

# Towards a Three-Dimensional Parallel, Adaptive, Multilevel Solver for the Solution of Nonlinear, Time-Dependent, Phase-Change Problems

J.R. Green\*, P.K. Jimack\*, A.M. Mullis<sup>+</sup> and J. Rosam\*<sup>+</sup>

\*School of Computing, University of Leeds, UK

<sup>+</sup>School of Process, Environmental & Materials Engng., University of Leeds, UK

## Abstract

This paper describes on-going research into the development of new computational algorithms and software which combine parallelism, space and time adaptivity, implicit stiff solvers, and multigrid methods. Such software is necessary in order to be able to attempt three-dimensional simulations of fundamental solidification phenomena at physically relevant parameter values. The research builds upon our prior work which successfully combined implicit stiff solvers with adaptivity and multigrid in order to simulate a class of two-dimensional solidification problems at physically realistic parameter values for the first time. The extension to three-dimensions will allow a much larger range of problems to be studied, however it requires a massive increase in computational resources which can only be delivered by the successful inclusion of efficient parallel algorithms, whilst also maintaining the use of both adaptivity and optimal multilevel solvers.

**Keywords:** phase-change problems, solidification, phase-field models, nonlinear partial differential equations, stiff systems of equations, multiscale problems, parallel computing, adaptive methods, multigrid methods.

## 1 Introduction

One of the fundamental open challenges that computational engineering practitioners continue to face is that of combining modern numerical algorithms, such as mesh adaptivity and efficient multilevel solvers, with a parallel implementation that permits scalable performance on large numbers of cores. The fundamental problems that must be overcome include the need to maintain an effective load balance, that responds dynamically to local mesh refinement and/or coarsening, and the difficulty of obtaining good parallel efficiencies for those operations that take place at the coarsest levels of a multilevel solver.

In this paper we report on our ongoing research which seeks to make progress towards resolving some of these challenges through the application of a parallel, adaptive, multilevel approach to the solution of a highly challenging class of phase-change problems involving a very wide range of length and time scales. The mathematical models that we seek to solve take the form of systems of highly nonlinear, stiff, parabolic partial differential equations (PDEs) in three space dimensions. The solutions of these PDEs require extremely high spatial resolution near to a moving interface and so mesh refinement is essential. Furthermore, the stiffness of the systems implies that implicit time-stepping must be used and the resulting nonlinear algebraic systems are solved using a nonlinear multigrid approach on these locally refined meshes. Further details of this challenging computational engineering problem are provided in the remainder of this section, and a brief introduction to our previous work is given in Section 2 below.

## 1.1 Solidification and Dendritic Growth

The modelling of solidification microstructures has become an area of intense interest in recent years. The properties of large-scale cast products are strongly influenced by the physics of processes occurring on the microscopic and mesoscopic lengths scales. One of the most fundamental and all pervasive microstructures produced during solidification is the dendrite. Remnants of these dendritic microstructures often survive subsequent processing operations such as rolling and forging and the length scales established by the dendrite can influence not only the final grain size but also the micro- and hence the macro-segregation patterns.

Theoretical and experimental studies of the development of dendrites have been numerous in recent years (e.g. [4, 5, 8, 10, 22, 25, 42]), and one of the most powerful techniques to emerge for modelling dendritic microstructures is the phase-field method. The novelty of the phase-field method is that the mathematically sharp interface between the solid and liquid phases is assumed diffuse, allowing the definition of a continuous (differentiable) order parameter,  $\phi$ , which represents the phase of the material. The evolution of  $\phi$  is governed by a free energy functional which can be solved using standard techniques for PDEs without explicitly tracking the solid-liquid interface, thus allowing the simulation of arbitrarily complex morphologies. However, in order obtain realistic solidification structures the width of the diffuse interface must be much smaller than the smallest structural feature to be resolved. Consequently, phase-field simulations usually require very fine meshing, but equally are ideally suited to adaptive mesh refinement.

Notable examples of the successful use of the phase-field approach in the simulation of the formation of dendritic structures include [16, 21, 23, 24, 30, 41, 43] and, more recently, [11, 14, 17, 18], in three dimensions. The earlier of this work was typically restricted to the use of uniform grids however the use of adapting grids has now been shown to be both feasible and advantageous (e.g. [14, 30, 34, 35]). In this research we build upon our successful work in two space dimensions which led to the

first fully implicit, fully adaptive (in space and time) numerical tool for the solution to the coupled set of equations governing both the evolution of the phase-field and the transport of heat and mass [34].

Historically, phase-field models have been used to considered either pure thermal solidification (appropriate only to pure metals or alloys at solidification velocities such that solute trapping is complete) or isothermal alloy solidification where the growth rate is so low that the transport of heat is considered instantaneous. However, there is a range of solidification velocities where these assumptions do not hold and the solid grows under the coupled control of both heat and solute diffusion. Such conditions apply to a wide range of industrially important materials processing techniques such as laser welding, laser surface heat treatment, spray atomisation, twin-roll casting and melt spinning. Simulating solidification under these conditions is immensely challenging due to the vastly different length scales over which the two diffusion processes operate. Consequently, although a few papers have appeared recently using phase-field techniques to solve the non-isothermal alloy solidification problem, these have generally used artificially low values of the Lewis number (ratio of thermal to solutal diffusivity), typically in the range 1-200, [31]. By contrast for most metallic alloys this ratio should be of the order  $10^4$ .

The major advantage of the fully implicit approach is that stiff, highly nonlinear PDE systems, involving very different rates of thermal and solutal diffusion, may be accurately modelled in a computationally efficient manner. The key computational tool that permits this is the inclusion of a robust multigrid scheme that is able to solve the highly nonlinear discrete equations at each time step in a fast and reliable manner on sequences of locally refined grids. Consequently, the starting point for this work is our ability to solve the non-isothermal alloy solidification problem for realistic Lewis numbers in 2-dimensions on a relatively modest desktop workstation. Unfortunately such a capability, whilst still of great theoretical interest, is of limited practical value since almost all physically relevant solidification and dendrite growth phenomena are fully three-dimensional. Hence it is essential to extend the computational capabilities from two to three dimensions. Given that two-dimensional runs take many hours to complete on a single processor (for the physically realistic parameter regimes which are of interest) it is immediately apparent that any extension to three dimensions will require the use of multiple processors.

There has been significant amount of research into the combination of adaptive mesh refinement (in three dimensions) and an efficient parallel implementation in recent years, see for example, [2, 15, 27, 36, 39, 44]. The effect of modifying the computational mesh, via local refinement or coarsening, is to alter the load-balance across the parallel processors and so it is essential to couple such an implementation with a suitable dynamic load-balancing strategy, e.g. [19, 37, 40]. Alongside this work, there has also been a substantial amount of activity in the development of parallel multi-level solvers that allow the solution of linear and nonlinear algebraic systems, arising as the result of the discretization of PDEs, at an optimal computational complexity. Significant examples include [3, 12, 13, 20, 26]. The goal of efficiently combining all

three of adaptivity, multilevel solution and parallel implementation has received notably less attention in the literature however, presumably due to the inherent difficulty of the task. There are some exceptions to this, including the research of [1] which is restricted to steady-state problems, or the more general work of the UG toolbox, e.g. [6, 7]. In this research we make use of a more recent software tool which is described in [28], for example.

## 1.2 Overview

Having described the motivating problem for this work, and the need to move from two to three space dimensions, Sections 3 and 4 of this paper provide a detailed description of our on-going research. It is impossible to consider the move to three dimensions without the use of parallel computing, and so the extension is tightly coupled to the parallel algorithm and software decisions that are taken. Consequently, Section 3 focuses on these algorithms and tools. In particular, we describe an open source software library called PARAMESH [28] that has been used as the basis for our studies. This library allows parallel adaptivity and parallel multigrid but has had to be adapted to permit the combination of adaptivity and nonlinear multigrid that is required for this project. These adaptations are briefly described and the performance on a standard test problem is considered. Section 4 then goes on to describe the three-dimensional phase-field model that we are using for our initial development and some provisional computational results are provided for illustration. The paper then concludes with a discussion of our proposed future research, including consideration of the remaining computational challenges for us and for the community as a whole.

