An Adaptive, Multilevel Scheme for the Implicit Solution of Three-Dimensional Phase-Field Equations

J.R. Green*, P.K. Jimack*, A.M. Mullis+ and J. Rosam*+

*School of Computing, University of Leeds, UK
+School of Process, Environmental & Materials Engng., University of Leeds, UK

Abstract

Phase-field models, consisting of a set of highly nonlinear coupled parabolic partial differential equations, are widely used for the simulation of a range of solidification phenomena. This paper focuses on the numerical solution of one such model, representing anisotropic solidification in three space dimensions. The main contribution of the work is to propose a solution strategy that combines hierarchical mesh adaptivity with implicit time integration and the use of a nonlinear multigrid solver at each step. This strategy is implemented in a general software framework that permits parallel computation in a natural manner. Results are presented which provide both qualitative and quantitative justifications for these choices.

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

1 Introduction

This paper describes the parallel and adaptive numerical solution of systems of nonlinear partial differential equations (PDEs) which are derived from phase-field models of solidification problems, [5, 6, 9, 11, 12, 13]. In particular, we demonstrate that it is possible to solve such systems efficiently in three space dimensions using fully-implicit time stepping. Previous research into the numerical solution of systems of this type has already shown the necessity of mesh adaptation, [8, 9, 21], however such results have typically been based upon the use of explicit time stepping, at least for the nonlinear components of the differential equations. Our earlier publications for systems in two space dimensions have clearly illustrated the advantages of using implicit time stepping once the maximum level of spatial refinement exceeds a certain threshold, [23, 24]: the key to obtaining these advantages being the efficient solution of the nonlinear algebraic equations that arise at each time step, through the use of nonlinear multigrid techniques [2, 4, 7, 28].

In this work we have taken the initial steps towards the generalization of our earlier work in two space dimensions, [23, 24], to three dimensions. It should be noted that in order to make the extension to three dimensions computationally tractable it is necessary to combine adaptivity and multigrid techniques with the use of parallel computer architectures, primarily in order to provide sufficient memory for runs of the required resolution, but also to reduce execution times. Parallel implementations of 3-d phase-field models are already described in the literature, [6, 9, 26] for example, however the combination of parallel, adaptive, implicit and multigrid has not been presented before to our knowledge.

The layout of this paper is as follows. In the next section the model equations are introduced, along with a brief description of their spatial and temporal discretization. The following section then provides an overview of the adaptive, parallel and multigrid techniques that have been applied, along with an overview of the PARAMESH software library, [19, 20], that has been used to realize an implementation of these techniques. In order to complete this work a number of extensions to PARAMESH have been developed and so, in Section 4, these are briefly described. Finally, in Sections 5 and 6 respectively, a selection of numerical results are presented and the contributions and future potential of this work are discussed.

2 Equations and Discretization

The phase-field method is one of the most powerful techniques to have emerged in recent years for the computational modelling of phase change problems. The novelty of the 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, ϕ, which represents the phase of the material (typically -1 in the liquid and $+1$ in the solid regions). The evolution of ϕ 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. One such morphology, which has been widely studied in the solidification literature, is the dendrite [1, 3, 5, 18]: a branched needle-like crystal. Dendrites are ubiquitous in nature, occuring in many cast metals and igneous minerals as well as giving rise to the multitude of patterns found in snowflakes. This morphology, which is one of the prime examples of spontaneous pattern formation, is indicative of the outward diffusion of either heat or some chemical species from the growing solid into the host medium (generally a melt, supersaturated solution or supersaturated vapour). Dendrites generally display either cubic or hexagonal symmetry, which reflects the underlying symmetry in the atomic packing in the crystal. This gives rise to a small anisotropy in the surface energy of the crystal that profoundly effects the final growth morphology.

2.1 Phase-field equations

A variety of different phase-field models have been proposed in recent years, many of which have been demonstrated to capture the formation of of dendritic structures, e.g. [11, 14, 15, 17, 21, 29] and, more recently, [6, 9, 12, 13], in three dimensions. In this paper we work exclusively with the three-dimensional model described in [13], which is appropriate for simulating the solidification of an under-cooled pure melt and depends only upon the temperature field u (in addition to the phase field ϕ). This model is formulated in the so-called thin-interface limit which means that the numerical results are independent of the width chosen for the solid-liquid interface and thus have a quantitative validity not possessed by many other formulations of the phase-field problem. In order to impose the asymmetry in the model that is required for the simulation of dendritic growth, use is made of an anisotropy term $A(\theta, \psi)$, where θ and ψ are the standard spherical angles that the outward pointing normal to the solid-liquid interface, and its projection in the $x-y$ plane, make with respect to the z-axis and the x-axis, respectively. Let $\underline{n}$ denote the outward pointing unit normal to the solid-liquid interface and let $\underline{e}_1$, $\underline{e}_2$ and $\underline{e}_3$ denote the usual Cartesian basis vectors. The connection between ϕ, $\underline{n}$ and the angles θ and ψ are

$$\begin{aligned} \nabla\phi &= |\nabla\phi|\underline{n} \\ &= \phi_x\underline{e}_1 + \phi_y\underline{e}_2 + \phi_z\underline{e}_3 \end{aligned} \tag{1}$$

and

$$\underline{n} = \sin\psi(\cos\theta\underline{e}_1 + \sin\theta\underline{e}_2) + \cos\psi\underline{e}_3 \ , \tag{2}$$

leading to

$$\tan\theta = \frac{\phi_y}{\phi_x} \quad \text{and} \quad \cos\psi = \frac{\phi_z}{|\nabla\phi|} \ \text{ or } \ \tan\psi = \frac{\sqrt{\phi_x^2+\phi_y^2}}{\phi_z} \ . \tag{3}$$

