INTERNATIONAL JOURNAL OF
NUMERICAL ANALYSIS AND MODELING
Volume 16, Number 2, Pages 297–318

DYNAMIC LOAD BALANCING FOR THE PARALLEL, ADAPTIVE, MULTIGRID SOLUTION OF IMPLICIT PHASE-FIELD SIMULATIONS

MENG-HUO CHEN, PETER C. BOLLADA, AND PETER K. JIMACK

Abstract. In this paper we assess the performance of a selection of load balancing strategies for a parallel, adaptive multigrid solver that has been developed for the implicit solution of phase-field problems. The strategies considered include a number of standard approaches and a new technique that we propose specifically for multigrid solvers. This technique takes account of the sequential nature of the grid correction used in multiplicative multilevel algorithms such as multigrid. The paper focuses on two phase-field example problems which model the rapid solidification of an undercooled binary alloy: using isothermal and non-isothermal models respectively. We undertake a systematic comparison of the different load-balancing strategies for a selection of different adaptive mesh scenarios. We conclude that the optimal choice of load-balancing strategy depends critically on the computation to communication ratio of the parallel multigrid solver, and that in the computation-dominated limit our proposed technique is typically the most effective of those considered.

Key words. Load-balancing, parallel computation, multigrid.

1. Introduction

In recent years parallel processing has been used extensively in solving partial differential equations (PDEs) arising from engineering applications. In the numerical solution of PDEs a standard approach is to introduce a grid and apply finite difference [1, 2] or finite element methods [3] to discretize the governing equations to generate a system of algebraic equations. The solution of this system of algebraic equations gives the value of the unknowns at each degree of freedom. On very fine meshes the number of unknowns may easily reach several hundreds of millions ([4, 5, 6]) and solving such system in a reasonable time is a significant challenge. To reduce this computational time parallel processing is employed. In parallel computing the unknowns are typically divided into groups (generally corresponding to the number of processors) and are assigned to processors. Then these processors solve the equations associated with the unknowns in their assigned group simultaneously, with interprocessor communication to ensure global solution.

The partition of the unknowns among processors invokes two main issues. Firstly, balancing the workload is important to parallel programs for performance reasons. In some cases the number of unknowns and/or their distribution may change over time, due to adaptive mesh refinement (AMR) for example [6, 7, 8, 9, 10]. In such situations the dynamic redistribution of the workload is necessary to maintain an equal load balance [6, 10, 11, 12]. The other main issue of importance is the communication between processors: ideally this should be minimized since it creates additional overhead. It is not uncommon that there is trade off between good load-balancing and minimized communication volume between processors. This trade off becomes more difficult with AMR as the mesh is no longer uniform [11, 12].

Received by the editors October 31, 2017 and, accepted February 5, 2018.
2000 *Mathematics Subject Classification.* 65N06, 65M06.

Moreover, the use of geometric muiltigrid creates another layer of difficulty due to the multiplicative nature of the algorithm, which requires work to be undertaken on each grid in a sequential manner. Consequently a load balancing strategy that does not take into account work on each grid or communication between grids at each level is necessarily not optimum [13, 14].

In this paper we introduce an adaptive parallel multigrid solver for a parabolic system in two or three spatial dimensions. We then consider different dynamic load balancing strategies and assess their impact on the performance of the solver for two different phase-field models arising from solidification problems. Phase field models of solidification are particularly demanding because they both track a moving boundary with a phase field, where a very fine mesh is required locally, and also evolve large scale diffusion fields such as temperature (where a much coarser mesh maybe sufficient). The solver that we use is described in detail in [6, 10]. It is based upon a cell centred finite difference scheme and fully implicit time stepping. We employ nonlinear FAS multigrid [15] as the solution scheme for the resulting nonlinear algebraic systems at each time step. The mesh generation and adaptivity are carried out by PARAMESH [14], a software library which builds a hierarchy of subgrids to cover the computational domain, with spatial resolution varying to satisfy the demands of the application. Our primary objective in this manuscript is to assess a selection of different load-balancing strategies applied to this family of phase-field problems. The approaches considered are Morton ordering, which is the default load-balancing approach used in PARAMESH [14], our own adaptation of Morton ordering for multigrid (described in [6]), as well as two standard approaches, which are recursive coordinate bisection (RCB) and graph partitioning, from the software package Zoltan [16, 17].