Before considering the parallel implementation in three dimensions however, the following section provides a summary of our existing technology for the solution of stiff, nonlinear PDE systems in two dimensions using implicit time-stepping, adaptivity and multigrid. Much of this work has already been published, see for example [34, 35], so the section is kept brief, focusing on the essential requirement for both adaptivity and multigrid.

## 2 Background

As described above, the phase-field method is one of the most popular and powerful techniques for the simulation of crystal growth in both pure materials and alloys. However, the mathematical realization of phase-field models leads to coupled systems of highly nonlinear and unsteady partial differential equations (PDEs). Typically, this complexity has led modellers to rely primarily on relatively simple numerical methods, however our prior work, such as [33, 34, 35], has shown that for many regions in parameter space it is advantageous, and often essential, to make use of advanced numerical methods such as adaptivity and implicit schemes. An illustration of some of these results is provided in this section, in which we make use of nonlinear multigrid

techniques in order to solve the nonlinear algebraic systems of equations that arise at each implicit time step.

## 2.1 Summary of Two-Dimensional Methodology

We illustrate the complexity of the governing PDEs by considering the coupled thermal-solute model for the simulation of microstructure in binary alloys that was first introduced in [32]:

$$\begin{aligned} \tau_0 A^2(\psi) \left[ \frac{1}{Le} + Mc_\infty [1 + (1 - k_E)U] \right] \frac{\partial \phi}{\partial t} = \nabla \cdot (W_0^2 A(\psi)^2 \nabla \phi) + \phi(1 - \phi^2) - \\ \lambda(1 - \phi^2)^2 (\theta + Mc_\infty U) - \\ \frac{\partial}{\partial x} \left( W(\psi) W'(\psi) \frac{\partial \phi}{\partial y} \right) + \frac{\partial}{\partial y} \left( W(\psi) W'(\psi) \frac{\partial \phi}{\partial x} \right), \quad (1) \end{aligned}$$

$$\begin{aligned} \left( \frac{1 + k_E}{2} - \frac{1 - k_E}{2} \phi \right) \frac{\partial U}{\partial t} = \nabla \cdot \left( \tilde{D} \frac{1 - \phi}{2} \nabla U + \frac{1}{2\sqrt{2}} [1 + (1 - k_E)U] \frac{\partial \phi}{\partial t} \frac{\nabla \phi}{|\nabla \phi|} \right) \\ + \frac{1}{2} [1 + (1 - k_E)U] \frac{\partial \phi}{\partial t}, \quad (2) \end{aligned}$$

$$\frac{\partial \theta}{\partial t} = \tilde{\alpha} \nabla^2 \theta + \frac{1}{2} \frac{\partial \phi}{\partial t}. \quad (3)$$

In the above system  $\phi$  is the phase variable,  $\psi = \arctan(\phi_y/\phi_x)$  the angle between the normal to the interface and the x-axis and  $A(\psi)$  is an anisotropy function (which controls the preferred direction of dendritic growth). Also,  $U$  and  $\theta$  represent solute concentration and temperature respectively, whilst  $\tilde{D}$  and  $\tilde{\alpha}$  are the solute and thermal diffusivity respectively: their ratio  $\tilde{\alpha}/\tilde{D}$  is called the Lewis number,  $Le$ . The numbers  $\tau_0$ ,  $Mc_\infty$ ,  $k_E$ ,  $W_0$  and  $\lambda$  are all assumed to be known. The cross-section of a typical solution with  $Le = 500$  is shown in Figure 1.

Figure 1 illustrates two important issues. The first of these is that the phase and solute variables change very rapidly over a small spatial region (which itself changes with time as the solidification front advances). The second issue concerns the importance of making the correct choice for the domain size. Since the boundary values are fixed (Dirichlet boundary conditions) and the temperature field is propagating so much further than the other variables one needs to choose sufficiently large domain sizes in order to prevent problems. If the temperature field reaches the boundary before it has decayed sufficiently then the temperature value in the interface region decreases, which will have a significant effect on the phase equation (1) and on the evolution of the microstructure. These two observations clearly demonstrate the need for adaptive mesh refinement since it would be exceptionally difficult to combine a very fine mesh near the moving interface with a large domain size without local mesh refinement.

Having demonstrated the need for adaptivity to provide very fine mesh resolution relative to the overall domain size, we now show the importance of using implicit

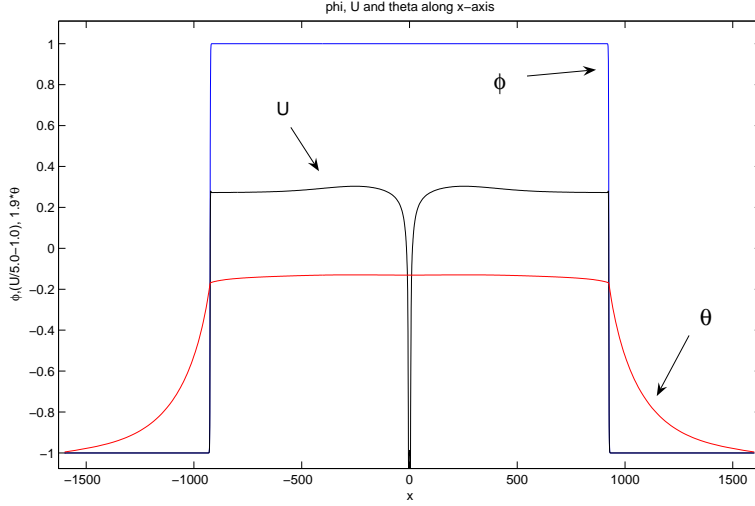


Figure 1: Typical solution values for  $\phi$ ,  $U$  and  $\theta$  along the  $x$ -axis for a simulation with  $Le = 500$

time stepping when solving problems with large values of the Lewis number  $Le$ . If we rewrite the equations (1)-(3) in operator form

$$\frac{\partial \phi}{\partial t} = F_\phi(t, \phi, U, \theta), \quad \frac{\partial U}{\partial t} = F_U(t, U, \phi, \frac{\partial \phi}{\partial t}), \quad \frac{\partial \theta}{\partial t} = F_\theta(t, \theta, \frac{\partial \phi}{\partial t}), \quad (4)$$

where  $F_\phi$ ,  $F_U$  and  $F_\theta$  are nonlinear differential operators, then the explicit Euler method has the following form

$$\phi^{k+1} - \phi^k = \Delta t F_\phi(t^k, \phi^k, U^k, \theta^k), \quad (5)$$

$$U^{k+1} - U^k = \Delta t F_U(t^k, U^k, \phi^k, \frac{\partial \phi^{k+1}}{\partial t}) \quad (6)$$

$$\theta^{k+1} - \theta^k = \Delta t F_\theta(t^k, \theta^k, \frac{\partial \phi^{k+1}}{\partial t}) \quad (7)$$

for  $k = 0, 1, 2, \dots$  and with  $\partial \phi^{k+1} / \partial t = (\phi^{k+1} - \phi^k) / \Delta t$ . The implementation of this scheme is relatively straightforward, even with adaptively refined spatial grids. A more robust time-stepping scheme is the implicit linear 2-step method that is known as BDF2. This is known to be A-stable and highly suited to the solution of stiff systems of ordinary differential equations. With a constant step size,  $\Delta t$ , this scheme takes the following form when applied to the phase-field equations:

$$\frac{1}{2}\phi^{k-1} - 2\phi^k + \frac{3}{2}\phi^{k+1} = \Delta t F_\phi(t^{k+1}, \phi^{k+1}, U^{k+1}, \theta^{k+1}), \quad (8)$$

$$\frac{1}{2}U^{k-1} - 2U^k + \frac{3}{2}U^{k+1} = \Delta t F_U(t^{k+1}, U^{k+1}, \phi^{k+1}, \frac{\partial \phi^{k+1}}{\partial t}) \quad (9)$$

$$\frac{1}{2}\theta^{k-1} - 2\theta^k + \frac{3}{2}\theta^{k+1} = \Delta t F_\theta(t^{k+1}, \theta^{k+1}, \frac{\partial \phi^{k+1}}{\partial t}) \quad (10)$$

for  $k > 1$ , with  $\partial\phi^{k+1}/\partial t = (\frac{1}{2}\phi^{k-1} - 2\phi^k + \frac{3}{2}\phi^{k+1})/\Delta t$ . The first order implicit Euler method is typically used for the first time step ( $k = 1$ ).

