: two phase-field variables, ϕ_T and ϕ_D; and three supplementary variables, μ, p and n. These PDEs are valid throughout a domain Ω, and there are no internal boundary conditions for the solid tumour, the necrotic core or other variables. Therefore, only one set of outer boundary conditions is required (provided the tumour is in the domain's interior) and this set is the following mixture of Neumann and Dirichlet boundary conditions:

$$\mu = p = 0, \quad n = 1, \quad \frac{\partial \phi_T}{\partial \nu} = \frac{\partial \phi_D}{\partial \nu} = 0 \ \text{ on } \ \partial\Omega, \tag{30}$$

where ν denotes the outward-pointing normal direction to the boundary $\partial\Omega$.

We discretise all of the derivatives (including those of odd order) in this model using second order centred differences. This approximation may introduce spurious numerical oscillations in the vicinity of steep fronts which could lead to unphysical values for the phase volume fractions. In order to retain the expected second order convergence rate while forcing the phase volume fractions to lie between 0 and 1 (or very close to that interval) additional penalty terms are added to Equations (16) and (20), which take the form

$$\frac{1}{\delta} \min\left(\phi_T, 0\right) \quad \text{and} \quad \frac{1}{\delta} \max\left(\phi_T - 1, 0\right) \tag{31}$$

for ϕ_T, with a corresponding term added to the ϕ_D equation. These terms have no impact when $0 \leq \phi \leq 1$, but create a large correction to the system whenever ϕ tries to take a value outside of this interval. The smaller the choice of the penalty parameter δ (i.e. the larger $1/\delta$) the larger this correction becomes, forcing the values of ϕ to be close to this range but at the expense of adding to the nonlinearity of the resulting system. The default value of δ used in this work is 10^{-4}. This is a slightly more relaxed constraint on the range of values that can be taken by ϕ_D and ϕ_T than that imposed by Wise et al. [3], who enforced $\phi \in [0, 1]$ in a non-smooth manner.

In order to define the initial conditions for the 2-D simulations, firstly the 2-D domain Ω is given by $(x, y) \in \Omega = [0, 40] \times [0, 40]$. An initial condition for ϕ_T is defined to be

$$\begin{aligned} \phi_T(x, y) =& 1 \quad \text{if} \quad \frac{(x-20)^2}{1.1} + (y-20)^2 \leq 2^2, \\ =& 0 \quad \text{otherwise.} \end{aligned} \tag{32}$$

This initial condition is discontinuous, so we employed a simple Jacobi iteration with 5 sweeps to smooth the initial conditions, both to allow second order accuracy to be seen and to avoid unnecessarily restrictive time steps at the start of the simulation. A

2-D version of this iteration is

$$\phi_{T\,i,j}^{\ell+1,t=0} = \frac{1}{4}\left(\phi_{T\,i+1,j}^{\ell,t=0} + \phi_{T\,i-1,j}^{\ell,t=0} + \phi_{T\,i,j+1}^{\ell,t=0} + \phi_{T\,i,j-1}^{\ell,t=0}\right). \tag{33}$$

In addition, $\phi_D(t=0) = 0$ is assumed so that there are initially no dead tumour cells. $\mu(t=0)$ is straightforward to calculate since μ is a function of ϕ_T, as shown in Equation (17). The initial values for the pressure p and nutrient n require the application of a solver. Due to the increased computational cost in 3-D, an additional multigrid solver is implemented to solve first for the steady state solution of $n(t=0)$ (since, in Equation (26), n is not dependent upon p), then for $p(t=0)$, using Equation (25). Two stopping criteria are used, the absolute criterion is dependent upon the infinity norm of the residuals of n and p, respectively, and it terminates when $||r||_\infty \leq 1 \times 10^{-9}$; the relative criterion is dependent upon the reduction of the infinity norms, and each of them terminates when the reduction is more than 1×10^{-10}.

The values of the parameters that are used in this paper for the multi-phase-field model of tumour growth are presented in Table 7.

Parameters	Values	Parameters	Values
M	10.0	ϵ	0.1
λ_L	1.0	λ_A	0.0
λ_N	3.0	γ	0.0
n_N	0.4	D_H	1.0
v_P^H	0.5	v_P^T	0.0
δ	0.0001	ϵ^s	0.2
n_C	1.0		

Table 7: The parameters of the multi-phase-field model of tumour growth. These were used by Wise et al. in [3], except for ϵ^s and δ, which are new parameters introduced in our implementation.

We briefly present some two-dimensional results before moving on to three-dimensional simulations. Note for this model, adaptive time-stepping is not used, because the phase-field (and other) variables do not demand smaller or larger time step sizes as the model evolves in time. In contrast, the droplet spreading in Section 3.1 clearly diffuses into a smoother form as it evolves in time. At each time step the multigrid stopping criterion used is very similar to the droplet spreading model, except that here the infinity norm of the residual from all five variables is considered.

We present the solution for ϕ_T in Figure 11, with a starting time step size of $\delta t^1 = 1 \times 10^{-3}$. The results in this figure show a similar tumour evolution to [3] (for a detailed discussion on validation, see [14]). Our choice for the AMR strategy is very similar to Equation (14), but now takes into account the scaled second derivative of multiple variables: ϕ_T, ϕ_D, p and n. Specifically, these 4 variables are individually assessed by Equation (14): if any one of them requires refinement then the block is marked for

refining; however only if all of them are marked for coarsening is the region marked to be (potentially) coarsened. This is a conservative measure we introduced to ensure accuracy.