In the following section a brief overview of the software tools (our implicit nonlinear multigrid solver, PARAMESH and Zoltan) will be given. Section 3 then discusses the dynamic load-balancing problem encountered in this work and briefly describes each of the load-balancing strategies considered in this study. The description of the time-dependent PDE problems that we solve and the background to the numerical experiments, are presented in Section 4. Finally we present our conclusions by reporting and discussing the results from a number of numerical experiments to compare these strategies when applied to a range of phase-field approximations with different characteristics.

2. Adaptive multigrid methods

The primary software tool used in this paper is called 'Campfire'. This package was developed by Goodyear et al ([6, 9, 18]) and contains several features. These features include spatial adaptivity, implicit and adaptive time stepping, dynamic load-balancing and a nonlinear geometric multigrid solver. The spatial adaptivity uses an external software library, PARAMESH [14].

PARAMESH generates meshes as the union of blocks of cells with different physical cell sizes, which are related to each other in a hierarchical fashion using a tree data structure. The blocks at the root of the tree have the physically largest cells, while their children have smaller cells and are said to be refined. Each child block is half as large as its parent block in each spatial dimension. The children of a block are nested so that they fit within their parent block and cannot overlap one another. In each block there are $N \times N \times N$ (N is a positive integer) cells arranged in a logically Cartesian fashion. Furthermore, blocks are wrapped by (typically) a single layer of guard cells which makes the block size $(N+2) \times (N+2) \times (N+2)$. Each block is specified by a process number and the block number in that process. Any two neighbouring blocks share a common border and function values associated with the cells immediately across the border may be stored in guard cells.

For computing the discrete solution of the differential equations on the mesh in parallel, blocks are distributed across MPI processes. In the course of a computation blocks exchange values on their outer cells with their neighbouring blocks. For multigrid algorithms, parent-children communication is also required for restriction and interpolation. Interprocess communications are performed between blocks that are located on different processes. Intuitively, a smaller block size allows more flexibility to manage an equal workload on each MPI process, especially for a very non-uniformly refined mesh. However, the choice of a smaller block size makes the number of guard cells overwhelming and creates a significant overhead. A range of block sizes will be considered within this paper as part of our assessment of the different dynamic load balancing strategies.

Campfire is designed as a fully implicit time integrator and uses a nonlinear multigrid solver, the Full Approximation Scheme (FAS)[15], which is implemented with adaptive mesh refinement via the Multi-level Adaptive Technique (MLAT) [15]. Let the system of discrete nonlinear algebraic equations which need to be solved at each implicit time step take the form $\mathcal{F}(u) = 0$, where u stores all the grid values to be determined. Here, $\mathcal{F}$ represents a large nonlinear vector function, however it is convenient to also express this as $\mathcal{F}(u) = \mathcal{A}(u) - f$ for some known right-hand side vector f (which is generally zero at the finest mesh level). The main feature of FAS is that we approximate the solution itself at each multigrid level (as opposed to approximating the error on coarser levels in traditional linear multigrid). This is achieved via a nonlinear smoother followed by a coarse grid correction. Algorithm 1 shows a recursive function that applies a V-cycle of FAS on a sequence of uniformly coarser grids Ω^h, Ω^{2h}, Ω^{4h}, etc. The nonlinear operators, right-hand side terms and the grid solution corresponding to discretization on the grid Ω^h are denoted by A^h, f^h and u^h, respectively (superscripts represent the grid space). The smoother (function SMOOTH in step 1) is typically based on a local Newton step at each cell of the mesh in turn (repeated p_1 times, where p_1 in step 1 is the number of pre-smooths), whilst the coarse grid correction solves the same nonlinear problem on the coarse grid but with a modified right-hand side (steps 5 to 9). The modified right-hand side is obtained by adding the difference between the residual restricted from the finer grid Ω^h (r^{2h} in step 3) and the residual of the restricted solution (w^{2h} in step 4). This is used to generate an approximation to the error which is interpolated back to the fine grid to create the coarse grid correction (steps 10 to 12). By further sweeps of the smoother (function SMOOTH in step 13) we obtain the improved solution u^h. Recursive application of this procedure (with an approximate coarse grid solve on the very coarsest grid, step 7) leads to a multigrid implementation.