When a second order finite difference stencil is used for the spatial discretization, and the MLAT multigrid approach (see the following section for a discussion of multigrid) is used to solve the nonlinear algebraic systems that arise at each time step for the BDF2 scheme, then Figure 2 provides a comparison between the two time-stepping methods considered (using a typical set of parameter values). For the explicit Euler scheme the maximum stable time-step size is shown on a mesh with 12 refinement levels as the Lewis number is varied. For the BDF2 scheme, the largest time step is shown for which the nonlinear solver converges as the Lewis number is varied (using the same spatial mesh). It can clearly be seen that, on the mesh, BDF2 allows the time step to be hundreds of times larger and that this ratio increases as the Lewis number increases. Although not illustrated here, it is also the case that the ratio increases as the number of refinement levels in the spatial mesh goes up.

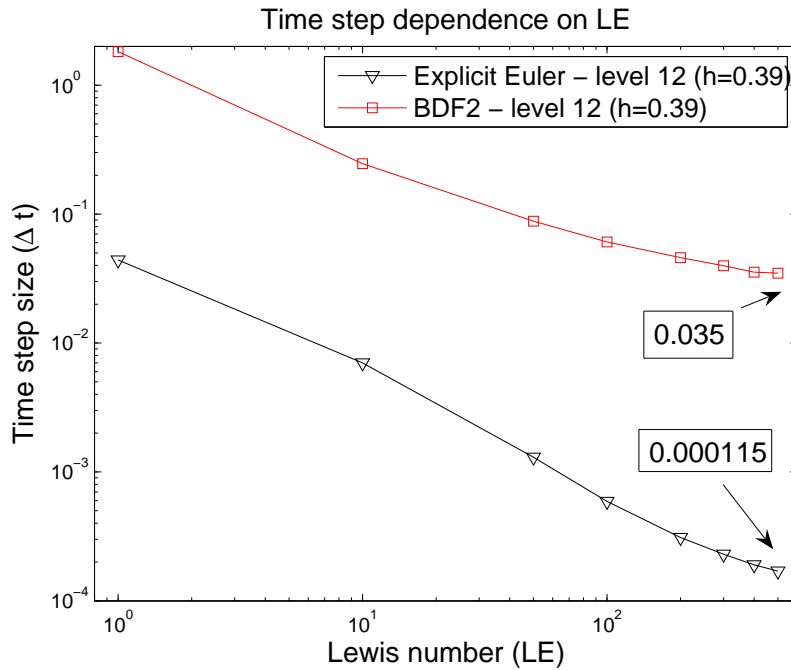


Figure 2: Comparison of maximum time steps for the implicit BDF2 method and the explicit Euler method for different Lewis numbers

## 2.2 Two-Dimensional Results

We now provide an illustration of the power of the fully implicit, fully adaptive multigrid approach by presenting some typical computational solutions. Further details of all of these solutions, as well as many more computational examples, may be found in [33].

The first set of results are for a moderate Lewis number of 40, for which it is possible to make comparisons against schemes which use explicit techniques to integrate the nonlinear terms forward in time, e.g. [31]. Figure 3 illustrates solutions at a relatively large time, starting with an initially circular seed at the centre of the computational domain, using parameters consistent with [31]. The top picture shows the solute concentration field (this is a slightly re-scaled version of  $U$  that is consistent with [31]), whilst the bottom picture shows the thermal field.

The second set of results that we present here are for a much larger Lewis number ( $Le = 500$ ), for which it would be computationally prohibitive to obtain simulation results without the use of implicit time-stepping and spatial adaptivity. The calculations have been performed using the same simulation parameters as in the previous case except that the domain size has been doubled in each direction (and the Lewis number has been increased of course). Full details may be found in [33]. In Figure 4 (top) one can see the mesh at the final time-step where 12 levels of refinement have been used. The mesh in this figure consists of 447727 nodes and is colour-coded in order to see the levels of mesh refinement. For resolving the steep gradients of the phase variable and the concentration in the interface region a very thin region of high mesh density is obtained. The phase variable at the final time step can be seen in Figure 4 (bottom).

### 2.3 Limitations

The simulation capability described in this section is of great value since the use of adaptivity and implicit time-stepping permits a number of problems to be investigated that would otherwise not be computationally feasible. In particular, we are able to undertake simulation of the solidification of a binary alloy for which the ratio of the thermal to the solutal diffusivities is large. Such systems require large computational domains, very fine spatial resolution at the interface and lead to extremely stiff systems of ordinary differential equations once the spatial discretization has been implemented. This new capability has been used in recent work, such as [35] for example, to investigate the behaviour of physical quantities such as dendrite tip radius and growth velocity as a function of the undercooling at large values of the Lewis number.

Unfortunately, despite the significant value of the two-dimensional simulation capability that we have described, in terms of helping to gain a deeper physical understanding of solidification processes, undertaking simulations in two space dimensions will always be somewhat inadequate. This is quite simply due to the fact that three-dimensional effects must always be present during any physical solidification process and will therefore influence key quantities such as orientation, growth velocity and tip geometry. In order to overcome this fundamental limitation, our current work seeks to implement a three-dimensional phase-field solver that extends the approach described in this section. For the reasons illustrated here, it will still be necessary to use mesh adaptivity and implicit time-stepping however, in order to gain access to the additional memory and cpu time that will be required, it will also be necessary to make use of parallel computing algorithms and environments. The remainder of this paper

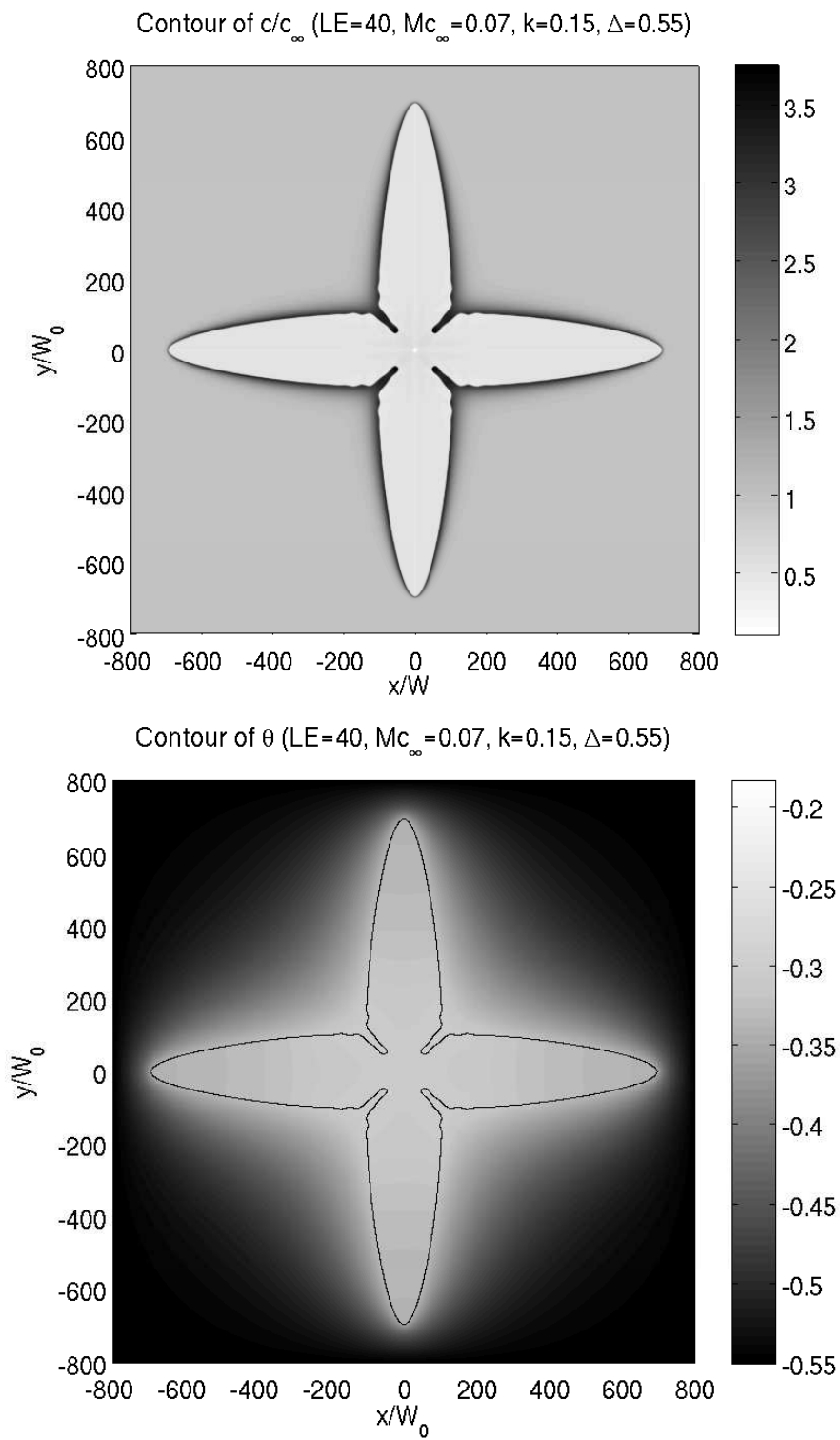
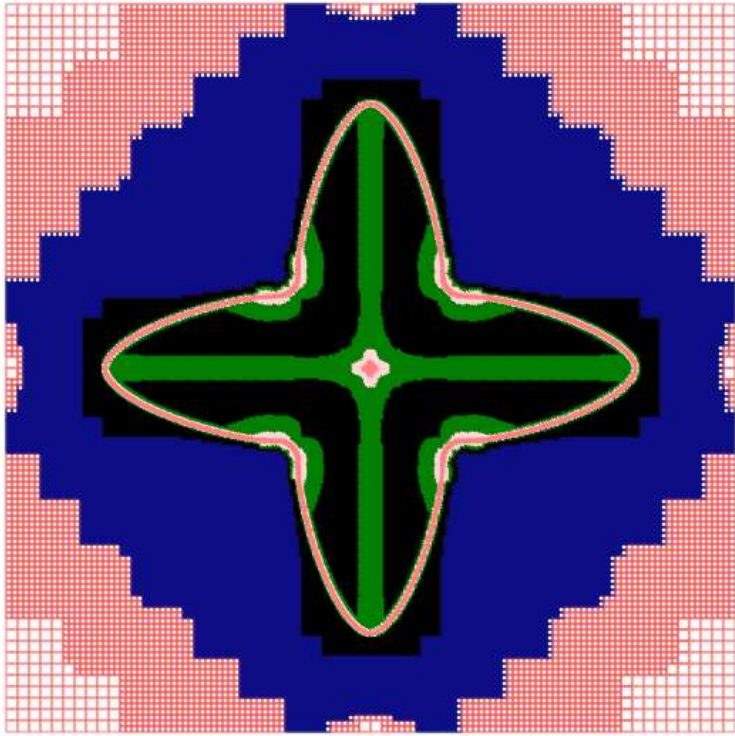


