An adaptive mesh method for phase-field simulation of alloy solidification in three dimensions

P C Bollada [1], P K Jimack[1] and A M Mullis[2]
University of Leeds: [1] School of Computing, [2] Institute of Materials Research
E-mail: p.c.bollada@leeds.ac.uk

Abstract. We present our computational method for binary alloy solidification which takes advantage of high performance computing where up to 1024 cores are employed. Much of the simulation at a sufficiently fine resolution is possible on a modern 12 core PC and the 1024 core simulation is only necessary for very mature dendrite and convergence testing where high resolution puts extreme demands on memory. In outline, the method uses implicit time stepping in conjunction with an iterative solver, adaptive meshing and a scheme for dividing the work load across processors. We include three dimensional results for a Lewis number of 100 and a snapshot for a mature dendrite for a Lewis number of 40 .

1. Introduction

This paper summarises our mathematical and numerical approach to solving the set of multi-scale, coupled, non-linear PDEs presented by alloy solidification using a phase field. In order to capitalise on routinely available parallel computing both on desktop PCs and high performance computing we adopt strategies that differ from that in which serial code is the end product. The major challenge is to take advantage of both adaptive grid and a multigrid solver in a parallel environment. A key strategy here is to use 'blocks'. Blocks are subsets of the computational domain containing a uniform grid and the adaptive grid strategy operates on the block level. Efficient communication between blocks is enhanced the larger the block, but efficient adaptive meshing is more effective with smaller blocks. We describe our approach to parallel implementation which rests on an ordering scheme akin to Morton ordering. In this approach communication at any level of grid refinement is minimised, but at a cost of communication across levels. In order to reduce communication between blocks we use a single 'ghost' layer of nodes in tandem with compact $3 \times 3 \times 3$ finite difference stencils for computing the PDE operators. Once again there is a balance to be struck since a double ghost layer is more expensive on communication but allows higher order accuracy in stencil operators. The multigrid solver we employ is the Full Approximation Scheme (FAS) which utilises Jacobi smoothing on each layer which, in turn, employs a pointwise Newton solver to approximate the non-linear algebraic terms. The Newton solver, being an extension of the Newton-Raphson method to systems of equations, becomes a generalisation of the Jacobi iterative method to non-linear systems. To find the pointwise Newton correction simply requires knowledge of the contribution of all the variables at any particular computational node. This is relatively easy to implement once the PDEs have been discretised. We employ (and adapt) the open source code PARAMESH to control the parallel communication between blocks so that once the FAS

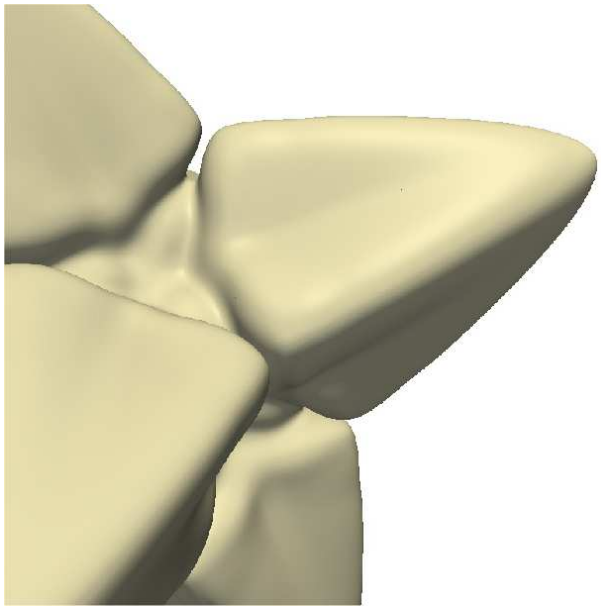

Figure 1. A close up of one of the arms of mature dendrite. An under cooling of $\Delta = 0.525$ at $Le = 40$ was used for this simulation. This took about three days on a 12 core desktop PC and represents a real growth time of about 0.6μs and size 0.25μm from an inital seed of about 6nm.

has been implemented the code is readily adapted to a variety of applications - not necessarily phase field. More details of this approach plus validation may be found in [1].