As in Section 3.1, we define the refining threshold (*ctore*) to be 0.01 and the coarsening threshold (*ctode*) to be 0.001. In this case however, since the profile of p does not exhibit the sharp fronts of the other variables and varies significantly over a larger subset of the domain, the highest level of mesh refinement at $t = 0$ covers a much larger area than the initial seed of ϕ_T (see Figure 12). Some examples of typical adaptive meshes arising from this AMR strategy in two dimensions are illustrated in Figure 12.

Within a typical time step in the 2-D simulation, we illustrate our optimal multigrid convergence rate with five different grid hierarchies in Figure 13. The maximum of the infinity norm of residuals from four variables (i.e. ϕ_T, μ, ϕ_D and n) is used to demonstrate the multigrid convergence rate. These results suggest that the reduction in the residuals (at least for these four variables) is independent of grid size. The pressure p is not included here because, as has been identified and discussed in [14], the pressure residual decreases more slowly than the residuals from the other governing equations but the lack of such strict convergence in p does not significantly affect the numerical results.

By employing second-order discretization schemes, we expect the overall convergence rate to be second order for this tumour model. We use the convergence measures defined in Equation (15) to evaluate this. The infinity norm and the two norm are computed separately for all five variables using 5 different grid hierarchies. The finest grid used, if refined everywhere, has a grid resolution equivalent to 2048^2. The results for convergence tests are presented in Table 8. The evidence for having obtained second order convergence is compelling, due to the ratio of approximately 4 between consecutive errors for all variables, each time dx and δt are halved.

For 3-D simulations, the imposed initial condition for ϕ_T is defined by three ellipsoids as

$$\begin{aligned}\phi_T(x,y,z) = 1 \quad & \text{if} \quad \frac{(x-19)^2}{1.1} + (y-19)^2 + (z-19)^2 \leq 2^2, \\ & \text{or} \quad (x-20)^2 + \frac{(y-20)^2}{1.1} + (z-20)^2 \leq 2^2, \\ & \text{or} \quad (x-21)^2 + (y-19)^2 + \frac{(z-19)^2}{1.1} \leq 2^2, \\ = 0 \quad & \text{otherwise},\end{aligned} \tag{34}$$

and, as in 2-D, this is smoothed in the manner of Equation (33) with 5 iterations. This model of tumour growth is solved in 3-D with the parameters stated in Table 7 and the initial condition given by Equation (34) in a domain Ω which is defined by $(x, y, z) \in \Omega = [0, 40] \times [0, 40] \times [0, 40]$. The finest grid resolution used, if refined everywhere, is 256^3. The grid points at which ϕ_T has values in the range of 0.5 to 1.0 are illustrated in Figures 14 and 15.