Figure 3: Snapshot of the contours of the concentration (top) and temperature (bottom) fields for a typical computation at  $Le = 40$



Contour of  $\phi$  ( $Le=500$ ,  $Mc_\infty=0.01$ ,  $k=0.15$ ,  $\Delta=0.55$ )

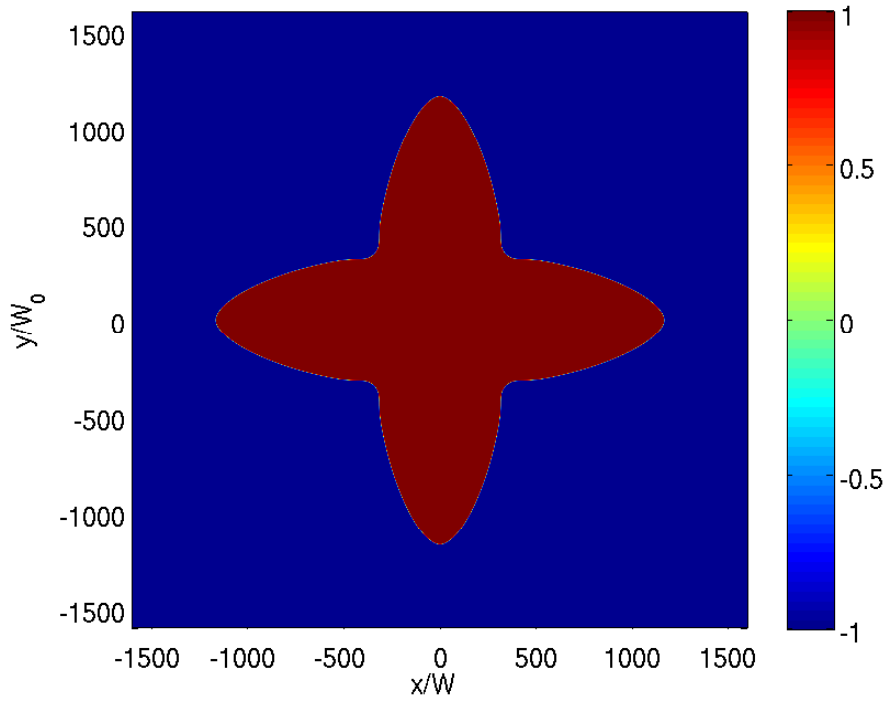


Figure 4: Snapshot of the adaptive mesh (top) and the phase field (bottom) typical computation at  $Le = 500$

discusses our progress towards this goal.

### 3 Three-Dimensional Problems Using PARAMESH

PARAMESH [28, 29] is a collection of functions designed to sit around an adaptive numerical solver in up to three spatial dimensions. The aim of these functions is to allow seamless use of parallel processing. In this section the component parts of PARAMESH which are used during the course of this study are discussed and explained. This includes a basic overview of how PARAMESH handles data, which is then extended to discuss how the data storage method lends itself to parallelism and local adaptivity. The use of multigrid within PARAMESH and the extensions required in order to run this in parallel with local adaptivity (MLAT) are explained. The section then moves on to discuss two test problems and to present some of the numerical results that have been obtained. PARAMESH can be used in one, two or three dimensions, however because the parts of this study which made use of PARAMESH were aimed at a three dimensional phase-field solver using a cell centered scheme, this introduction mainly covers three dimensional implementations using cell-centered data structures.

#### 3.1 Introduction to PARAMESH

A key part of PARAMESH is the use of data blocks, these are collections of data cells. A data cell can contain cell-centered, vertex-centered, edge-centered or face-centered data points, or a mixture. Each of the different types of data point is stored in a different array, for instance cell centered data is stored within *unk* while vertex centered data is stored within *unk\_n*. Every data block within a simulation is identical in terms of structure (the number of data cells), and from the programmer's point of view consists of a collection of "real" data cells and "guardcells". The guardcells are cells around the edge of each block which hold data from other blocks, they are not acted upon computationally when updating the current block. It is the use of this block structure which allows the seamless use of parallelism: the programmer need not worry as to whether two neighbouring blocks are stored by the same process or not, the guardcell update algorithm handles the updating of the guardcells and any necessary parallel communication. Regarding the guardcells, the only thing that the programmer needs to consider is how to handle the guardcells for the blocks at the edge of the domain where the boundary condition must be implemented.

In addition to holding data each block has several variables associated with it. Firstly there is a *nodetype* variable, this defines what type of nodes are on the block: whether they are "leaf" nodes, "parent" nodes or higher (corresponding to nodetypes 1, 2 or 3 respectively). The refinement level of the data on a block is stored in *refine*. Other useful pieces of information are the physical coordinates of the block and the physical size of the block. Finally the address of each of the block's neighbours (or

the index for the physical boundary of the domain) is stored in the *neigh* array.

As well as the real space arrays there are workspace arrays, these are a copy of the real space data structure (for every real space block there is a work space block), however they can have a different number of variables in each cell and/or a different number of guardcells. There are several motivations for using workspace arrays within PARAMESH. Firstly there are many inbuilt functions which will operate on all the real space arrays (eg. guardcell or prolongation), if a copy of the real space data is required it is much more efficient to use the workspace rather than define an extra real space variable. A second motivation is very closely linked to the first: rather than use the workspace for data which one does not want a function to operate on, one can use workspace for data that one does want a function to operate on. An example of this is if only one variable needs its guardcells updating, rather than update all the variables, copy the data for the one variable into the workspace and call the guardcell routine on workspace, then copy the data back.