The basic idea for MLAT is to work with a sequence of grids which are subject to local nested refinement. As an example, Fig. 1 shows a two dimensional adaptive mesh with four levels (note that only the blocks are represented in this figure, not the $N \times N$ meshes within each block). Starting with the block at the coarsest level (level 1) which covers the whole domain, there are 4 blocks at each level 2, 3 and 4 which are child blocks of their parent block at the coarser level (block 3 has child blocks 6, 7, 8, 9 and block 7 has child blocks 10, 11, 12, 13). The spatial variables at

Algorithm 1 V-cycle nonlinear FAS multigrid method

```
    The superscripts $h$ and $2h$ denote fine and coarse grid ($\Omega^h$ and $\Omega^{2h}$) values
    respectively.
    **Function :** $u^h$ = V-cycleFAS($h$, $u^h$, $f^h$)
 1: Apply $p_1$ iterations of the pre-smoother on $A^h(u^h) = f^h$
    $u^h = \text{SMOOTH}\left(h, u^h, f^h, p_1\right)$
 2: Compute the residual $r^h$ on $\Omega^h$
    $r^h = f^h - A^h(u^h)$
 3: Restrict the residual $r^h$ from $\Omega^h$ to $\Omega^{2h}$ to obtain $r^{2h}$
    $r^{2h} = I_h^{2h} r^h$
 4: Restrict the find grid approximation solution $u^h$ from $\Omega^h$ to $\Omega^{2h}$ to obtain $w^{2h}$
    $w^{2h} = I_h^{2h} u^h$
 5: Compute the modified RHS
    $f^{2h} = r^{2h} + A^{2h}(w^{2h})$
 6: **if** $n < 0$ **then**
 7:     Perform an "exact" coarsest grid solve on $A^{2h}(u^{2h}) = f^{2h}$
 8: **else**
 9:     $u^{2h}$ = V-cycleFAS($2h$, $w^{2h}$, $f^{2h}$ )
10: Compute the error approximation $e^{2h}$
    $e^{2h} = u^{2h} - w^{2h}$
11: Interpolate the error approximation $e^{2h}$ from $\Omega^{2h}$ to $\Omega^h$ to obtain $e^h$
    $e^h = I_{2h}^h e^{2h}$
12: Perform correction
    $u^h = u^h + e^h$
13: Apply $p_2$ iterations of the post-smoother on $A^h(u^h) = f^h$
    $u^h = \text{SMOOTH}\left(h, u^h, f^h, p_2\right)$
```

levels 3 and 4 represent finer grids covering subdomains near the corner, provided with boundary values from their parent's neighbouring blocks. In order to apply the FAS algorithm in this case, where the finer grid covers a smaller subdomain than its parent, the modification of the right-hand side of the coarse grid correction (step 5 in algorithm 1) is only applied in the part of the coarse grid that is covered by the fine grid. Similarly, the updated error from the coarse grid correction (steps 11 and 12) is interpolated only onto the refined region. Consequently the main steps that need to be implemented as part of FAS/MLAT are: (i) the nonlinear smoother on a block; (ii) transfer from coarse to fine and vice versa; (iii) a nonlinear solve on the coarsest level. Significantly, the combination of these components requires each mesh level to be considered in strict order. For example, the multigrid V-cycle illustrated in Figure 2 shows that computations on grid 3 take place after those on grid 4; and those on grid 2 after those on grid 3; etc.