Without repeating aspects of the derivation here (see [12, 13] for details), the mathematical formulation may be stated as the following nonlinear system of coupled parabolic PDEs:

$$\begin{aligned} \tau(\theta,\psi)\frac{\partial\phi}{\partial t} &= \nabla\cdot(W(\theta,\psi)^2\nabla\phi) + \phi(1-\phi^2) - \lambda u(1-\phi^2)^2 + \\ &\quad \frac{\partial}{\partial x}\left[W(\theta,\psi)\left(-\frac{\partial W(\theta,\psi)}{\partial\theta}\frac{\phi_y|\nabla\phi|^2}{\phi_x^2+\phi_y^2} + \frac{\partial W(\theta,\psi)}{\partial\psi}\frac{\phi_z\phi_x}{\sqrt{\phi_x^2+\phi_y^2}}\right)\right] + \\ &\quad \frac{\partial}{\partial y}\left[W(\theta,\psi)\left(\frac{\partial W(\theta,\psi)}{\partial\theta}\frac{\phi_x|\nabla\phi|^2}{\phi_x^2+\phi_y^2} + \frac{\partial W(\theta,\psi)}{\partial\psi}\frac{\phi_z\phi_y}{\sqrt{\phi_x^2+\phi_y^2}}\right)\right] - \\ &\quad \frac{\partial}{\partial z}\left[W(\theta,\psi)\frac{\partial W(\theta,\psi)}{\partial\psi}\sqrt{\phi_x^2+\phi_y^2}\right] \end{aligned} \tag{4}$$

and

$$\frac{\partial u}{\partial t} = \alpha\nabla^2 u + \frac{1}{2}\frac{\partial\phi}{\partial t} \ . \tag{5}$$

The first of these is the phase equation and the second is the temperature equation. The latter is linear (since the thermal coefficient, α, is constant) however the former is highly nonlinear: λ is a prescribed constant, whilst τ and W take the form $\tau_0 A(\theta,\psi)^2$ and $W_0 A(\theta,\psi)$ respectively. For the purposes of this paper, we take the anisotropy term $A(\theta,\psi)$ to be given by

$$A(\theta,\psi) = A_0(1+\tilde{\epsilon}\left(\cos^4\psi + \sin^4\psi\left\{1-2\sin^2\theta\cos^2\theta\right\}\right)), \tag{6}$$

where τ_0, W_0 and A_0 are prescribed constants. As will be evident from the numerical results below, this has the effect of prescribing a cubic symmetry, similar to that found in most simple metals, with preferred growth along the coordinate axis (see [22] for further details). The strength of the anisotropy is controlled by $\tilde{\epsilon}$.

For practical purposes, in order to obtain the term $\nabla^2\phi$ explicitly, it is necessary to expand out the first term on the right-hand side of the phase equation prior to discretization. This is achieved as follows:

$$\begin{aligned}
\nabla\cdot\left(W(\theta,\psi)^2\nabla\phi\right) &= \frac{\partial}{\partial x}\left(W(\theta,\psi)^2\right)\frac{\partial\phi}{\partial x} + \frac{\partial}{\partial y}\left(W(\theta,\psi)^2\right)\frac{\partial\phi}{\partial y} \\
&\quad + \frac{\partial}{\partial z}\left(W(\theta,\psi)^2\right)\frac{\partial\phi}{\partial z} + W(\theta,\psi)^2\nabla^2\phi \\
&= \left(2W(\theta,\psi)\frac{\partial W(\theta,\psi)}{\partial x}\frac{\partial\phi}{\partial x}\right) + \left(2W(\theta,\psi)\frac{\partial W(\theta,\psi)}{\partial y}\frac{\partial\phi}{\partial y}\right) + \\
&\quad \left(2W(\theta,\psi)\frac{\partial W(\theta,\psi)}{\partial z}\frac{\partial\phi}{\partial z}\right) + W(\theta,\psi)^2\nabla^2\phi \\
&= 2W(\theta,\psi)\left(\frac{\partial W(\theta,\psi)}{\partial\theta}\left[\frac{\partial\theta}{\partial x}\frac{\partial\phi}{\partial x} + \frac{\partial\theta}{\partial y}\frac{\partial\phi}{\partial y} + \frac{\partial\theta}{\partial z}\frac{\partial\phi}{\partial z}\right] + \right. \\
&\quad \left.\frac{\partial W(\theta,\psi)}{\partial\psi}\left[\frac{\partial\psi}{\partial x}\frac{\partial\phi}{\partial x} + \frac{\partial\psi}{\partial y}\frac{\partial\phi}{\partial y} + \frac{\partial\psi}{\partial z}\frac{\partial\phi}{\partial z}\right]\right) \\
&\quad + W(\theta,\psi)^2\nabla^2\phi\,.
\end{aligned} \tag{7}$$