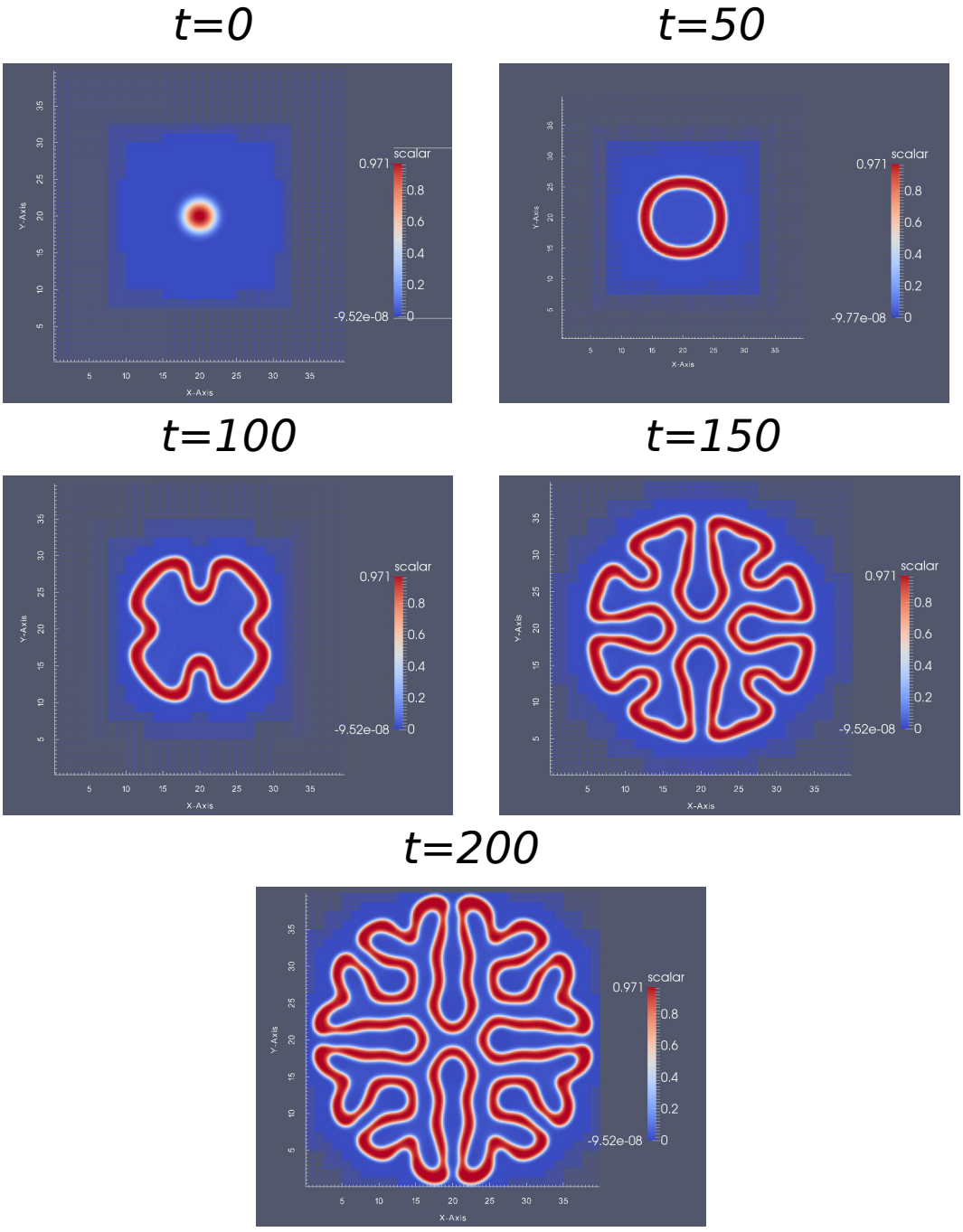

Figure 11: 2-D simulation, showing the evolution of ϕ_T. These results are generated from a grid hierarchy which has 8^2 as the coarsest grid and, if refined everywhere, 2048^2 as the finest grid.

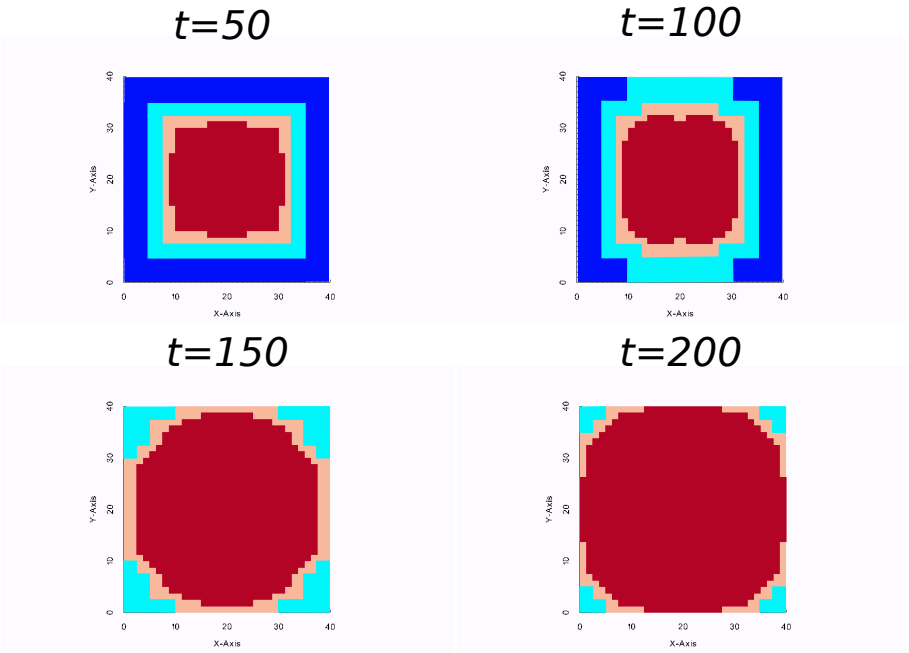

Figure 12: The adaptive meshes from the 2-D simulation shown in Figure 11. Mesh refinement levels: 256^2 (dark blue); 512^2 (light blue); 1024^2 (pink); 2048^2 (red). Note the adaptive meshes are the same at $t = 0$ and $t = 50$, thus only meshes at $t = 50$ are shown here.