The particular phase field model we employ is an extension of [2], and is based on the three dimensional thermal- phase field model of [3] and two dimensional thermal-solutal phase field model of [4]. Fig. 1 shows a close up of one arm of a mature dendrite from a simulation in which the undercooling, $\Delta = 0.525$ and Lewis of 40. Such a dendrite does not differ in appearance to a simulation without coupling to a evolving temperature field, but a fully coupled model (with temperature field) becomes increasingly demanding on computing resources because of the stiffness of the resulting equations and extent of the temperature field. It is this fact that has generated the computing scheme detailed here.

Finally, we would like to draw the attention of researchers using an anisotropic phase field model in three dimensions to the equation given in Eq 19 which is significantly more efficient to compute in this form than its expanded equivalent of partial differential operators, resulting from, say, a **Maple** evaluation. It also aids the non-linear solver implementation detailed in Sec. 3.4.

2. Governing equations

The non-dimensional equations for the phase field, ϕ, the solute concentration, c and the dimensionless temperature, θ, are given via a specification of the free energy

$$F \equiv \int_V A(\mathbf{n})^2 \nabla\phi \cdot \nabla\phi + f(\theta, \phi) V \tag{1}$$

and the relations

$$\tau(c,\phi) A^2(\mathbf{n}) \dot{\phi} = -\frac{\delta F}{\delta \phi} \tag{2}$$

$$\dot{c} = \nabla \cdot \left(K(\phi) \nabla \frac{\delta F}{\delta c} - \mathbf{j} \right), \tag{3}$$

$$\dot{\theta} \quad = \quad D_\theta \nabla^2 \theta + \dot{\phi}. \tag{4}$$

The solute diffusion parameter is given by

$$K = D_c \frac{(1-\phi)}{2}. \tag{5}$$

The parameter D_c is a diffusion constant, thus $K = 0$ in the solid ($\phi = 1$) and $K = D_c$ in the liquid ($\phi = -1$). D_θ is the temperature diffusion coefficient (assumed constant). The normal to the interface is given by

$$\mathbf{n} = \frac{\nabla \phi}{|\nabla \phi|}, \tag{6}$$

which is well defined around $\phi = 0$, and the anisotropy function for cubic symmetry (growth is preferred along the normals to the faces) is given for three dimensions by [3],

$$A(\mathbf{n}) \quad \equiv \quad A_0 \left[1 + \tilde{\epsilon} \left(n_x^4 + n_y^4 + n_z^4\right)\right] \tag{7}$$

where $\mathbf{n} = [n_x, n_y, n_z]$, $A_0 = 1 - 3\epsilon$, $\tilde{\epsilon} = 4\epsilon/(1-3\epsilon)$ and $\epsilon \approx 0.02$ governs the amount of anisotropy. The dimensionless relaxation time function is defined by

$$\tau(c, \phi) \equiv \frac{1}{Le} + M c_\infty [1 + (1 - k_E)U], \tag{8}$$

where the Lewis number $Le = D_\theta / D_c$ and

$$U \equiv \frac{1}{1-k_E} \left(\frac{2c/c_\infty}{1 + k_E - (1-k_E)\phi} - 1 \right). \tag{9}$$

Here k_E is the equilibrium partition coefficient, c_∞ is the far boundary condition for c. The anti-trapping current $\mathbf{j}$, appearing in the solute equation, Eq. 3, is prescribed by

$$\mathbf{j} = -\frac{1}{2\sqrt{2}} [1 + (1 - k_E)] U \dot{\phi} \mathbf{n}, \tag{10}$$

The profile of c exhibits a spike at the interface, which can present computational difficulties. Following [4], this is largely overcome by rewriting the solute equation using the variable U:

$$\left(\frac{1 + k_E}{2} - \frac{1 - k_E}{2} \phi \right) \frac{\partial U}{\partial t} = \nabla \cdot \left\{ D_c \frac{1-\phi}{2} \nabla U + \mathbf{j} \right\} + \frac{1}{2} [1 + (1 - k_E)U] \frac{\partial \phi}{\partial t}. \tag{11}$$

The physical temperature field, T, can be recovered by the relationship

$$\theta = \frac{T - T_M - m c_\infty}{L / C_p}, \tag{12}$$

where L and C_p are the latent heat of the phase transition and heat capacity respectively. The slope of the liquidus line is given by $m = ML/[C_p(1 - \kappa_E)]$ and T_M is the melting temperature of the alloy.