This last expression may be evaluated with the following derivative expressions:

$$\frac{\partial\theta}{\partial x} = \frac{\phi_x\phi_{xy}-\phi_y\phi_{xx}}{\phi_x^2+\phi_y^2}, \quad \frac{\partial\theta}{\partial y} = \frac{\phi_x\phi_{yy}-\phi_y\phi_{xy}}{\phi_x^2+\phi_y^2} \text{ and } \frac{\partial\theta}{\partial z} = \frac{\phi_x\phi_{zy}-\phi_y\phi_{zx}}{\phi_x^2+\phi_y^2}; \tag{8}$$

and

$$\begin{aligned}
\frac{\partial\psi}{\partial x} &= \frac{-\phi_{xz}(\phi_x^2+\phi_y^2)+\phi_x\phi_z\phi_{xx}+\phi_y\phi_z\phi_{xy}}{|\nabla\phi|^2\sqrt{\phi_x^2+\phi_y^2}} \\
\frac{\partial\psi}{\partial y} &= \frac{-\phi_{yz}(\phi_x^2+\phi_y^2)+\phi_x\phi_z\phi_{xy}+\phi_y\phi_z\phi_{yy}}{|\nabla\phi|^2\sqrt{\phi_x^2+\phi_y^2}} \\
\frac{\partial\psi}{\partial z} &= \frac{-\phi_{zz}(\phi_x^2+\phi_y^2)+\phi_x\phi_z\phi_{xz}+\phi_y\phi_z\phi_{yz}}{|\nabla\phi|^2\sqrt{\phi_x^2+\phi_y^2}}\,.
\end{aligned} \tag{9}$$

2.2 Discretization

We have successfully used both finite difference, [23], and low-order continuous finite element, [10], discretizations in combination with adaptivity and multigrid in two space dimensions. Detailed comparisons of our implementations for both second and fourth order schemes, see [22], show that the overhead of assembling the finite element systems at each step means that the computational times required, to achieve equivalent accuracy, are consistently greater than with the use of finite difference schemes. For this reason we base our initial computations in three dimensions upon the use of a second order seven point finite difference stencil for the $\nabla^2\phi$ and $\nabla^2 u$ terms. This is selected for its relative simplicity, despite the complexity of the nonlinear equations being solved.

Similarly, for the sake of simplicity, our implicit time discretization is based upon the backward Euler scheme, although we note that in two dimensions the use of a second order BDF scheme is significantly more efficient, [23].

An important feature, not described in detail here, of the equations introduced in the previous section is that the width of the phase interface (i.e. the distance indicative of the requirement for ϕ to vary from -1 to $+1$) is determined *a priori* by the choice of the constant λ that is used in the model (see [12, 13]). For an accurate simulation to be performed an interface width which is thin relative to the smallest feature to be resolved by the simulation (typically the dendrite tip) is required, and so a correspondingly fine spatial mesh is needed. It is not feasible, or necessary, to have such a fine mesh throughout the domain and so mesh adaptivity is required in order to refine the mesh around the phase interface (and to un-refine the mesh after the interface has passed through a region). Details of this are provided in the next section however such a refinement strategy clearly has an effect on the discretization scheme. In particular, the finite difference, or finite element, discretization must be consistent at the boundary between regions of different levels of mesh refinement.

Fortunately, this issue may be handled efficiently within the implementation of the solver, rather than explicitly within the discretization scheme. This is due to our use of the MLAT (Multi-Level Adaptive Technique) scheme, [4], for the multigrid solution of the discrete systems of algebraic equations arising at each time step. This scheme is designed to allow multigrid to be applied on domains where nested local refinement has taken place. Assuming that the coarsest grid covers the entire domain, an initial solution is obtained on this grid. This is then interpolated onto the locally refined grid (just two grid levels are used in this explanation for simplicity) and pre-smoothing takes place on the refined region only, using the interpolated values at the boundary of this region as Dirichlet conditions. A coarse grid correction then takes place, in which the boundary conditions on the refined region may be updated, followed by the post-smoothing steps in the refined region only. For multigrid V-cycles this process is repeated. One of the advantageous side-effects of this approach is that the discretization scheme used for the smoother is only ever applied on regions of uniform levels of refinement. Consequently, no special consideration is required to account for the local mesh refinement at the discretization stage.

3 Solver Details

3.1 Nonlinear multigrid

When a fully implicit time integrator, such as the backward Euler scheme, is used, it is necessary to solve a large nonlinear system of algebraic equations at each step. As indicated above, this is achieved using a nonlinear multigrid version of the MLAT scheme: namely the FAS (full approximation scheme), as described in [28]. For a given time step, let the discrete equations on a fine mesh, of size h say, be denoted as the nonlinear system

$$\underline{F}^h(\underline{v}^h) = \underline{N}^h[\underline{v}^h] - \underline{f}^h = 0 \ , \tag{10}$$

where $\underline{v}^h$ is the set of unknown nodal values at the new time level. The pre-smoothing stage of the FAS multigrid requires the current estimate of $\underline{v}^h$, $\underline{v}^{h,k}$ say, to be updated using nonlinear sweeps of the form

$$\begin{aligned} \overline{v}_i^{h,k+1} &= v_i^{h,k} - \frac{F_i^h(\underline{v}^{h,k})}{\frac{\partial F_i^h}{\partial v_i}(\underline{v}^{h,k})} \\ v_i^{h,k+1} &= \omega \times \overline{v}_i^{h,k+1} + (1-\omega) \times v_i^{h,k} \ . \end{aligned} \tag{11}$$