### 3.2 Parallel Adaptivity

The block structure within PARAMESH is specifically designed to support local adaptivity in parallel. If refinement is carried out a block will refine into eight new blocks (in three dimensions). Additionally, coarsening or derefinement may be carried out and the supplied functions ensure that neighbouring blocks refine/derefine consistently. All the programmer need provide is a condition for coarsening and a condition for refining. A simple example of this would be to find the maximum value of some error function on a block, then test this value against refinement and derefinement tolerances to decide whether a block qualifies for refinement or derefinement. The pseudo-code in Figure 5 demonstrates this.

This pseudo-code gives a demonstration of how the refinement flags are set, once this is done the *refine\_derefine* subroutine is called which in turn calls all the required subroutines to create new data blocks (or release data blocks in the case of derefinement) and any required restriction/prolongation of data, which can be done using either injection, linear, quadratic or user-supplied interpolation functions. The selection between these interpolation functions is carried out by setting the values of the *interp\_mask* array. Additionally subroutines are called which carry out load balancing and attempt, where possible, to ensure that neighbouring blocks are on the same process.

### 3.3 Parallel Multigrid

Creating a parallel multigrid solver within PARAMESH is actually no more difficult than creating a single process multigrid solver. The only place where communication occurs between processes is in the restriction/prolongation algorithms and within the guardcell algorithm, all of which are provided in a form which there is no need to alter. To enable multigrid within PARAMESH a second copy of the inbuilt restriction and

```

loop over all blocks
if block nodetype is 1 or 2
error_max = 0.0
loop over all points on block
error = error_function()
if error > error_max
error_max = error
end if
end loop
if error_max > refinement tolerance
refine_flag = true
else if error_max < derefinement tolerance
derefine_flag = true
end if
end if
end loop

```

Figure 5: Pseudo-code for a PARAMESH test refinement algorithm

prolongation algorithms are provided. Rather than operating upon the real data arrays they operate upon the workspace data. This is necessary because when using the coarse grid correction method ([38]) the old data is corrected rather than overwritten. These routines make multigrid possible within PARAMESH however for an actual implementation they must be surrounded by other transfer and solver operations to create a full v-cycle. The v-cycle algorithm is recursive and an example for a linear v-cycle for the problem  $[L](\underline{v}) = \underline{f}$ , where  $[L]()$  is the discrete linear operator,  $\underline{v}$  is stored in  $unk(1)$  and  $\underline{f}$  is stored in  $unk(2)$  is shown in Figure 6.

The scheme shown in Figure 6 will only work for a linear problem. However, the phase-field problem which is intended to be solved using PARAMESH is nonlinear and so this linear scheme needs to be extended to incorporate the FAS scheme ([38]). If the nonlinear problem is written as  $[N](\underline{v}) = \underline{f}$  (with  $\underline{v}$  and  $\underline{f}$  as before and  $[N]()$  denoting the discrete nonlinear operator) then the v-cycle algorithm can be written as shown in Figure 7 In this figure, the new line 8 is where the modified right-hand side is created to allow for non-linear problems, and the modified line 10 reflects that we are solving for a coarse grid solution, rather than a coarse grid error, in the FAS scheme.

### 3.4 The MLAT Scheme

The MLAT scheme ([9]) provides a way to handle grids containing local adaptivity using multigrid. In this scheme the locally refined regions are consider grids in their own right and smoothing occurs upon them in the same manner as it does on a normal non-adapted region. The edges between locally refined regions and coarser grids are considered to be Dirichlet boundary nodes when considering the more refined grid. Smoothing is carried out on this fine grid, then the values are interpolated onto the

1. Smooth  $unk(1)$ ,  $\nu_1$  times.
2. Update guardcells.
3. Calculate Defects.
4. Copy  $unk(1)$  into  $work(1)$  and defects into  $work(2)$ .
5. Restrict  $work$  array.
6. Update guardcells.
7. Copy  $work(1)$  to  $unk(1)$  and  $work(2)$  to  $unk(2)$ .
8. Call v-cycle for the new level.
9. Copy  $unk(1)$  into  $work(1)$ .
10. Prolong  $work$  array.
11. Update guardcells.
12. Correct:  $unk(1) \rightarrow unk(1) + work(1)$ .
13. Smooth  $unk(1)$ ,  $\nu_2$  times.
14. Update guardcells.

Figure 6: A scheme for linear multigrid within PARAMESH

next grid in the hierarchy and a correction scheme is applied.

It is quite simple to extend the nonlinear multigrid scheme to account for local adaptivity (by way of the MLAT scheme), because in effect it is already an MLAT scheme. The problems occur however because PARAMESH only carries out guardcell updates upon blocks with *nodetype* 1 or 2 and only transfers data between blocks with the same type. This is not a problem for a non-multigrid solver with local adaptivity, any block which is the most refined in a particular physical space (i.e. blocks which store the current solution) would be *nodetype* 1 and thus would exchange data correctly during the guardcell update. The problems occur because during the multigrid cycle the blocks which hold the current solution (or in this case the current blocks being operated on) will change, this means that the blocks upon which guardcell operates must also change. This change is handled by the PARAMESH multigrid algorithms in the case where uniform grids are used, however these routines were not written with the MLAT scheme in mind. This means that in order to handle non-uniform grids a new subroutine is required to reset the *nodetype* variable for all blocks from the value set by the multigrid restrict/prolong algorithms to a value which will cause guardcell to act correctly. The motivation for producing this additional code is to provide subroutines which sit around the existing PARAMESH subroutines rather than to modify the PARAMESH subroutines themselves. A schematic for this reset algorithm applied to a single *block* for a particular refinement *level* is shown in Figure 8.

It should be noted that the “has children” (more refined blocks occupying the same physical space) condition is fairly simple to test, since if a block has any children it must have the maximum number. The condition on being a grandparent is slightly

1. Smooth  $unk(1)$ ,  $\nu_1$  times.
2. Update guardcells.
3. Calculate Defects.
4. Copy  $unk(1)$  into  $work(1)$  and defects into  $work(2)$ .
5. Restrict  $work$  array.
6. Update guardcells.
7. Copy  $work(1)$  to  $unk(1)$  and  $unk(3)$ .
8. Set  $unk(2) = work(2) + [N](unk(1))$
9. Call v-cycle for the new level.
10. Set  $work(1) = unk(1) - unk(3)$ .
11. Prolong  $work$  array.
12. Update guardcells.
13. Correct:  $unk(1) \rightarrow unk(1) + work(1)$ .
14. Smooth  $unk(1)$ ,  $\nu_2$  times.
15. Update guardcells.

Figure 7: A scheme for non-linear multigrid using FAS within PARAMESH

more complex, since for this to be true all of a block’s children must also have children. This routine, when applied to all blocks, reconstructs the *nodetype* data tree for a specific *level* such that multigrid and guardcell can work together. Additionally this updated data must be communicated between the various processes which is handled by “`amr_get_new_nodetypes`”.

### 3.5 Test Problems

The test problem that we present here for our PARAMESH MLAT implementation is an adapted version of the PARAMESH tutorial ([29]). This was adapted to three dimensions, rather than the two used in the tutorial, additionally the initial seed was moved off center. This problem was chosen because it is a simple diffusion problem and, based upon the tutorial, it was known that it would show local adaptivity.

To create this new version of the test problem, rather than having the grid adapt itself during run time, the grid was adapted before the first timestep and then run for a fixed amount of time. Figure 9 shows a snapshot of the output for this test problem, where the locally adapted section of the grid is easily visible.