Finally the bulk free energy density is given by

$$f(\theta, \phi) \equiv \frac{\phi^2}{2} \left(\frac{\phi^2}{2} - 1 \right) + \lambda (\theta + c_\infty U) \left(\phi - \frac{2\phi^3}{3} + \frac{\phi^5}{5} \right). \tag{13}$$

We solve the system of equations 2,4 and 11 plus initial (typically small) solid seed and far boundary conditions

$$\begin{aligned} \phi|_{far} &= -1 \\ U|_{far} &= 0 \quad (\equiv c|_{far} = c_\infty) \\ \theta|_{far} &= -\Delta \end{aligned} \tag{14}$$

where Δ is the given under-cooling. The equation for temperature is a standard diffusion equation with a heating term, $\dot{\phi}$, proportional to the solidification rate. The phase equations, though compact in variational form are complicated in PDE form. However, it is possible to write the resulting PDEs in a reasonably compact form convenient for computation.

$$\tau(c,\phi)A(\mathbf{n})^2\dot{\phi} = \frac{\partial^2\phi}{\partial x^i \partial x^j} g_{ij} - \frac{\partial f}{\partial \phi} \tag{15}$$

where $\tau(c,\phi)$ is given by Eq. 8, $\dfrac{\partial f}{\partial \phi}$ is given by

$$\frac{\partial f}{\partial \phi} = \phi^3 - \phi + \lambda(\theta + c_\infty U)(1 - 2\phi^2 + \phi^4), \tag{16}$$

and g_{ij} defined by

$$\begin{aligned} g_{ii} &\equiv (24X_i - 3)A^2 + (-48X_i^2\tilde{\epsilon} + 12X_i\tilde{\epsilon} - 40X_i + 4)A_0A + 16X_i(X_i\tilde{\epsilon} + 1)^2A_0^2 \\ g_{ij} &\equiv \frac{\phi_{,i}\phi_{,j}}{g}\left[24A^2 + (-24X_i\tilde{\epsilon} - 24X_j\tilde{\epsilon} - 40)A_0A + (16(X_j\tilde{\epsilon} + 1))(X_i\tilde{\epsilon} + 1)A_0^2\right], \quad i \neq j. \end{aligned} \tag{17}$$

with $X_i \equiv \phi_{,i}^2/|\nabla\phi|^2$. However, for the purposes of discretisation we adopt the following form by defining

$$M_{ij} \equiv \frac{\partial\phi}{\partial x^i \partial x^j} - \frac{1}{3}\nabla^2\phi\delta_{ij} \tag{18}$$

so that the phase field equations Eq. 15 becomes

$$\tau(c,\phi)A(\mathbf{n})^2\dot{\phi} = \frac{1}{3}\delta_{ij}g_{ij}\nabla^2\phi + M_{ij}g_{ij} - \frac{\partial f}{\partial\phi} \tag{19}$$

The fully coupled system is then given by Eq 19, Eq. 11 and Eq. 4.

2.1. Parameter values

For the purposes of this paper we choose a selection of parameters to use as default values for the simulations below in Table 1.

3. Discretisation

The approach taken to discretisation is based upon a cell centred finite difference scheme, in that the nodes of the domain are located at the centre of cubic cells. and thus, we use the term 'node' and 'cell centre' inter changeably. One consequence of this is that there are no nodes on the domain boundary, thus making the use of Dirichlet boundary conditions non-trivial. The scheme makes use of the PARAMESH library to support mesh adaptivity in parallel [6, 7]. The meshes obtained by this approach take the form of an oct tree of regular blocks, within which the mesh is uniform, and it is the spatial discretisation on any one of these blocks that we discuss here.

Table 1. Table of parameter values used for the simulations in this paper.

Physical property	Symbol	value
Anisotropy	ϵ	0.02
Boundary concentration	Mc_∞	0.05
Equilibrium partition coefficient	κ_E	0.3
Dimensionless interface width	λ	2
Ratio of solute diffusivity to characteristic diffusivity	D_c	1.2534
Lewis number - D_θ/D_c	Le	40 and 100
Dimensionless Undercooling at the far boundary	Δ	0.25 to 0.80
Initial nuclear radius	R_0	5.0
Computational property	symbol	value
Finest grid size	Δx	0.195 to 0.78
Computation domain size	L^3	800×800×800

3.1. Temporal discretisation
For temporal discretisation we employ backward differential formula using two previous values (BDF2).