Use of the under-relaxed Jacobi iteration (in our case with a value $\omega = 0.85$) is known to yield better smoothing properties than standard Jacobi iteration (at least in the linear case, [28]) – a nonlinear Gauss-Seidel form is also possible but we only consider this weighted-Jacobi smoother in this work. The next step is to move to the coarser grid in order to correct this latest solution. The transfer between the fine and the coarser grids requires a restriction operator, I_h^{2h}, and a prolongation operator, I_{2h}^h, to be defined. If the defect on the fine grid, $\underline{d}^h$, is written as:

$$\underline{d}^h = \underline{f}^h - \underline{N}^h[\underline{v}^h], \tag{12}$$

then on the coarser grid

$$\underline{N}^{2h}[\underline{w}^{2h}] = \underline{f}^{2h} \tag{13}$$

must be solved, where $\underline{f}^{2h} = I_h^{2h}[\underline{d}^h] + \underline{N}^{2h}[\underline{v}^{2h}]$. Once a solution to this has been found the coarse grid correction, $\underline{e}^{2h}$ is recovered from $\underline{e}^{2h} = \underline{w}^{2h} - I_h^{2h}[\underline{v}^h]$ and transferred back to the fine grid in order to correct the previous fine grid solution as follows:

$$\underline{e}^h = I_{2h}^h[\underline{e}^{2h}] \tag{14}$$

$$\underline{v}^h \rightarrow \underline{v}^h + \underline{e}^h. \tag{15}$$

Having completed this coarse grid correction step the solution on the finest grid is smoothed further and a check is made for convergence. If this has not occurred the whole process is repeated.

The procedure described above applies to two grids of spacing h and $2h$ respectively. However, this procedure may be applied recursively, up to a coarsest level, in order to solve (13). This recursive form defines the multigrid scheme.

3.2 Parallel adaptivity using PARAMESH

In order to ensure that sufficient memory and processing power is available for large three-dimensional simulations it is necessary to make use of parallel computing architectures. In order to facilitate mesh adaptivity in parallel this work makes use of the open source library PARAMESH [19, 20]. This is a collection of Fortran90 functions designed to support the development of adaptive numerical solvers in one, two or three space dimensions (from now on, only the three-dimensional case will be discussed however). In particular, the functions permit the seamless use of parallel processing by taking control of all issues associated with data locality and message passing.

The key structure that PARAMESH uses to permit, and control, adaptive mesh refinement is the data block. Each block contains a number of data cells corresponding to a uniform hexahedral mesh: this mesh consists of a constant number of *real cells* in each co-ordinate direction and a constant number of *guard cells* which surround the real cells in each direction. The real cells store the data for this particular block while the guard cells hold copies of data from neighbouring blocks. The number of real and guard cells per block is decided by the programmer. However the number of guard cells must be sufficient for the size of stencil that is used by the discretization. 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, a guard cell update algorithm handles any necessary parallel communication. Dynamic load-balancing, [27, 31], is also used to ensure that the computational load is evenly distributed across the available processes. For all of the results presented in this paper data blocks with eight real cells and one guard cell in each direction have been used (in practice there are two guard cells in each direction: one at each end of the real cell region). Consequently, each block contains 1000 (10^3) cells in total, of which 512 (8^3) contain degrees of freedom whilst the remainder must be updated with copies of data owned by neighbouring cells. This significant storage overhead is the price that is paid for the flexibility of the data block approach.

The data block is also the fundamental unit that is used by the mesh refinement and coarsening functions within PARAMESH. Indeed, the locally refined mesh, at any point in time, is stored as a hierarchical oct-tree structure (possibly with more than one block at the root level), where each node of the tree is a data block and the leaf nodes of the tree (i.e. those without children) form the blocks on which the current solution is sought. Each block may be refined into eight children, and these blocks may themselves be refined, until a predefined maximum level is reached, or some other criterion is satisfied. The refinement (and coarsening) of the blocks is based upon the use of flags that are set by the programmer based upon a chosen error estimate or error indicator. Once the flags are set 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. In addition, subroutines are called to carry out load balancing and to attempt, where possible, to ensure that neighbouring blocks are on the same process. The overall

structure of this PARAMESH adaptive solution process is illustrated by the following pseudo-code:

1. Create non-uniform mesh hierarchy up to the highest level;
2. Set up initial condition $t = 0$ and fill guard cells;
3. Go to next time step
   - Advance solution to $t + dt$
   - Fill guard cells
   - Mark for refinement or coarsening
   - Refine and/or coarsen mesh
   - Interpolate the data to the new mesh and update guard cells;
4. Got to step 3.

Note that there is an important trade-off that must be considered in selecting the dimension of each data block. If the local adaptivity is to be focused only in the regions where it is needed (for example, around a phase interface) then the number of real cells in each block should be small, however this leads to two significant problems: firstly, the total number of guard cells will be much greater, thus increasing the overhead; secondly, more frequent remeshing events will be triggered as the interface evolves, providing a further computational overhead. Hence the optimal choice of block size requires a balance between the need for an efficiently refined mesh against the costs of additional guard cells and excessive remeshing.

4 Parallel Adaptive Multigrid for PARAMESH

PARAMESH already includes routines required to carry out multigrid prolongation and restriction since these are also needed following local mesh refinement. A number of additional components are required however in order to implement the nonlinear multigrid solver on the non-uniform hierarchical data structure. In the interests of brevity only a very short outline of these developments is provided here: the first concerns the implementation of the FAS algorithm, described above, and the second relates to the use of the locally refined mesh hierarchy for the MLAT scheme.

The main issue that must be addressed in implementing the FAS scheme is to manage the data that exists at each level of the mesh hierarchy. For example, when restriction operations are undertaken on the latest solution estimate and the corresponding defect, this data is copied into temporary (work) variables which are then passed to the restriction functions. This requires some modification of the PARAMESH data tree however, because the default restriction operation is designed to be applied after a mesh has been coarsened and so the labelling of the leaf nodes is normally updated.