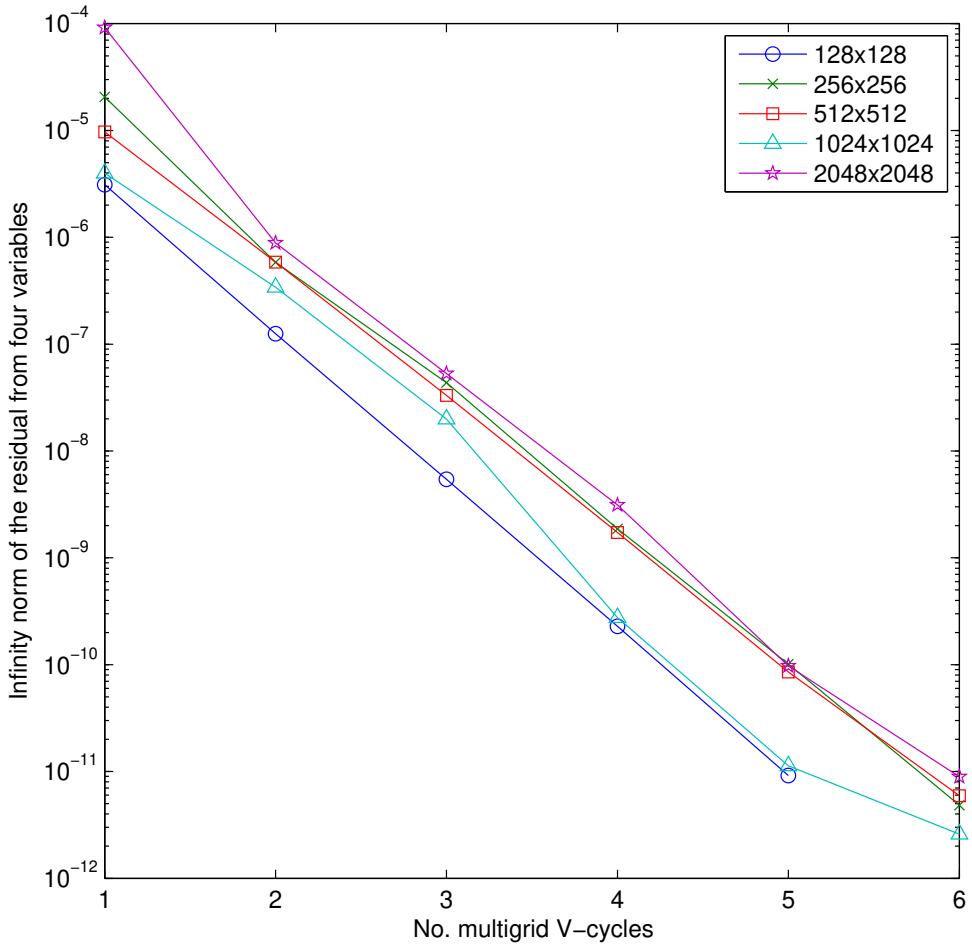

Figure 13: The optimal multigrid convergence rate within a typical time step. There are five different grid hierarchies and the same coarsest grid (8^2) is used, but the number of grid points on the finest grid quadruples each time the number of grid levels increases. The infinity norm of residual is $\max(|r(\phi_T)|, |r(\mu)|, |r(\phi_D)|, |r(n)|)$.

For variable ϕ_T						
Finest levels	δt	Time steps	Infinity norm	Ratio	Two norm	Ratio
$5\ (8^2 - 128^2)$	8×10^{-3}	1250	-	-	-	-
$6\ (8^2 - 256^2)$	4×10^{-3}	2500	0.719×10^{0}	-	4.969×10^{-2}	-
$7\ (8^2 - 512^2)$	2×10^{-3}	5000	6.228×10^{-2}	11.5	4.876×10^{-3}	10.2
$8\ (8^2 - 1024^2)$	1×10^{-3}	10000	1.249×10^{-2}	4.99	1.142×10^{-3}	4.27
$9\ (8^2 - 2048^2)$	5×10^{-4}	20000	3.054×10^{-3}	4.09	2.806×10^{-4}	4.07
For variable μ						
$5\ (8^2 - 128^2)$	8×10^{-3}	1250	-	-	-	-
$6\ (8^2 - 256^2)$	4×10^{-3}	2500	1.367×10^{-2}	-	1.279×10^{-3}	-
$7\ (8^2 - 512^2)$	2×10^{-3}	5000	1.205×10^{-3}	11.3	1.103×10^{-4}	11.6
$8\ (8^2 - 1024^2)$	1×10^{-3}	10000	3.241×10^{-4}	3.72	2.888×10^{-5}	3.82
$9\ (8^2 - 2048^2)$	5×10^{-4}	20000	8.226×10^{-5}	3.94	7.275×10^{-6}	3.97
For variable ϕ_D						
$5\ (8^2 - 128^2)$	8×10^{-3}	1250	-	-	-	-
$6\ (8^2 - 256^2)$	4×10^{-3}	2500	0.245×10^{0}	-	1.923×10^{-2}	-
$7\ (8^2 - 512^2)$	2×10^{-3}	5000	1.663×10^{-2}	14.7	1.976×10^{-3}	14.7
$8\ (8^2 - 1024^2)$	1×10^{-3}	10000	4.303×10^{-3}	3.86	4.837×10^{-4}	4.08
$9\ (8^2 - 2048^2)$	5×10^{-4}	20000	1.076×10^{-3}	4.00	1.206×10^{-4}	4.01
For variable p						
$5\ (8^2 - 128^2)$	8×10^{-3}	1250	-	-	-	-
$6\ (8^2 - 256^2)$	4×10^{-3}	2500	4.918×10^{-2}	-	1.203×10^{-2}	-
$7\ (8^2 - 512^2)$	2×10^{-3}	5000	5.940×10^{-3}	8.28	1.726×10^{-3}	6.97
$8\ (8^2 - 1024^2)$	1×10^{-3}	10000	1.469×10^{-3}	4.04	4.487×10^{-4}	3.85
$9\ (8^2 - 2048^2)$	5×10^{-4}	20000	3.673×10^{-4}	4.00	1.127×10^{-4}	3.98
For variable n						
$5\ (8^2 - 128^2)$	8×10^{-3}	1250	-	-	-	-
$6\ (8^2 - 256^2)$	4×10^{-3}	2500	0.102×10^{-0}	-	1.012×10^{-2}	-
$7\ (8^2 - 512^2)$	2×10^{-3}	5000	7.385×10^{-3}	13.8	1.003×10^{-3}	10.1
$8\ (8^2 - 1024^2)$	1×10^{-3}	10000	1.508×10^{-3}	4.90	2.365×10^{-4}	4.24
$9\ (8^2 - 2048^2)$	5×10^{-4}	20000	3.696×10^{-4}	4.08	5.913×10^{-5}	4.00

Table 8: Results show the differences in consecutive solutions, at $T = 10.0$, measured in the stated norm, followed by the ratio of consecutive differences.