This test problem was compared against a uniformly adapted multigrid code, and the explicit code from the tutorial (which had the same locally adapted region). While there were very small differences between each simulation in terms of raw numbers the differences cannot be seen on a graphical output apart from the differences in the grid where local adaptivity was present and when it was not. The quantitative

1. If *block* is more refined than *level* then
2. *nodetype*  $\rightarrow$  1
3. Else if *block* has the same refinement level as *level* then
4. *nodetype*  $\rightarrow$  1
5. Else if *block* is coarser than *level* and *block* has no children then
6. *nodetype*  $\rightarrow$  1
7. Else if *block* is one level coarser than *level* and *block* has children then
8. *nodetype*  $\rightarrow$  2
9. Else if *block* is coarser than *level* and *block* has children and *block* is not a grandparent then
10. *nodetype*  $\rightarrow$  2
11. Else if *block* is coarser than *level* and *block* is a grandparent then
12. *nodetype*  $\rightarrow$  3

Figure 8: A scheme for resetting of the *nodetype* variable within PARAMESH

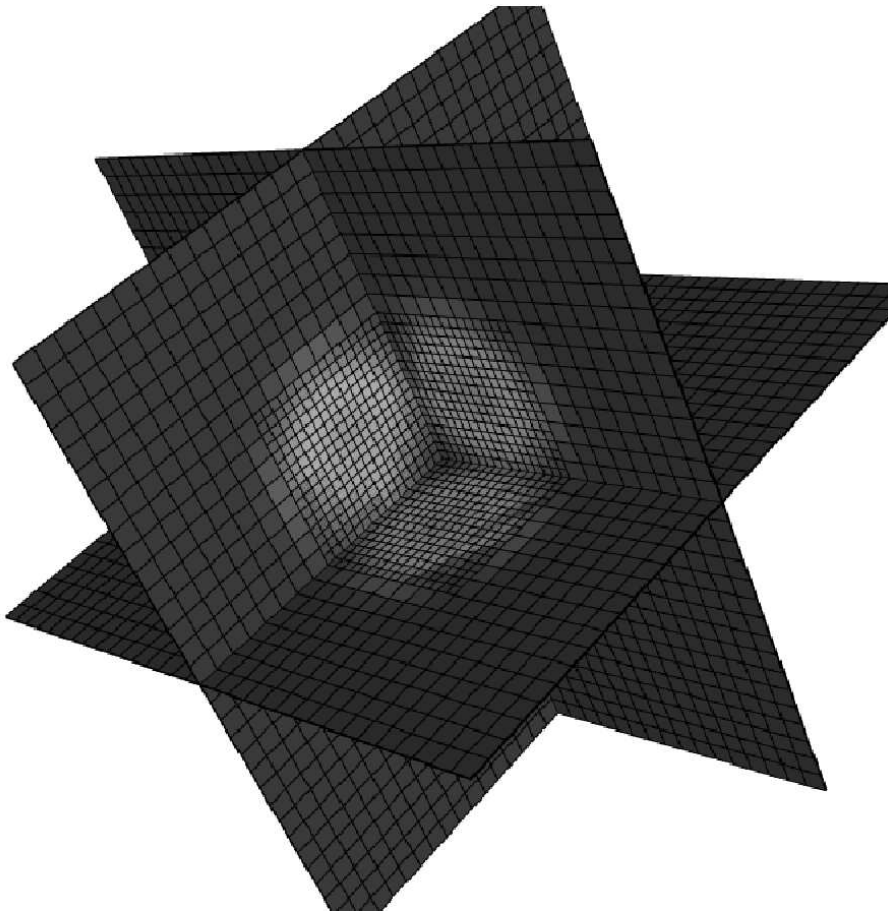


Figure 9: Output snapshot from a PARAMESH test problem

differences were therefore not a concern.

Additionally tests were run to find out how well PARAMESH performs as the number of processes varies. A second test problem was designed for this because the test problem presented in Figure 9 is relatively simple and only takes a short time to run. The new test problem was designed specifically to use more memory and to take a longer time to solve than the initial problem, and this problem was run as a 1, 2 and 4 process job. Because this test was being run on a single computer with 4 cores (in a 2 processor, each with 2 cores, configuration) additional runs were performed with multiple instances of a single process program. This was carried out to provide an estimate of the slow down incurred due to competition between processes for the resources on the one machine. One feature of PARAMESH is that the amount of memory per process is set by the user rather than dynamically, which means that, for these runs, each of the four parallel processes use the same amount of memory as the single process job and that when multiple instances of the single process job are running they will use the same amount of memory in total as the parallel jobs. Tables 1 and 2 presents the results for these scalability tests.

Nprocs	Serial	Parallel	Efficiency 1	Efficiency 2
1	485	-	1	1
2	487	286	0.85	0.85
4	507	192	0.63	0.66

Table 1: PARAMESH Parallel scalability for test problem 1

Nprocs	Serial	Parallel	Efficiency 1	Efficiency 2
1	1814	-	1	1
2	1821	1079	0.84	0.84
4	1840	674	0.67	0.68

Table 2: PARAMESH Parallel scalability for test problem 2

The column headings in Tables 1 and 2 are as follows. Nprocs is the number of cores in use, in the case of the serial code this means the number of instances, in the case of the parallel code this means the number of processes working upon the job. Serial lists the time for the serial code runs and Parallel lists the time for the parallel runs, both of which are in seconds. The efficiency is measured by

$$E = \frac{t_s}{t_p * Nprocs},$$

where  $t_s$  and  $t_p$  are the execution times for the serial and parallel codes respectively. Efficiency 1 uses the 1 core time for  $t_s$  while Efficiency 2 uses the time for the 1, 2 or 4 core appropriately. This is appropriate in order to negate the effect of competition

between the processes for the resources of the one machine. In the case of both of these tests the grids were uniform. A scalability test for locally adapted grids is presented within the phase field results in Section 4.3.

## 4 Three-Dimensional Phase-Field

The motivation to extend phase-field modelling from two dimensions to three is clear, the third dimension adds significant capability for simulating real world systems. The complexity of doing this, both in terms of the model equations themselves and in terms of the computational work increase brought about by the third dimension, can not be understated. The introduction of PARAMESH is a direct result of this complexity, attempting to solve a three dimensional phase-field problem as a single process job on a uniform grid would be beyond the expectations of current computing ability. Even with just local adaptivity or just parallelism this problem is extremely complex and is unlikely to yield noteworthy results in a reasonable amount of time. It is this combination of issues which has resulted in the use of PARAMESH to bring together both parallel processing and local adaptivity to make this problem tractable on the resources currently available.

### 4.1 Computational Model

The three dimensionally phase field model considered here is an extension of the two dimensional one used in Section 2.1 however, for initial simplicity, it is for a pure metal only ( hence no concentration variable or terms based on this). The governing equations for this model are therefore given by Equations 11 and 12.

$$\begin{aligned} \tau(\psi, \beta) \frac{\partial \phi}{\partial t} &= \nabla \cdot (W(\psi, \beta)^2 \nabla \phi) + \phi(1 - \phi^2) - \lambda \theta (1 - \phi^2)^2 \\ &+ \frac{\partial}{\partial x} \left( W(\psi, \beta) \left[ -\frac{\partial W(\psi, \beta)}{\partial \psi} \frac{\phi_y |\nabla \phi|^2}{\phi_x^2 + \phi_y^2} + \frac{\partial W(\psi, \beta)}{\partial \beta} \frac{\phi_z \phi_x}{\sqrt{\phi_x^2 + \phi_y^2}} \right] \right) \\ &+ \frac{\partial}{\partial y} \left( W(\psi, \beta) \left[ \frac{\partial W(\psi, \beta)}{\partial \psi} \frac{\phi_x |\nabla \phi|^2}{\phi_x^2 + \phi_y^2} + \frac{\partial W(\psi, \beta)}{\partial \beta} \frac{\phi_z \phi_y}{\sqrt{\phi_x^2 + \phi_y^2}} \right] \right) \\ &- \frac{\partial}{\partial z} \left( W(\psi, \beta) \frac{\partial W(\psi, \beta)}{\partial \beta} \sqrt{\phi_x^2 + \phi_y^2} \right) \end{aligned} \quad (11)$$

$$\frac{\partial \theta}{\partial t} = \alpha \nabla^2 \theta + \frac{1}{2} \frac{\partial h(\phi)}{\partial t}. \quad (12)$$

Here  $\lambda$  is a coupling parameter between the thermal and phase fields, and the functions  $\tau(\psi, \beta)$ ,  $W(\psi, \beta)$  and  $h(\phi)$  are given by

$$\begin{aligned} W(\psi, \beta) &= W_0 (1 + \epsilon (\cos^4 \beta + \sin^4 \beta (1 - 2 \sin^2 \psi \cos^2 \psi))) \\ \tau(\psi, \beta) &= \tau_0 (1 + \epsilon (\cos^4 \beta + \sin^4 \beta (1 - 2 \sin^2 \psi \cos^2 \psi)))^2 \\ h(\phi) &= \phi, \end{aligned}$$

where  $\phi_x = \frac{\partial\phi}{\partial x}$ . In terms of modifications from the two dimensional model, the only change is to add a second angle ( $\beta$ ) to the function  $W(\psi)$ . This makes the phase equation significantly more complex due to the extra differentials when the spatial derivatives are found. The thermal equation is unchanged apart from the addition of an extra dimension to the spatial differential.