Whilst the next set of solver operations are indeed required at this coarser level it is important to ensure that the finer blocks are available for the subsequent prolongation of the coarse grid correction. Furthermore, another temporary array must be used for this prolongation so as to ensure that the saved (pre-corrected) solution value on the fine grid is not over-written by the prolongation function. The situation is made more complicated by the fact that for each cell in the block there are two unknowns: the phase and the temperature values. Hence the restriction and prolongation functions must be called separately for each variable at each stage, requiring the labelling of the leaf nodes to be reset after each such function call.

The application of the MLAT scheme also requires some care due to the fact that PARAMESH only carries out guard cell updates upon blocks which are leaf nodes in the tree or which are the parents of leaf nodes. This is all that is required for a non-multigrid solver with local adaptivity, however during an FAS multigrid cycle the blocks which hold the current solution will change (and will not necessarily be leaf nodes of the tree hierarchy). This means that the blocks upon which the guard cell update function operates must also change. We have achieved this by temporarily re-labelling the cells at the current level of the multigrid hierarchy to ensure that the guard cell updates apply to the correct data. The motivation for accepting this additional bookkeeping overhead (and the associated code development) was to provide wrapper subroutines which sit around the existing PARAMESH functions, rather than to modify the PARAMESH routines themselves.

5 Numerical Results

5.1 Qualitative features

Figures 1 to 3 show snapshots of a typical dendritic structure that has been simulated using the solution techniques described in this paper. These results have been computed using an anisotropy parameter $\tilde{\epsilon} = 0.05$, λ = 3.2 and an initial undercooling, Δ, of 0.65 (that is u is set initially everywhere to -0.65). For this run, up to six levels of refinement of the initial $8 \times 8 \times 8$ block (level 0) have been permitted, which provides equivalent resolution to that of a uniform mesh of dimension $512 \times 512 \times 512$, containing ~ 134.2M cells. An initial spherical seed of solid phase is defined at the centre of the domain ($[-200, 200]^3$) and the evolution of the solution is computed. In this example a fixed time step is used and the spatial adaptivity is based upon the gradient of the phase variable. Results were computed on a dual (quad-core) processor workstation with 16Gb of RAM.

Figure 1 shows the $\phi = 0$ isosurface at the end of this representative computation, by which time a clear dendritic structure has formed. Figure 2 shows part of this isosurface along with cuts through the locally refined mesh at this final time. It is apparent that the maximum level of refinement (level 6 in this case) occurs in the vicinity of the solid-liquid interface and that, by this stage of the computation, the coarsest mesh is at level 3 (equivalent to a $64 \times 64 \times 64$ grid). Figure 3, which shows

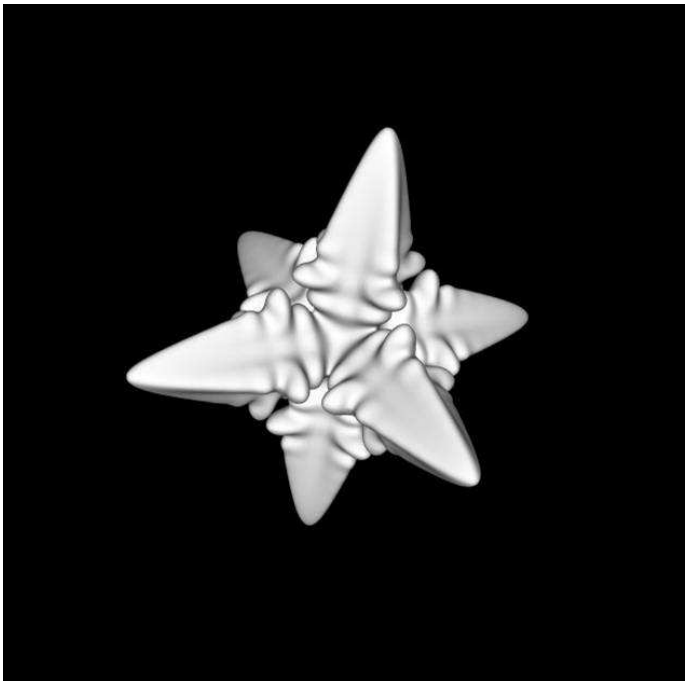

Figure 1: snapshot of a typical anisotropic solution, showing the $\phi = 0.0$ isosurface at the end of a simulation

the adapted mesh only, illustrates that mesh coarsening has begun to occur at the centre of the domain: which was originally refined to the maximum level.

5.2 Quantitative comparison with an explicit code

In this section we compare the performance of the implicit, multigrid solver described above with a much simpler explicit model employing forward Euler time-stepping. This solver, full details of which are given elsewhere [22], is also based around the PARAMESH package. Both solvers use the block-based adaptive capabilities of PARAMESH and comparative tests have been conducted on the same multi-processor workstation with up to 4 cores and 8Gb of RAM available. In each case exactly the same equation set is solved however, when using explicit methods, [9, 21], a major constraint in the computation is the time-step restriction in order to assure stability of the scheme. The unconditional stability of the implicit scheme allows much greater time steps to be taken but, even with the efficiency of multigrid, the cost per step is far greater. The purpose of these tests is to ascertain the extent to which this higher computational overhead per time step is compensated by the ability to take larger time steps. However, the comparison also provides a useful check of the solver integrity, provided both explicit and implicit models converge to the same result.