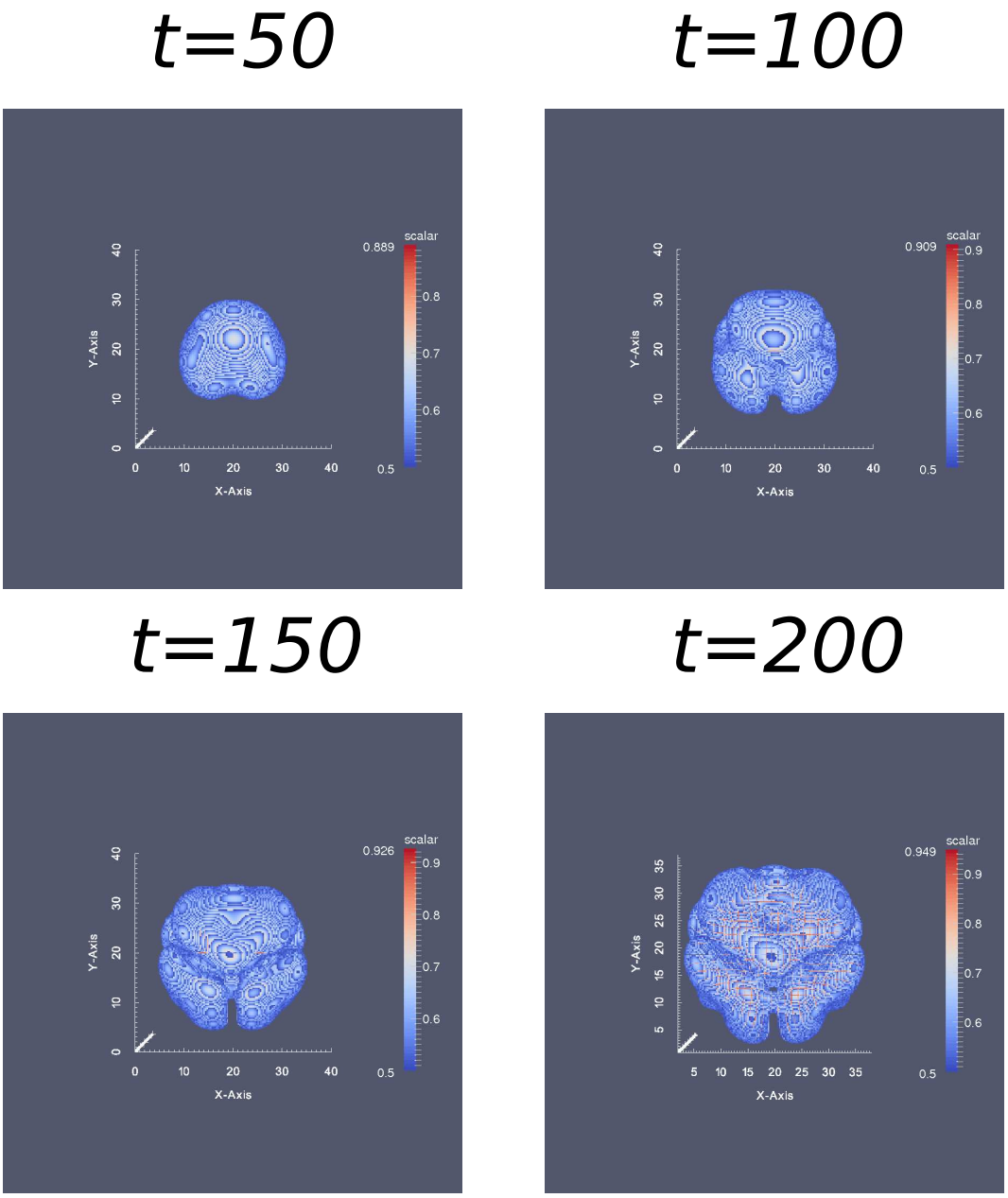

Figure 14: 3-D solutions of variable ϕ_T at $t = 50, 100, 150$ and 200. Images in this figure display the grid points at which ϕ_T takes values in the interval $[0.5, 1.0]$.