The adaptivity and dynamic load-balancing steps in Campfire use subroutines from PARAMESH. After the end of each time step we decide on whether the mesh requires adapting (local refinement and/or coarsening). If so then the adaptivity

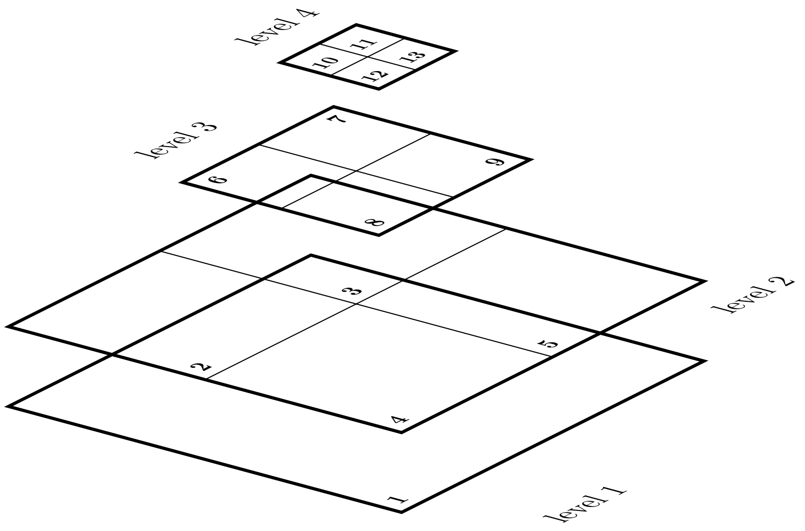


FIGURE 1. Adaptive grid with four grid levels.

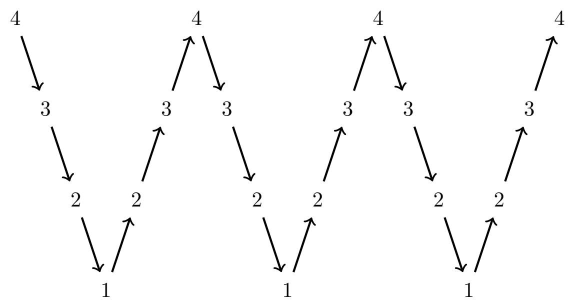


FIGURE 2. Schematic showing the application of 3 V-cycles using 4 grids.

takes place in parallel, however the resulting mesh will typically be poorly load balanced. Consequently, after each mesh adaption a load-balancing algorithm is used to assign each block to a (possibly) new process. The list of blocks is partitioned into approximately equal parts, based on the specific load balancing algorithm, and then PARAMESH performs the data (blocks) migration according to this information. The default load-balancing algorithm available in PARAMESH uses Morton ordering to number the blocks into an array which is partitioned linearly [13]. However, this approach may not achieve an ideal load-balancing in a multigrid setting due to the fact that the refinement levels of blocks are not considered in this simple partitioning approach. In an attempt to reach a good load-balance on *each* multigrid level, a modification of this approach was implemented in Campfire [6, 9], described in section 3.2 below.

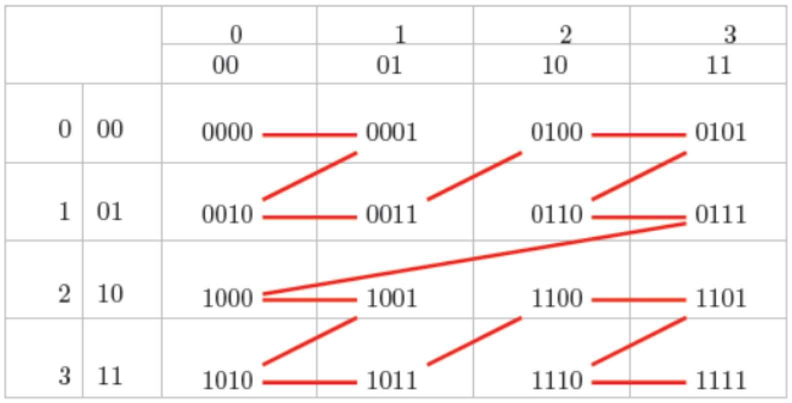
For comparison purposes, we also consider two widely used load-balancing strategies from the library Zoltan, developed by Devine et al [16]. This is a toolkit of combinatorial algorithms for parallel, unstructured, and/or adaptive scientific applications. Zoltan's largest component is a suite of dynamic load-balancing and partitioning algorithms that aim to increase an application's parallel performance by reducing processor idle time. Among the load-balancing algorithms from Zoltan, recursive coordinate bisection (RCB) and graph partitioning (GP) are chosen and employed in this work. For completeness, all the load-balancing strategies mentioned above will be described briefly in the next section.