Comparative tests have been run for an initial seed growing into a uniformly undercooled melt with Δ = 0.65 on a $[-200, 200]^3$ domain. As above, the interface width has been fixed by setting λ = 3.2. Tests have been performed for both isotropic growth

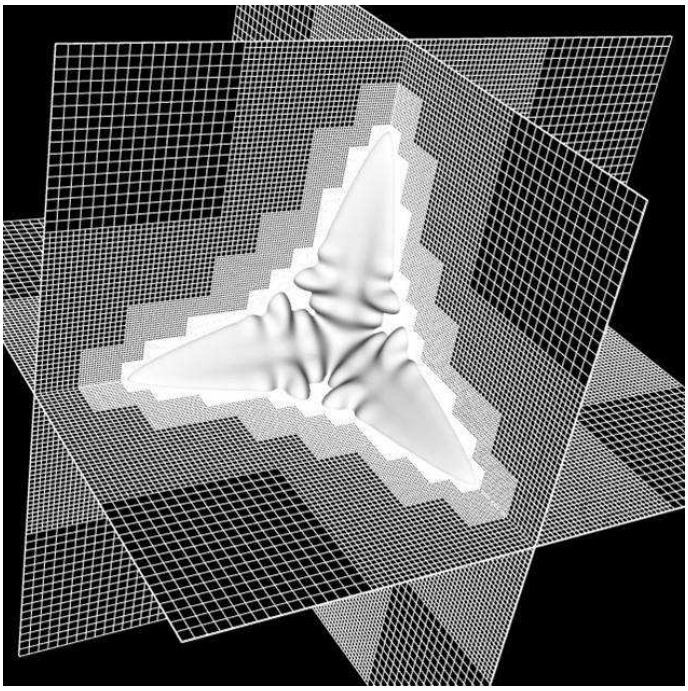

Figure 2: snapshot of the same solution as in Fig. 1, showing the adapted grid as well as the $\phi = 0.0$ isosurface

($\tilde{\epsilon} = 0$), in which case a simple spherical morphology results, and anisotropic growth ($\tilde{\epsilon} = 0.05$), giving rise to dendrites such as that shown in Figure 1.

In order to show the effect of moving to progressively finer mesh spacing, tests have been conducted for 5-7 levels of refinement, corresponding to minimum mesh sizes of $h = 1.56$ to $h = 0.39$ respectively on the $[-200, 200]^3$ domain employed here. For the explicit model, the maximum stable time-step, as determined by numerical experimentation, was 0.15, 0.0375 and 0.009375 for 5, 6 and 7 levels of refinement respectively, irrespective of whether the solution was isotropic or anisotropic. Conversely, for the implicit model the selected time step was held constant for all simulations, at 0.15, except the anisotropic case with 7 levels of refinement. In this case the nonlinear multigrid did not always converge and so the step was reduced to 0.06.

On the coarsest meshes employed some differences in the solutions obtained using explicit and implicit methods are observed (of the order 5 % in the position of the dendrite tip), which is most likely due to the structured hexahedral mesh imparting additional anisotropy into the solution. This problem is well documented in phase-field simulation and has been shown to be most severe when coarse meshes are used, [16]. Convergence to the same result for the explicit and implicit solvers is observed as the mesh spacing is reduced, with the results on level 7 being virtually indistinguishable.

Results of the comparative run time are reported based on the real time required on four cores for the model to reach a (non-dimensional) simulation time of $t = 150$ and are given in Table 1. Note that, on the coarsest mesh (level 5) the same time step, $dt = 0.15$, has been used for both the explicit and the implicit runs. Consequently,

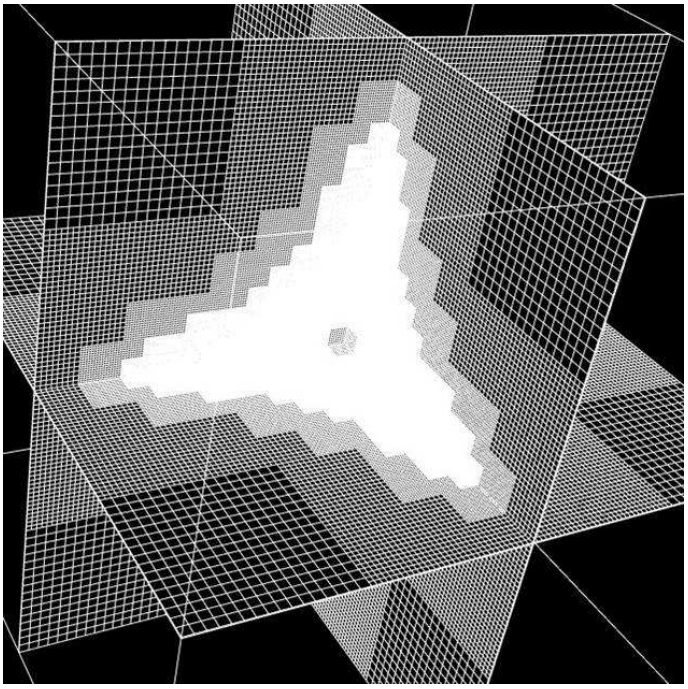

Figure 3: snapshot of the locally adapted grid corresponding to the solution shown in Fig. 1

due to the lower computational load per time step, the explicit model clearly performs much better in this case. However, as we move to progressively finer meshes the time step for the explicit model scales, as expected, as h^2, while the time-step for the implicit model remains h independent. Consequently, with 6 levels of refinement we see similar computational times, while with 7 levels of refinement the implicit scheme clearly performs much better.