3.2. Spatial discretisation
Compact finite difference stencils (3×3×3), are used to discretise the first and second derivatives. Denoting these 27 points by $\mathbf{Q}$ and defining a generic 27 point Laplacian stencil, W_{abc}, around a point $\mathbf{p} = [i, j, k]$ by

$$\nabla^2 u|_{\mathbf{Q}} = \frac{1}{(\Delta x)^2} \sum_{a=-1}^{1} \sum_{b=-1}^{1} \sum_{c=-1}^{1} W_{abc} u_{\mathbf{p}+[a,b,c]} \tag{20}$$

where Δx is the physical distance between nearest neighbours, we can recover the 7 point Laplacian stencil, built from only the centre node, $\mathbf{p}$ and the 6 nearest neighbours ($a^2+b^2+c^2 = 1$) by setting the weights

$$W_{abc} = \begin{cases} -6 & a^2+b^2+c^2 = 0 \\ 1 & a^2+b^2+c^2 = 1 \\ 0 & otherwise \end{cases} . \tag{21}$$

However, this stencil is more prone to grid anisotropy than the following 27 point Laplacian stencil (see [5]), with weights

$$W_{abc} = \begin{cases} -128/30 & a^2+b^2+c^2 = 0 \\ 14/30 & a^2+b^2+c^2 = 1 \\ 3/30 & a^2+b^2+c^2 = 2 \\ 1/30 & a^2+b^2+c^2 = 3 \end{cases} \tag{22}$$

The discretisation of Eq 19 is achieved using the Laplacian stencil above and stencils for g_{ij} and M_{ij}. The latter have discretisations that do not have any contribution from the central node. This is, firstly, because g_{ij} is a rational polynomial function of first derivatives of ϕ and, secondly, we deliberately adopt the 7 point stencil for the Laplacian in the definition of M_{ij} for this purpose. For example, M_{11} and M_{12} at grid point, $\mathbf{p} = [i, j, k]$ are discretised

$$M_{11}|_{\mathbf{p}} \equiv \frac{2}{3\Delta x^2}(\phi_{i+1,j,k} + \phi_{i-1,j,k}) + \frac{1}{\Delta x^2}(\phi_{i,j+1,k} + \phi_{i,j-1,k} + \phi_{i,j,k+1} + \phi_{i,j,k-1}) \tag{23}$$

$$M_{12}|_{\mathbf{p}} \equiv \frac{\partial^2 \phi}{\partial x^1 \partial x^2}|_{\mathbf{p}} = \frac{1}{4\Delta x^2}(\phi_{i+1,j+1,k} + \phi_{i-1,j-1,k} - \phi_{i+1,j-1,k} - \phi_{i-1,j+1,k}) \tag{24}$$

3.3. Adaptive mesh and block tree structure

The domain is first divided into a number of mesh blocks each of which contains $N \times N \times N$ hexahedral cells, where we typically choose $N = 8$. We employ a domain of 800^3, which is large enough for Lewis numbers of the order of 100. We divide this domain into 4^3 blocks, so that when N=8, each cell is of size 25^3 and refer to this as level 1. The adaptive mesh strategy then imposes a hierarchical sub-division of some or all of these blocks, and their descendants, based upon an oct tree structure. This subdivision aims to concentrate cells where gradients of the oct tree variables are highest and to ensure neighbour blocks differ by at most one level. The finest grid we work with is at level 7, with a corresponding $\Delta x = 25/2^7 = 0.1953125$. We find that the minimum finest level necessary to obtain qualitatively reasonable results is level 5, corresponding to $\Delta x = 0.78125$.

In order to discuss further the tree structure of the blocks we denote any block by its label, i and its contents/properties, $B(i)$:

$$B(i) = [l, s, p, \mathbf{c}, \mathbf{x}] \tag{25}$$

where $l \in [1, n]$ is the level, $s \in [1, 8]$ is the sibling number (i.e. an index for which child of p the block is), p is the parent index, $c_i, i \in [1, 8]$ are the 8 child indices, and $\mathbf{x} = [x, y, z]$ is the Cartesian position coordinates of the block origin. Any one of these properties can be accessed by the block number, i. Some examples: $p(i)$ is the block number of the parent of block i; $\mathbf{x}(p(i))$ is the position of the parent's origin; $c_{s(i)}(p(i)) = i$ is an identity.