## 4.2 Methodology

To solve Equations (11) and (12) implicitly a backward Euler scheme is used:

$$\begin{aligned}\frac{\partial\phi_i}{\partial t} &= f_\phi(\underline{\phi}^{n+1}) \\ \phi_i^{n+1} - \phi_i^n &= \delta t \times f_\phi(\underline{\phi}^{n+1}) \\ \phi_i^{n+1} - \delta t \times f_\phi(\underline{\phi}^{n+1}) &= \phi_i^n,\end{aligned}\tag{13}$$

where  $f_\phi(\phi^{n+1})$  can be found by rearranging Equation (11) and a similar equation can be used for the thermal variable  $\theta$ . In order to obtain a point-wise smoother for the nonlinear multigrid solver a Newton approximation is applied as follows, to give an updated approximation to  $\phi^{n+1}$ :

$$F_{\phi_i} = \phi_i^{n+1} - \delta t \times f_\phi(\underline{\phi}^{n+1}) - \phi_i^n = 0\tag{14}$$

$$\tilde{\phi}_i^{n+1} = \phi_i^{n+1} - w \frac{F_{\phi_i}}{\frac{\partial F_{\phi_i}}{\partial \phi_i^{n+1}}},\tag{15}$$

where  $w$  is a weighting parameter, typically between 0.6 and 0.9. This approximation is applied to each  $\phi_i$  in turn, followed by each  $\theta_i$ , which is approximated in a similar manner. This process is the smoothing mentioned in the non-linear multigrid scheme in Figure 7. This smoothing process could be applied repeatedly until the defects were sufficiently small such that the equation could be considered solved, i.e. when  $F_{\phi_i}$  and  $F_{\theta_i}$  are below a certain tolerance for all  $\phi_i$  and  $\theta_i$ . With multigrid however this process is performed much faster by using a hierarchy of coarser grids to find better approximations to  $\phi_i^{n+1}$  and  $\theta_i^{n+1}$  faster than is possible using just the finest grid.

## 4.3 Provisional Results

The results obtained so far are still quite provisional, this is due to the ongoing development of the solver within PARAMESH. The results presented in Figure 10 show a phase-field simulation utilising local adaptivity. It should be noted that this simulation used only an isotropic implementation of the three dimensional phase-field equation, which is obtained by setting  $\tau(\psi, \beta)$  and  $W(\psi, \beta)$  to  $\tau_0$  and  $W_0$  respectively and all their derivatives to zero. The result of this is that rather than exhibiting dendritic growth, spherical growth should be visible.

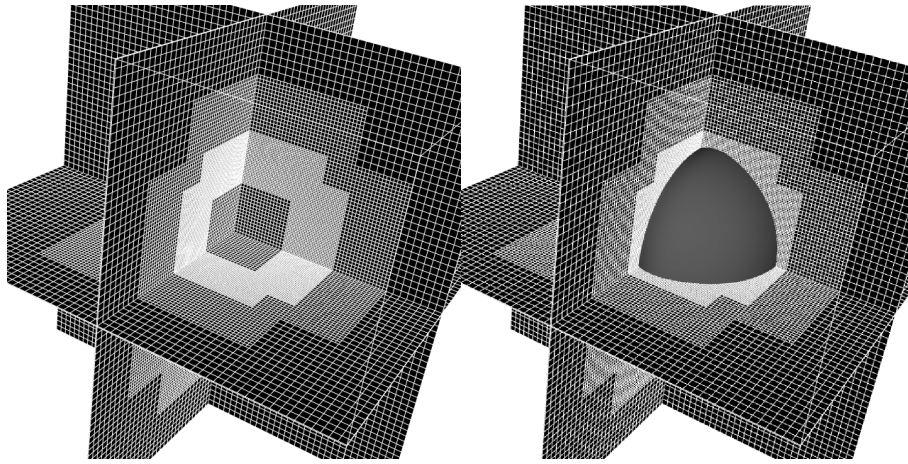


Figure 10: PARAMESH three dimensional phase field output showing the locally adapted grid and solid contour ( $\phi = 0.0$ )

The results shown in Figure 10 demonstrate the phase field model operating with local adaptivity (including the derefinement of the central region). Additional testing was carried out to assess how well this performed when the number of processes available increases from 1 to 4 in the same manner as the tests that were carried out for Tables 1 and 2. This set of results is presented in Table 3.

Nprocs	Serial	Parallel	Efficiency 1	Efficiency 2
1	1546	-	1	1
2	1552	849	0.91	0.91
4	1584	539	0.72	0.73

Table 3: PARAMESH Parallel Scalability Test for the Phase-field Problem

The results presented in Table 3 are for a significantly more complex problem than those in Tables 1 and 2, this third test was for a two equation system, which included local adaptivity and a dynamically changing grid. This second feature is important since this requires PARAMESH to carry out load balancing during the simulation itself rather than just at the beginning (as was the case for the first two tests). The efficiencies listed in Table 3 are actually higher than the previous two tests, this demonstrates that as the problem becomes increasingly complex the scope for parallelism is enhanced.

## 5 Discussion

This section provides an brief overview of the work presented as well as suggesting the direction that this ongoing research will take in the future.

## 5.1 Summary

During the course of this study we have presented our ongoing development of the PARAMESH library to extend this to handle a locally adapted multigrid solver. This solver has then been applied to a simple test problem in order to verify this additional functionality against both a uniformly adapted multigrid solver and a separate explicit solver. Additionally parallel scalability tests were performed to demonstrate how PARAMESH performs as the number of processes applied to a problem is varied. Preliminary results for the three dimensional phase-field solver have also been presented along with additional scalability tests for this more complex problem.

## 5.2 Future Work

Due to the ongoing development of this project the immediate extensions are relatively obvious: to complete the integration of the phase field problem into the MLAT scheme and to test this fully on increasing numbers of processes. Further development then splits into two distinct branches, firstly incorporating additional elements into the model, for instance moving to an isothermal binary alloy model, and then to a fully coupled thermal-solute-phase model would demonstrate fully the functionality provided by the PARAMESH MLAT code which has been developed. A second branch of development is to further improve the numerical side of the code: one of the simplest modifications is to incorporate a more complex adaptive time stepping scheme, rather than the standard backward Euler, such as the BDF2 scheme used for the two-dimensional work. At each stage of the development, the scalability to a larger number of processes will need to be considered.

## Acknowledgement

This work is supported by the Engineering and Physical Sciences Research Council, via grant EP/F010354/1.

## References

- [1] Bank, R E & Holst, M, A New Paradigm for Parallel Adaptive Meshing Algorithms. *SIAM J Sci. Comput.*, 22, 1411–1443, 2000.
- [2] Bank, R E & Jimack, P K, A New Parallel Domain Decomposition Method for the Adaptive Finite Element Solution of Elliptic Partial Differential Equations. *Concurrency and Computation*, 13, 327–350, 2001.
- [3] Bank, R E, Jimack, P K, Nadeem, S A & Nepomnyaschikh, S V, A Weakly Overlapping Domain Decomposition Preconditioner for the Finite Element Solution of Elliptic Partial Differential Equations. *SIAM J. Sci. Comput.*, 26, 1535–1554, 2002.