		Isotropic			Anisotropic		
h	level	Explicit	Implicit	Ratio	Explicit	Implicit	Ratio
1.56	5	2052	6480	3.14	2412	13788	5.78
0.78	6	27108	35064	1.29	42912	47484	1.11
0.39	7	307476	176688	0.57	437832	537912	1.23

Table 1: Comparative execution times required for the explicit and implicit solvers to advance the phase-field simulation to $t = 150$. Times based on parallel execution on 4 cores.

Results for the anisotropic case follow a similar trend except that at the highest level of refinement the implicit time step has been reduced to $dt = 0.06$, leading to computational times that are marginally longer than for the equivalent explicit scheme. While this result is somewhat inconclusive with regard to the choice of explicit versus implicit scheme for this system, we have demonstrated elsewhere in 2-dimensions,

[22, 24], that for the much stiffer system that result when chemical, as well as thermal, diffusion is introduced into the crystallization problem, solution via explicit methods becomes infeasible once the ratio of thermal to chemical diffusivity (the Lewis number) becomes significant. Conversely, stiff implicit multigrid methods permit solution in 2-dimensions even when Lewis numbers of order 10000 are used, [25]. Consequently, we believe that implicit, multigrid methods are the only way in which such systems can be tackled in 3-dimensions.

5.3 Parallel performance

The primary purpose of implementing the three-dimensional solver in parallel using PARAMESH is to allow larger problems to be solved than would otherwise be possible (on a single processor). Before considering the degree to which this has been achieved however it is also informative to assess the efficiency of the underlying parallel implementation for problems of a fixed size. Table 2 shows the results of two such tests using uniform mesh refinement up to levels 3 and 4 respectively. For the smaller of these two problems it is possible to obtain a parallel efficiency of just over 50% on 16 cores, and for the larger problem an efficiency of just under 50% (relative to the 8 core case) is achieved using 64 cores. As the problem size is increased (by increasing the number of levels of refinement) the memory requirement grows, and so the minimum number of cores needed to run the problem also goes up. The system used for all of the parallel runs in this sub-section consists of dual core AMD processors with up to 2Gb of RAM per core.

Case One: 262144 Cells				Case Two: 2097152 cells			
Cores	Time	Speed-up	Efficiency	Cores	Time	Speed-up	Efficiency
1	5198	-	-				
2	2755	1.89	94%				
4	1445	3.60	90%				
8	770	6.75	84%	8	6611	-	-
16	616	8.44	53%	16	4298	1.54	77%
				32	2796	2.36	59%
				64	1734	3.81	48%

Table 2: Parallel scalability results for the phase-field solver on two problems of fixed sizes

Clearly the overhead associated with managing the block data structures and the oct-tree hierarchy means that the efficiency figures in Table 2 are not particularly good for cases where the problem size remains fixed as the number of processors grows. An alternative assessment of parallel scalability is presented in Table 3, which considers the impact of scaling the problem size in proportion to the number of processors. In this case the problem size is measured by the number of cells in the finest mesh (uniform mesh refinement is used once more), which grows by a factor of 8 each time

the number of cores is increased by this factor. The execution time of each V-cycle of the multigrid solver grows linearly with the problem size and so, even though the total number of V-cycles is dependent on the mesh size (primarily due to smaller time steps being required on a finer mesh), the execution time per V-cycle should remain constant in each run if the parallel efficiency is 100%. Table 3 clearly shows that this parallel efficiency, when the work per core is kept constant, is indeed very good. The figure of 136% for the 8 core case is quite difficult to explain but may be due to the single core runs being adversely affected by other users on the system.

Cores	Cell count	Time per V-cycle	Efficiency
1	262k	27.9	-
8	2097k	20.5	136%
64	16777k	30.5	91%

Table 3: Parallel scalability results for the phase-field solver as the problem size is scaled with the number of cores

Of course, the primary goal of this work is not to achieve parallel scalability on uniformly refined meshes but to allow a parallel capability for the implicit solution of phase-field problems on locally adapting meshes. Table 4 shows how well this has been achieved by comparing the simulation capability that is possible using a parallel solver with uniform mesh refinement versus the capability that adaptive mesh refinement allows. It is clear that, for this particular run (which depends upon the dendrite morphology and the domain size in the adaptive case), the same spatial resolution may be obtained on just 2 cores with adaptivity as on 64 cores without. Similarly, adaptivity permits refinement level 7 to be reached using just 20 cores however the memory requirement for uniform refinement to level 7 (saving the mesh hierarchy for the multigrid solver) would require approximately 4096 cores. Although limits on the cpu time available to us have prevented a full simulation from being completed (and so the results are not shown in Table 4) we have also been able to show that adaptive simulations up to refinement level 8 (equivalent to over 8 billion cells on a uniform grid) are possible using 128 cores.

Cores	Uniform Grid Level (Cells)	Adaptive Grid Level (Equiv. Cells)
1	3 (262k)	4 (2.1M)
2		5 (16.8M)
8	4 (2.1M)	6 (134.2M)
20		7 (1073.7M)
64	5 (16.8M)	

Table 4: Comparison of the capability of the parallel adaptive solver (right) versus a uniform solver (left)

6 Discussion

This paper presents what the authors believe to be the first example of the successful application of parallelism, adaptivity, implicit time-stepping and multigrid together, for the solution of three-dimensional phase-field systems. The work is a natural extension of our earlier results in two space dimensions, [22, 23], where the advantages of implicit time-stepping are shown, provided that the mesh is sufficiently fine and a fast (i.e. multigrid) algebraic solver is implemented. The implementation of the multigrid solver in three dimensions presents no additional mathematical difficulties however the technical problems associated with moving from two to three dimensions are significant. These have been overcome through the use, and further development, of the PARAMESH software library [19, 20].