A complete specification of all the blocks in the oct tree is then specified by the list:

$$\mathbf{B} = \{B(i), i \in [1, B_N]\} \tag{26}$$

where B_N is the total block number. Moreover, a childless block, i, can be indicated by specifying, $\mathbf{c}(i) = 0$ and so the oct tree also can be specified by a listing of just the leaf blocks

$$\mathbf{B} = \{B(i), i \in [1, B_N] : \mathbf{c}(i) = \mathbf{0}\}. \tag{27}$$

Each block has uniform meshing and the adaptive strategy is further restricted by only allowing blocks at level n adjacent to blocks of $n-1, n$ and $n+1$. Thus, even though a block may be flagged for coarsening, this (latter) restriction often prevents this happening. Blocks are flagged for refinement if, for any point, $\mathbf{p}$, in the block, the following criterion is satisfied:

$$e \equiv \max\{e_\phi|\phi_{\mathbf{p}} - \phi_{\mathbf{p-q}}|, e_U|U_{\mathbf{p}} - U_{\mathbf{p-q}}|, e_T|T_{\mathbf{p}} - T_{\mathbf{p-q}}|\} > \eta, \tag{28}$$

where we use, for tolerance, $\eta \sim 1$ and

$$|\phi_{\mathbf{p}} - \phi_{\mathbf{p-q}}| \equiv \sqrt{(\phi_{\mathbf{p}} - \phi_{\mathbf{p}-[1,0,0]})^2 + (\phi_{\mathbf{p}} - \phi_{\mathbf{p}-[0,1,0]})^2 + (\phi_{\mathbf{p}} - \phi_{\mathbf{p}-[0,0,1]})^2}, \tag{29}$$

etc. Typically the weights, e_ϕ, e_U and e_T are chosen to sum to unity. If $e < 0.1\eta$, the block is flagged for derefinement.

For communication between blocks the PARAMESH implementation allocates a block of dimension $(N+2) \times (N+2) \times (N+2)$. The first and last cells in each dimension are ghost cells - an update function may be called at any time in order to populate these guard cells with the corresponding values from the interior of each of the neigbouring blocks. The application of a discrete stencil on any block requires access to neighborouring blocks via the guard cell nodes. When the neighbouring block is coarser the guard cell of the coarse block is found by a weighted average of the 8 surrounding coarse nodes. Using a tri-linear function of x, y and z, gives the weightings, in order of nearest neighbours first

$$\mathbf{w} = \left[\frac{27}{64}, \frac{9}{64}, \frac{9}{64}, \frac{9}{64}, \frac{3}{64}, \frac{3}{64}, \frac{3}{64}, \frac{1}{64}\right]. \tag{30}$$

The process is known as prolongation. The reverse process, of finding a guard cell value for a coarse block when one or more neighbours is refined is known as restriction and is the simple average of the eight nearest, one level finer, cell centres. Both operations, restriction and prolongation using Eq. 30, are also required for multigrid as detailed in the next section.

3.4. Nonlinear smoother

The non-linear system of algebraic equations can be written using the generic vector notation $\mathbf{v}_\mathbf{p}^{n+1} \equiv [\phi_\mathbf{p}^{n+1}, \theta_\mathbf{p}^{n+1}, U_\mathbf{p}^{n+1}]$ by

$$\mathbf{A}_\mathbf{p}(\mathbf{v}_\mathbf{Q}^{n+1}) = \mathbf{0} \tag{31}$$

for each node, $\mathbf{p}$, in the grid and $\mathbf{Q}$ indicates coupling between points $\mathbf{p}$ and neighbouring nodes. Using an iteration method, with $\mathbf{v}_\mathbf{p}^{n+1}$ approximated by $\mathbf{v}_\mathbf{p}^{n+1,m}$, we define the defect

$$\mathbf{d}_\mathbf{p}^{n+1,m} = -\mathbf{A}_\mathbf{p}(\mathbf{v}_\mathbf{Q}^{n+1,m}). \tag{32}$$