3. Dynamic Load-Balancing

To achieve dynamic load-balancing following parallel mesh adaptivity, the partitioning and migration of data is necessary. In Campfire the smallest unit for data partitioning and migration across processors is the block. As mentioned in the previous section, a block contains $(N+2)^3$ cells. The choices of N are typically 8 ,16 or 32. It is clear that the larger the block size, a smaller total number of blocks are required for a specific mesh resolution. In the refinement steps the spatial adaptivity generates different number of blocks on each process as the computation proceeds, therefore the redistribution of blocks between processors is needed following mesh adaptivity.

The goal of load balancing and partitioning is to divide data and work among processes in a way that minimizes the overall application's execution time. This goal is most often achieved when work is distributed evenly to processes (eliminating process idle time), while at the same time minimizing the volume of interprocessor communications. Such a problem is known to be NP hard however, so heuristic approaches have typically been employed over many years [12, 19, 20, 21]. For the application considered here we have even more complexity since it is also preferable to minimize the cost of data migration from the existing partition to the new one, and, when a multigrid solver is used, it will also be advantageous to avoid partitions which have a poor load balance on the finest meshes or which have poor parent-child data locality. Note that the choice of load balancing strategy only affects the time taken for each multigrid V-cycle; and so the computed solution and the number of cycles required to converge at each time-step are independent of the load-balancing strategy.

3.1. Morton Order (MO). The dynamic load-balancing implementation in the PARAMESH library uses Morton order to number the blocks. The Morton order, also called Z-order, is a mapping of multidimensional data to one dimension that preserves locality of the data ([13]). The first step of mapping is to find the binary coordinate values of blocks. For an object in an n-dimensional space, this creates an n-tuple of binary coordinates for each block. Interleaving these binary coordinates creates a binary Z-value for each block. For example, the binary coordinate $(001001, 100101)$ gives the Z-number 010010010011. More explicitly, Figure 3 shows the binary Z-numbers of a set of 16 cells in a regular 4×4 grid covering a 2D domain. In the figure, the top row represents the ordering and binary coordinates along the x direction, while the leftmost column exhibits this information along y direction. As seen from the figure, connecting the z-values in their numerical order produces the recursively Z-shaped curve that motivates the alternative name of the ordering. Figures 4 - 5 illustrate this algorithm for a set of locally refined blocks. The former shows the block structure, while the latter illustrates this in the form of a quad-tree, where each block has been ordered. The partition is achieved by dividing the ordered list into approximately equal parts: four in this example.



		0	1	2	3
		00	01	10	11
0	00	0000	0001	0100	0101
1	01	0010	0011	0110	0111
2	10	1000	1001	1100	1101
3	11	1010	1011	1110	1111

FIGURE 3. Z values of a regular 4 × 4 grid with the origin at the upper left corner.

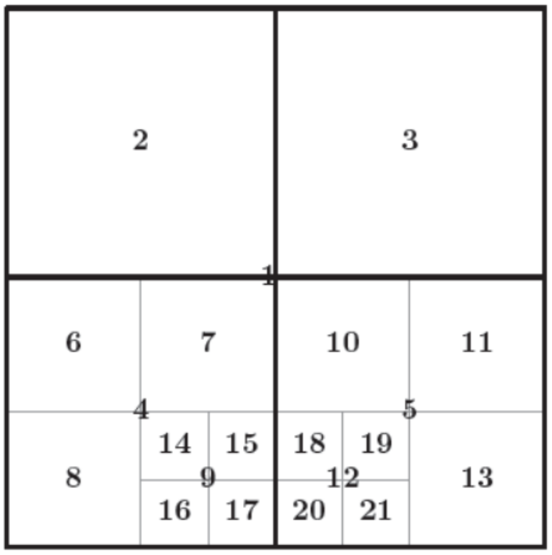


FIGURE 4. Sketch showing a 2-D Cartesian grid covered by blocks of various levels, where blocks are enumerated from number 1 to 21.