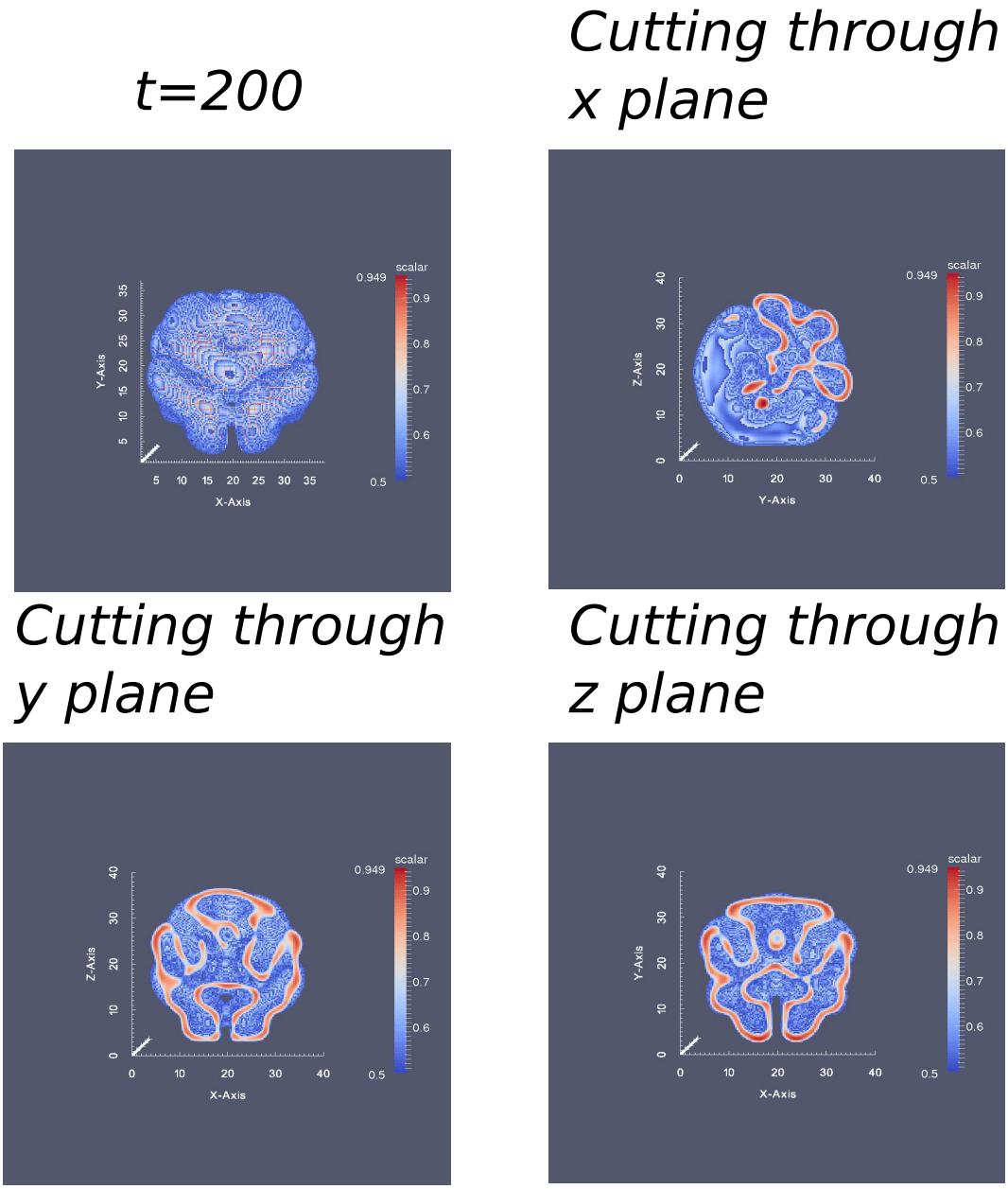

Figure 15: Images of three cross-sections, one through the middle of the computational domain parallel to each coordinate plane, for the solution of ϕ_T at all grid points which have values in the interval $[0.5, 1.0]$ at $t = 200$.

With the capability to run our solver in a parallel environment, here we present results from parallel efficiency tests up to 64 cores. The choice of mesh block size is 16^3, and within this simulation, the coarsest grid is 32^3, the finest grid, if refined everywhere, is 256^3. We run the simulation of tumour growth for 10 time steps starting from $t = 150$, and the resulting parallel efficiency is illustrated on the left-hand side of Figure 16. We denote this test as AMR $32^3 - 256^3$ in the figure. In this case, the deterioration in parallel efficiency from 16 to 64 cores is caused, in part, by the fact that there is not enough workload on the coarsest grid (which only has 8 mesh blocks). Finding a robust solution to this issue is non-trivial, and there are a number of trade-offs that need to be considered. First of all, one may suggest to reduce the block size, in order to have more mesh blocks at the coarsest grid level. However, as discussed in Section 2.4.2, this results in the use of many more guard cells, thus causing a heavier burden on the memory, as well as the parallel communication. Another logical suggestion would be using a finer coarsest grid (i.e. 64^3 in this case). However, nonlinear multigrid with FAS requires an "exact" solution of the nonlinear problem on the coarsest level which may require many more iterations of the coarsest grid solver, which also deteriorates the overall performance of our solver.

In order to verify the latter, we conducted an additional test, using the 64^3 grid as our coarsest grid with a mesh block size of 8^3. This test is identified as AMR $64^3 - 256^3$, and its parallel efficiency is shown on the left-hand side of Figure 16. Clearly AMR $64^3 - 256^3$ has much better parallel efficiency compared to AMR $32^3 - 256^3$. However, although the parallel efficiency may have improved, the actual computational time is seen to be higher. This is due in part to communications, but primarily it is due to increased number of iterations of the coarsest grid solver. This is because solving the coarsest grid problem "exactly" on a finer grid is much more costly. We plot the averaged computational time costs for one V-cycle on the right-hand side graph in Figure 16. With 64 cores, the AMR $64^3 - 256^3$ test case costs more than double the amount of time needed by the AMR $32^3 - 256^3$ case.