The pointwise Newton update for this iteration is given by

$$\mathbf{v}_\mathbf{p}^{n+1,m+1} = \mathbf{v}_\mathbf{p}^{n+1,m} - \omega\tilde{\mathbf{d}}_\mathbf{p}^{n+1,m}, \tag{33}$$

where $\tilde{\mathbf{d}}_\mathbf{p}^{n+1,m}$ is found by solving the 3×3 system

$$\mathbf{J}_\mathbf{p}^{n+1,m}\tilde{\mathbf{d}}_\mathbf{p}^{n+1,m} \equiv \mathbf{d}_\mathbf{p}^{n+1,m} \tag{34}$$

with the 3×3 Jacobian matrix defined by

$$\mathbf{J}_\mathbf{p}^{n+1,m} \equiv \frac{\partial \mathbf{d}_\mathbf{p}^{n+1,m}}{\partial \mathbf{v}_\mathbf{p}^{n+1,m}}. \tag{35}$$

In practice we typically select, $\omega \approx 0.9$ and find that off diagonal terms of $\mathbf{J}_\mathbf{p}^{n+1,m}$ are not essential to obtain a convergent iteration.

3.5. Multigrid and parallel implementation

We supplement the Jacobi smoother with the Full Approximation Scheme (FAS) as developed in [8]. Our parallel implementation requires communication between blocks both on the same level and between levels. We identify communication across any one level as being as being more costly and so we implement a depth first traversal of the blocks and then, using this numbering, divide the work load at each level in turn between all processors. For a uniform mesh this strategy results in a near optimum allocation to cores. For adaptive meshes the communication on any level is also optimum, but communication between levels is compromised. See [1] for details.

4. Results

We demonstrate the code with sets of parameters in both 2D and 3D and with two Lewis numbers. The higher Lewis number, more undercooling and higher dimension combine to give a severe challenge to a desktop computer. Ultimately, for higher Lewis numbers, HPC resources are necessary with this approach. This is because the problem becomes stiff and the time step size required for stability becomes smaller and the domain required by the temperature field also becomes larger with larger Lewis number. Fig. 2 shows the evolution of tip radius for 2D $Le = 100$ and, for both $Le = 40$ and $Le = 100$, in 3D. Here, the capillary length,

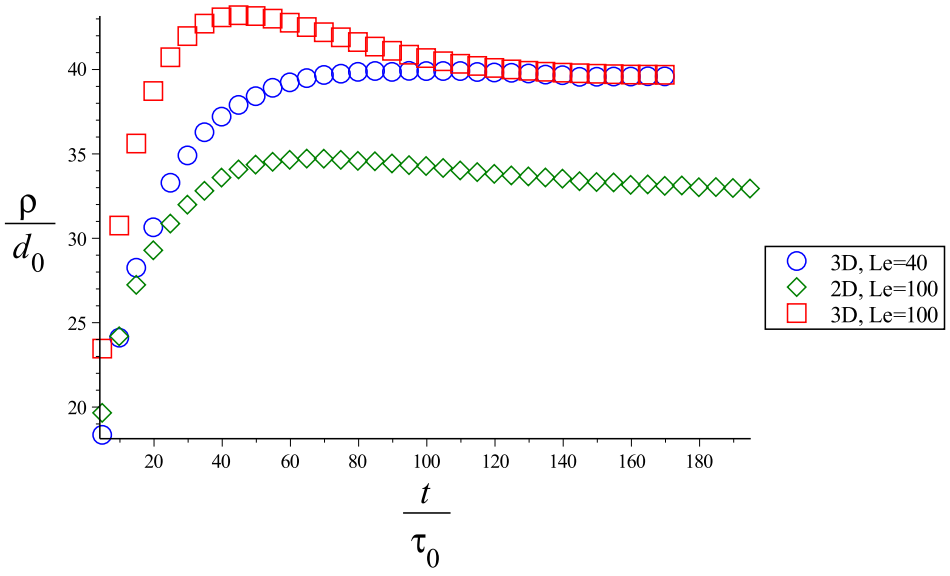

Figure 2. The tip radius for an undercooled dendrite, $\Delta = 0.325$ and $Le = 100$ and $Le = 40$