Results presented show that the development of complex structures, such as dendrites, can be successfully simulated in three dimensions, and that the adaptive data structure permits an efficient representation of the solution fields. Comparison with an existing explicit solver, [22], demonstrates both the accuracy and the potential efficiency of the implicit approach. Finally, the parallel performance is shown to be adequate to allow a significantly improved capability over the use of either parallelism or adaptivity alone.

There are a number of important extensions of this work that are still to be developed. Relatively minor examples, already available in our 2-d solver [23], include the use of a second order time-stepping scheme, such as BDF2, the development of adaptive time-stepping based upon a local error estimate, and the use of symmetry boundary conditions to permit simulations in just one eighth of the current computational domain. More substantial developments that are planned centre primarily around the generalization to a much wider class of phase-field models. In particular, our main motivation for this work is to be able to solve not only thermal problems, but those involving the diffusion of a chemical species (alloy solidification). This problem is somewhat more complex as the transport equation now also becomes non-linear due to the complex form of the source and anti-trapping terms. In the first instance this can be solved in the isothermal approximation (the temperature field is assumed constant), which is appropriate to very slow solidification: although ultimately it is desirable to be able to solve coupled thermal-solutal models which arise in the non-isothermal solidification of binary alloys [24]. These PDE systems have the additional complexity of requiring highly disparate length and time scales to be resolved, leading to extremely stiff differential systems. Based upon our observations in two dimensions, [24, 25], the advantages of a fully implicit solver are likely to be highly significant in these cases however there is additional complexity due to an increased number of nonlinear PDEs.

Acknowledgement

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

References

[1] Barbieri, A & Langer, J S, Predictions of Dendritic Growth Rates in the Linearized Solvability Theory. Phys. Rev. A, 39, 5314–5325, 1989.

[2] Bastian, P, Lang, S & Eckstein, K, Parallel Adaptive Multigrid Methods in Plane Linear Elasticity Problems. Numer. Linear Alebr., 4, 153–176, 1997.

[3] 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.

[4] Brandt, A, Multi-Level Adaptive Solutions to Boundary-Value Problems. Mathematics of Computations, 31(138), 333-390, 1977.

[5] Brener, E, Needle-Crystal Solution in 3-Dimensional Dendritic Growth. Phys. Rev. Lett., 71, 3653–3656, 1993.

[6] George, W L & Warren, J A, A Parallel 3D Dendritic Growth Simulator Using the Phase-Field Method. J. Comput. Phys., 177, 264–283, 2002.

[7] Griebel M & Zumbusch, G, Parallel Multigrid in an Adaptive PDE Solver Based on Hashing and Space-Filling Curves. Parallel Comput., 25, 827–843, 1999.

[8] Hu, X L, Li, R & Tang, T, A Multi-Mesh Adaptive Finite Element Approximation to Phase Field Models. Comm. Comput. Phys., 5, 1012–1029, 2009.

[9] 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.

[10] Jones, A C & Jimack, P K, An Adaptive Multigrid Tool for Elliptic and Parabolic Systems. Int. J. Numer. Meth. Fluids, 47, 1123–1128, 2005.

[11] 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.

[12] Karma, A & Rappel, E-J, Phase-Field Simulation of Three-Dimensional Dendrites: Is Microscopic Solvability Theory Correct? J. Crystal Growth, 174, 54–64, 1997.

[13] Karma, A & Rappel, E-J, Quanitative Phase-Field Modeling of Dendritic Growth in Two and Three Dimensions. Phys. Rev. E, 57, 4323–4349, 1998.

[14] 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.

[15] 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

[16] Mullis, A M, Quantification of mesh induced anisotropy effects in the phase-field method. Comp. Mater. Sci., 66, 345–353, 2006

[17] 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.
[18] 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
[19] Olson, K, PARAMESH: A Parallel Adaptive Grid Tool. In Parallel Computational Fluid Dynamics 2005: Theory and Applications, ed. A. Deane et al. (Elsevier), 2006.
[20] Olson, K, PARAMESH tutorial. www.physics.drexel.edu/˜olson/paramesh-doc/Users_manual/amr_tutorial.html, accessed 10 December 2008.
[21] 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.
[22] Rosam, J, A Fully Implicit, Fully Adaptive Multigrid Method for Multiscale Phase-Field Modelling. PhD Thesis, University of Leeds, 2007.
[23] 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
[24] 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.
[25] Rosam, J, Jimack, P K & Mullis, A M, Quantitative phase-field modelling of thermo-solutal solidification at high Lewis number. Phys. Rev. E, 79, art. no. 030601, 2009
[26] Suwa, Y, Saito, Y & Onodera, H, Parallel computer simulation of three-dimensional grain growth using the multi-phase-field model . Mater. Trans., 49, 704–709, 2008.
[27] 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.
[28] Trottenberg, U, Oosterlee, C, Schüller, A, Multigrid. Academic Press 2001
[29] Warren, J A & Boettinger W J, Prediction of Dentritic Growth and Microsegregation Patterns in a Binary Alloy using the Phase-Field Method. Acta Matall. Mater., 43, 689–703, 1995.
[30] Wheeler, A A, Murray, B T & Schaefer, R J, Computation of Dendrites Using a Phase-Field Model. Physica D, 66, 243–262, 1993.
[31] Williams, R D, Performance of Dynamic Load Balancing for Unstructured Mesh Calculations. Concurrency: Practice and Experience, 3, 457–481, 1991.