4 Conclusion

We have introduced a new engineering software tool, Campfire, which is designed specifically for the solution of parabolic systems of PDEs which involve multiple length and time scales. An essential feature is our built-in nonlinear, optimal multigrid solver which permits stable implicit time-stepping to be utilized. In addition, this is coupled with dynamic AMR, adaptive time-stepping and parallelism through domain decomposition and parallel communication through MPI. We have briefly described the nonlinear multigrid method with FAS, its variation with MLAT and its grid transfer operators. For coupled nonlinear systems, we have proposed a general weighted nonlinear block Jacobi method as the multigrid smoother.

The effectiveness and robustness of this software framework is demonstrated for two applications, one based upon a thin film flow model of droplet spreading [2] and the other a multi-phase-field model of tumour growth [3]. We have validated our re-

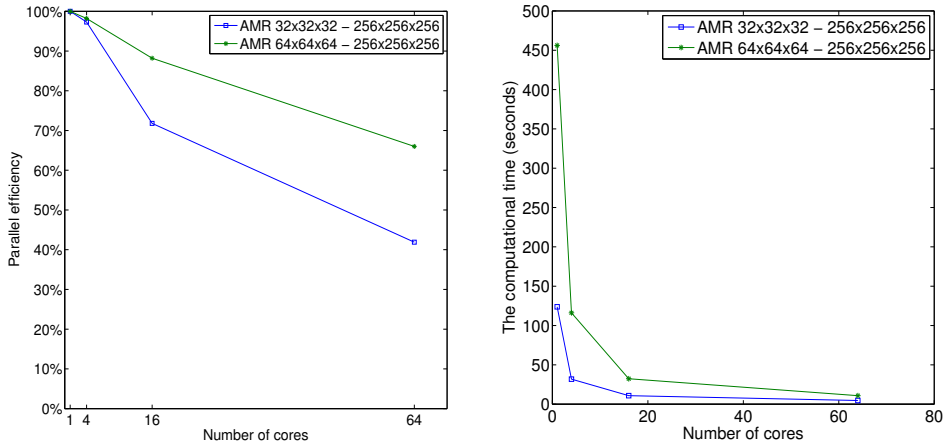

Figure 16: On the left-hand side, parallel efficiency tested up to 64 cores for 10 time steps with a partially developed tumour. On the right-hand side, the average computational time for one V-cycle in seconds is summarised.

sults and evaluated the effectiveness of our adaptive techniques using the thin film flow model. We have also illustrated our dynamic evolving meshes, as well as the optimal multigrid convergence. For the tumour growth model, through the use of penalty terms and a smoothed Heaviside function, we are able to obtain, for the first time, an overall second order convergence rate (only first order solutions were obtained in [3]). This model is also solved in a computationally demanding 3-D context. We present selected computational solutions, as well as results from a typical parallel efficiency test. Although our parallel scaling is not optimal, the efficiency may be increased as the amount of computational work on the finest grid level is increased. The reasons for the challenges associated with obtaining high parallel efficiency are multiple and have been discussed. In particular, the way in which the coarse grid problem is solved is of great importance.

This observation provides one area of focus for our future research. For example, it may be possible to improve overall parallel efficiency through the use of only some of the cores at the coarsest levels. This technique, namely agglomeration, has been discussed in [8]. Future research will also consider whether the dynamic load-balancing algorithm that we use could also be improved: currently we focus only on the efficiency of the parallel smoothing, but this results in relatively inefficient grid transfer operators (in terms of data movement). Other planned enhancements include simplifying the user experience though providing an automated routine that delivers the desired derivatives (see Equation (4)) based upon numerical differentiation, once the discrete system is specified by the user. Finally, one may consider a change in the multigrid algorithm, for example, using a Newton multigrid approach (see [30, 14] for detail), which may allow a much finer coarsest grid (with its linear problem) and this may improve the parallel scaling further.

Acknowledgements

During the writing up stages of this article, Yang was supported by the Leverhulme Trust Research Project Grant (RPG-2014-149).

References

[1] K. Olson, P. MacNeice, "An Overview of the PARAMESH AMR Software and Some of Its Applications", Adaptive Mesh Refinement-Theory and Applications, Lecture Notes in Computational Science and Engineering 41, eds. T. Plewa, T. Linde, G. Weirs (Springer), 2005.

[2] P.H. Gaskell, P.K. Jimack, M.Sellier, H.M. Thompson, "Efficient and Accurate Time Adaptive Multigrid Simulations of Droplet Spreading", International Journal for Numerical Methods in Fluids, 45, 1161-1186, 2004.

[3] S.M. Wise, J.S. Lowengrub, V. Cristini, "An Adaptive Multigrid Algorithm for Simulating Solid Tumor Growth Using Mixture Models", Mathematical and Computer Modelling, 53, 1-20, 2011.

[4] P. Bollada, C. Goodyer, P. Jimack, A. Mullis, F. Yang, "Three Dimensional Thermal-Solute Phase Field Simulation of Binary Alloy Solidification", Journal of Computational Physics, 287, 130-150, 2015.

[5] P.H. Gaskell, P.K. Jimack, M. Sellier, H.M. Thompson, M.C.T. Wilson, "Gravity-Driven Flow of Continuous Thin Liquid Films on Non-Porous Substrates with Topography", Journal of Fluid Mechanics, 509, 253-280, 2004.