$d_0 \equiv \Gamma T_M C_p / L^2$, where Γ is the Gibbs Thompson coefficient; T_M the melting temperature; C_p heat capacity; and L the latent heat of fusion. The characteristic time, $\tau_0 = D_c W_0^2 / \tilde{D}$ where $\tilde{D} = 10^{-9}\mathrm{m}^2/\mathrm{s}$, $D_c = 1.25$ and interface width $W_0 = \lambda d_0 / a_1 = 2.26 d_0$. In our simulation we set both W_0 and τ_0 to unity. A larger domain is needed for the the 2D simulations due to the extent of the temperature profile in 2D for a given Lewis number. So despite the large increase in degrees of freedom in 3D there is some compensation due to this feature. These simulations were run on 12 core desktop PCs and typically take 3 or 4 weeks to achieve steady state. Note that the $Le = 40$ achieves steady tip radius after about $t/\tau_0 = 100$ but at $Le = 100$ steady state is achieved after $t/\tau_0 = 180$. This is generally a feature of the higher lewis number and is an additional significant reason for the difficulty of computing higher Lewis numbers.

5. Conclusion

We have presented a numerical method that permits computation in three dimensions of fully coupled alloy solidification of dendritic growth at the meso scale. The model is restricted to dilute alloys in order to take advantage of a model formulation that compensates for a larger than realistic liquid-solid interface – anti solute trapping. Despite this we opt to use a near realistic value for the interface in order to get good resolution of the tip radius. This is reflected in our parameter, $\lambda = 2$, which corresponds to approximately twice a realistic interface width. The computational methods we employ are summarised as follows:

- finite difference compact stencils
- adaptive mesh
- implicit time stepping
- non-linear Newton-multigrid solver
- parallel code and division of labour across cores

The main difficulty of the above method, partly overcome, is that combining adaptive meshing and multigrid makes it very difficult to allocate an even loading to the parallel cores.

A realistic Lewis number for typical alloy solidification is $Le \sim 10,000$, which corresponds to the ratio in length scales between heat and solute diffusion. We here simulate the model at comparatively modest Lewis numbers, $Le = 40$ and 100, to demonstrate the method. These being accessible on parallel desktop machines within a few weeks of computation time. Early results suggest that not only run-times are longer for higher Lewis numbers but physical (model) time is longer for steady state behaviour to be established.

References

[1] P.C. Bollada, C.E. Goodyer, P.K. Jimack, A.M. Mullis and F.W. Yang 2015 Three dimensional thermal-solute phase field simulation of binary alloy solidification *Journal of Computational Physics*, vol 287, pp 130 – 150

[2] A. M. Mullis, C. E. Goodyer, and P. K. Jimack 2012 Towards a Three-Dimensional Phase-Field Model of Dendritic Solidification with Physically Realistic Interface Width. *Transactions of the Indian Institute of Metals*, vol 65(6): pp 617–621

[3] Alain Karma and Wouter-Jan Rappel 1996/1997. Phase-field simulation of three-dimensional dendrites: is microscopic solvability theory correct? American Crystal Growth 1996 and Vapor Growth and Epitaxy. *Journal of Crystal Growth*, vol 174(14): pp 54 – 64

[4] J. C. Ramirez, C. Beckermann, A. Karma, and H-J J. Diepers May 2004 . Phase-field modeling of binary alloy solidification with coupled heat and solute diffusion. *Physical review. E, Statistical, nonlinear, and soft matter physics*, vol 69(5 Pt 1)

[5] W. F. Spot and G. F. Carey 1996 A high-order compact formulation for the 3d poisson equation. *Numer. Methods Partial Differential Eq.*, vol 12: pp 235 243

[6] Peter MacNeice, Kevin M. Olson, Clark Mobarry, Rosalinda de Fainchtein, and Charles Packer 2000. Paramesh: A parallel adaptive mesh refinement community toolkit. *Computer Physics Communications*, vol 126(3): pp 330 – 354

[7] K. Olson 2006. Paramesh: A parallel adaptive grid tool. in parallel computational fluid dynamics 2005: Theory and applications. *ed A. Deane et al. (Elsevier)*

[8] A. Brandt 1977. Multi-level adaptive solutions to boundary-value problems. *Math. Comp.*, vol 31: pp 333 – 390