- [4] Barbieri, A & Langer, J S, Predictions of Dendritic Growth Rates in the Linearized Solvability Theory. *Phys. Rev. A*, 39, 5314–5325, 1989.
- [5] Bassler, B T, Hofmeister, W H, Carro, G & Bayuzick, R J, The Velocity of Solidification of Highly Undercooled Nickel. *Metall. Mater. Trans. A*, 25, 1301–1308, 1994.
- [6] Bastian, P, Lang, S & Eckstein, K, Parallel Adaptive Multigrid Methods in Plane Linear Elasticity Problems. *Numer. Linear Alebr.*, 4, 153–176, 1997.
- [7] Bastian, P & Lang, S, Couplex Benchmark Computations Obtained with the Software Toolbox UG: Simulation of Transport Around a Nuclear Waste Disposal Site. *Comput. Geosciences*, 8, 125–147, 2004.
- [8] Battersby, S E, Cochrane, R F & Mullis, A M, Microstructural evolution and growth velocity-undercooling relationships in the systems Cu, Cu-O and Cu-Sn at high undercooling. *J. Mater. Sci.*, 35, 1365–1373, 2000.
- [9] Brandt, A, Multi-Level Adaptive Solutions to Boundary-Value Problems. *Mathematics of Computations*, 31(138), 333-390, 1977.
- [10] Brener, E, Needle-Crystal Solution in 3-Dimensional Dendritic Growth. *Phys. Rev. Lett.*, 71, 3653–3656, 1993.
- [11] George, W L & Warren, J A, A Parallel 3D Dendritic Growth Simulator Using the Phase-Field Method. *J. Comput. Phys.*, 177, 264–283, 2002.
- [12] Griebel M & Zumbusch, G, Parallel Multigrid in an Adaptive PDE Solver Based on Hashing and Space-Filling Curves. *Parallel Comput.*, 25, 827–843, 1999.
- [13] Heise, B & Jung, M, Efficiency, scalability, and robustness of parallel multilevel methods for nonlinear partial differential equations. *SIAM J. Sci. Comput.*, 20, 553–567, 1998.
- [14] Jeong, J-H, Goldenfeld, N & Dantzig, J A, Phase Field Model for Three-Dimensional Dendritic Growth with Fluid Flow. *Phys. Rev. E*, 64, 041602, 2001.
- [15] Jimack, P K & Nadeem, S A, Parallel Application of an Optimal Domain Decomposition Preconditioner for Adaptive Finite Element Solution of Three-Dimensional Convection-Dominated PDEs. *Concurrency and Computation: Practice and Experience*, 15, 939–956, 2003.
- [16] Karma, A & Rappel, E-J, Phase-Field Method for Computationally Efficient Modeling of Solidification with Arbitrary Interface Kinetics. *Phys. Rev. E*, 53, 3017–3020, 1996.
- [17] Karma, A & Rappel, E-J, Phase-Field Simulation of Three-Dimensional Dendrites: Is Microscopic Solvability Theory Correct? *J. Crystal Growth*, 174, 54–64, 1997.
- [18] Karma, A & Rappel, E-J, Quantitative Phase-Field Modeling of Dendritic Growth in Two and Three Dimensions. *Phys. Rev. E*, 57, 4323–4349, 1998.
- [19] Karypis G, & Kumar, V, A parallel algorithm for multilevel graph partitioning and sparse matrix ordering. *J. Parallel Distr. Com.*, 48, 71–95, 1998.
- [20] Llorente, I M, Prieto-Matias, M & Diskin, B, A parallel multigrid solver for 3D convection and convection-diffusion problems *Parallel Computing*, 27, 1715–1741, 2001.
- [21] McFadden, G B, Wheeler, A A, Braun, R J, Coriell, S R & Sekerka, R F, Phase-

- Field Models for Anisotropic Interfaces. *Phys. Rev. E*, 48, 2016–2024, 1993.
- [22] Mello, M C A & Kiminama, C S, Undercoolability of Copper Bulk Samples. *J. Mater. Sci. Lett.*, 8, 1416–1417, 1989.
- [23] Mullis, A M, The effect of the ratio of solid to liquid conductivity on the stability parameter of dendrites within a phase-field model of solidification. *Phys. Rev. E*, 68, art. no. 011602, 2003
- [24] Mullis, A M & Cochrane, R F, A phase field model for spontaneous grain refinement in deeply undercooled metallic melts. *Acta Mater.*, 49, 2205–2214, 2001.
- [25] Mullis, A M, Cochrane, R F, Walker, D J & Battersby, S E, Deformation of dendrites by fluid flow during rapid solidification. *Mater. Sci. Eng. A*, 304-306, 245–249, 2001
- [26] Nadeem, S A & Jimack, P K, Parallel Implementation of an Optimal Two Level Additive Schwarz Preconditioner for the 3-D Finite Element Solution of Elliptic Partial Differential Equations. *Int. J. Num. Meth. Fluids*, 40, 1571–1579, 2002
- [27] Olike, L, Biswas R & Strawn, R C, Parallel Implementation of an Adaptive Scheme for 3D Unstructured Grids on the SP2. In *Parallel Algorithms for Irregularly Structured Problems*, LNCS 1117 (Springer-Verlag), 1996.
- [28] Olson, K, PARAMESH: A Parallel Adaptive Grid Tool. In *Parallel Computational Fluid Dynamics 2005: Theory and Applications*, ed. A. Deane et al. (Elsevier), 2006.
- [29] Olson, K, PARAMESH tutorial. [www.physics.drexel.edu/~olson/paramesh-doc/Users\\_manual/amr\\_tutorial.html](http://www.physics.drexel.edu/~olson/paramesh-doc/Users_manual/amr_tutorial.html), accessed 10 December 2008.
- [30] Provatas, N, Goldenfeld, N & Dantzig, J A, Adaptive Mesh refinement Computation of Solidification Microstructures Using Dynamic data Structures. *J. Comput. Phys.*, 148, 265–290, 1999.
- [31] Ramirez, J C & Beckerman, C, Examination of Binary alloy Free Dendritic Growth Theories with a Phase-Field Model. *Acta Materialia*, 53, 1721–1736, 2005.
- [32] Ramirez, J C, Beckerman, C, Karma, A & Diepers, H-J, Phase-Field Modelling of Binary Alloy Solidification with Coupled Thermal Heat and Solute Diffusion. *Phys. Rev. E*, 69, 051607, 2004.
- [33] Rosam, J, A Fully Implicit, Fully Adaptive Multigrid Method for Multiscale Phase-Field Modelling. PhD Thesis, University of Leeds, 2007.
- [34] Rosam, J, Jimack, P K & Mullis, M M, A Fully Implicit, Fully Adaptive Time and Space Discretisation Method for Phase-Field Simulation of Binary Alloy Solidification. *J. Comp. Phys.*, 225, 1271–1287, 2007
- [35] Rosam, J, Jimack, P K & Mullis, A M, An Adaptive, Fully Implicit, Multigrid Phase-Field Model for the Quantitative Simulation of Non-isothermal Binary Alloy Solidification. *Acta Mat.*, 56, 4559–4569, 2008.
- [36] Selwood, P, & Berzins, M, Parallel unstructured tetrahedral mesh adaptation: algorithms, implementation and scalability. *Concurrency: Practice and Experience*, 11, 863-884., 1999.
- [37] Touheed, N, Selwood, P, Jimack, P K & Berzins, M, A Comparison of Some Dynamic Load-Balancing Algorithms for a Parallel Adaptive Flow Solver. *Parallel*

- Computing, 26, 1535–1554, 2000.
- [38] Trottenberg, U, Oosterlee, C, Schüller, A, Multigrid. Academic Press 2001
  - [39] Vidwans, A, Kallinderis, Y & Venkatakrishnan, V, Parallel Dynamic Load\_Balancing Algorithm for Three\_Dimensional Adaptive Unstructured Grids. AIAA Journal, 32, 497–505, 1994.
  - [40] Walshaw, C, Cross, M & Everett, M G, Parallel Dynamic Graph Partitioning for Adaptive Unstructured Meshes. J. Par. Dist. Comput., 47, 102–108, 1997.
  - [41] Warren, J A & Boettinger W J, Prediction of Dendritic Growth and Microsegregation Patterns in a Binary Alloy using the Phase-Field Method. Acta Matall. Mater., 43, 689–703, 1995.
  - [42] Willnecker, R, Herlach, D M & Feuerbacher, B, Evidence of Nonequilibrium Processes in Rapid Solidification of Undercooled Melts. Phys. Rev. Lett. 62, 2707–2710, 1989.
  - [43] Wheeler, A A, Murray, B T & Schaefer, R J, Computation of Dendrites Using a Phase-Field Model. Physica D, 66, 243–262, 1993.
  - [44] Williams, R D, Performance of Dynamic Load Balancing for Unstructured Mesh Calculations. Concurrency: Practice and Experience, 3, 457–481, 1991.