[6] A. Brandt, "Multi-Level Adaptive Solutions to Boundary-Value Problems", Mathematics of Computation, 31, 333-390, 1977.

[7] W.L. Briggs, V.E. Henson, S.F. McCormick, "A Multigrid Tutorial", Society for Industrial and Applied Mathematics, 2000.

[8] U. Trottenberg, C.Oosterlee, A.Schuller, "Multigrid", Academic Press, 2001.

[9] W. Bangerth, T. Heister, L. Heltai, G. Kanschat, M. Kronbichler, M. Maier, B. Turcksin, T.D. Young, "The Deal.II Library, Version 8.1", arXiv preprint at http://arxiv.org.abs.1312.2266v4, 2013.

[10] A. Burri, A. Dedner, R. Klöfkorn, M. Ohlberger, "An Efficient Implementation of An Adaptive and Parallel Grid in DUNE", in proceedings of the 2nd Russian-German Advanced Research Workshop on Computational Science and High Performance Computing, 2005.

[11] F.W. Yang, C.E. Goodyer, M.E. Hubbard, P.K. Jimack "Parallel Implementation of an Adaptive, Multigrid Solver for the Implicit Solution of Nonlinear Parabolic Systems, with Application to a Multi-Phase-Field Model for Tumour Growth", in Proceedings of the Fourth International Conference on Parallel, Distributed, Grid and Cloud Computing for Engineering, Iványi, P. and Topping, B. H. V. (Editors), Paper 39, Civil-Comp Press, Stirlingshire, Scotland, United Kingdom, 2015, doi:10.4203/ccp.107.39.

[12] U. Ayachit "The ParaView Guide: A Parallel Visualization Application", Kitware, 2015.
[13] C.E. Goodyer, P.K. Jimack, A.M. Mullis, H.B. Dong, Y. Xie, "On the Fully Implicit Solution of a Phase-Field Model for Binary Alloy Solidification in Three Dimensions", Advances in Applied Mathematics and Mechanics, 4, 665-684, 2012.
[14] F.W. Yang, "Multigrid Solution Methods for Nonlinear, Time-Dependent Systems", PhD Thesis, Thesis Collection, School of Computing, University of Leeds, 2014.
[15] A. Brandt, "Multi-Level Adaptive Technique (MLAT) for Fast Numerical Solution to Boundary Value Problems", Lecture Notes in Physics, 18, 82-89, 1973.
[16] J.M. Ortega, W.C. Rheinbolt, "Iterative Solution of Non-linear Equations in Several Variables", Academic Press, 1970.
[17] E. Emmerich, "Stability and Error of the Variable Two-Step BDF for Semilinear Parabolic Problems", Journal of Applied Mathematics and Computing, 19, 33-55, 2005.
[18] C.E. Goodyer, M. Berzins, "Adaptive Timestepping for Elastohydrodynamic Lubrication Solvers", SIAM Journal on Scientific Computing, 28, 626-650, 2006.
[19] M.D. Lelah, A. Marmur "Spreading kinetics of drops on glass", Journal of Colloid and Interface Science, 82, 518-525, 1981.
[20] A. Oron, S.H. Davis, S.G. Bankoff "Long-scale evolution of thin liquid films", Reviews of Modern Physics, 69, 931-980, 1997.
[21] A. Bertozzi "The mathematics of moving contact lines in thin liquid films", Notices of the AMS, 45, 689-697, 1998.
[22] L.W. Schwartz, R.R. Eley "Simulation of droplet motion on low-energy and heterogeneous surfaces", Journal of Colloid and Interface Science, 202, 173-188, 1998.
[23] L.W. Schwartz "Hysteretic effects in droplet motion on heterogeneous substrates: direct numerical simulation", Langmuir, 14, 3440-3453, 1998.
[24] J.A. Diez, L. Kondic, A. Bertozzi "Global models for moving contact lines", Physical Review E, 63, 011208:1-011208:13, 2000.
[25] P.G. de Gennes "Wetting: statics and dynamics", Reviews of Modern Physics, 57, 827-863, 1985.
[26] W.D. Bascom, R.L. Cottington, C.R. Singleterry "Dynamic surface phenomena in the spontaneous spreading of oils on solids", Advances in Chemistry, 43, 355-379, 1964.
[27] R.P. Araujo, D.L.S. McElwain, "A History of the Study of Solid Tumour Growth: The Contribution of Mathematical Modelling", Bulletin of Mathematical Biology, 66, 1039-1091, 2004.
[28] H.M. Byrne, T. Alarcon, M.R. Owen, S.D. Webb, P.K. Maini, "Modelling Aspects of Cancer Dynamics: A Review", Philosophical Transactions of the Royal Society A, 364, 1563-1578, 2006.
[29] J.S. Lowengrub, H.B. Frieboes, F. Jin, Y-L. Chuang, X. Li, P. Macklin, S.M. Wise, V. Cristini, "Nonlinear Modelling of Cancer: Bridging the Gap Between

Cells and Tumours", Nonlinearity, 23, R1-R91, 2010.
[30] K.J. Brabazon, M.E. Hubbard, P.K. Jimack, "Nonlinear Multigrid Methods for Second Order Differential Operators with Nonlinear Diffusion Coefficient", Computers and Mathematics with Applications, 68, 1619-1634, 2014.