As we shall see later, the Morton order maintains good data locality and the volume of interprocessor communication is not large. However the partitioning step does not consider load-balancing on each refinement level so it is likely that the blocks of the same refinement level are not evenly distributed across all of the processors. This may cause significant load imbalance for multigrid solves, which require each mesh level to be visited in turn. Therefore a variant algorithm, that is targeted at multigrid iterations is described in the next subsection.

3.2. Level Morton Order (LMO). In an attempt to achieve load-balancing in multigrid iterations, Bollada et al [6] suggested a modified Morton partition so that

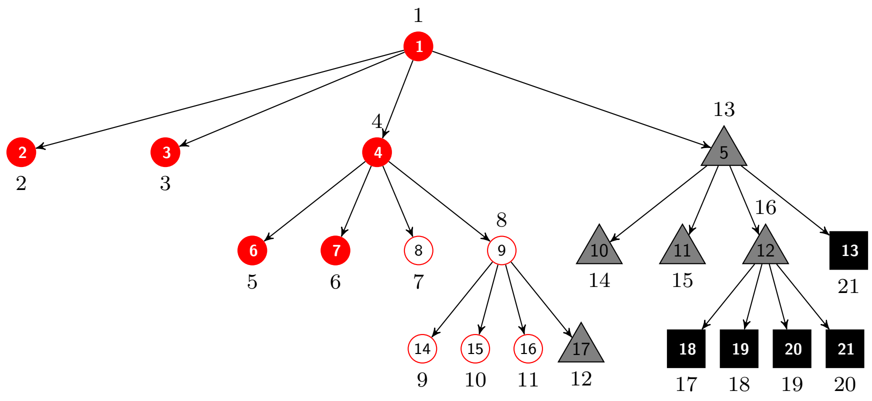


FIGURE 5. Partition of blocks in Fig. 4 using Morton order. The Morton order of a node is represented by the number next to it.

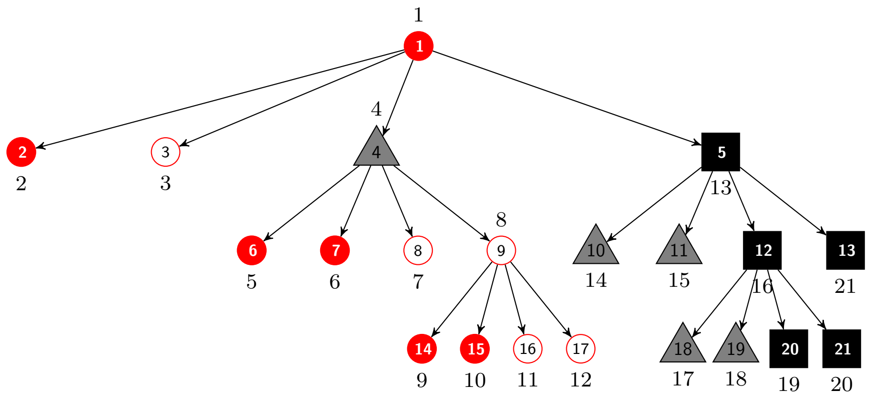


FIGURE 6. Partition of blocks in Fig. 4 using Level Morton order. The partition is different from that in Fig. 5. Note that the number next to a node represents its Morton order.

load-balancing at each refinement level is considered. That is, after the list of blocks with Morton order is generated, a loop which iterates over the refinement levels of the meshes is implemented. In each iteration of the loop, the block numbers in the ordered list belonging to the level are collected to form a sub-list whose members are still Morton ordered. Then the sub-list is evenly partitioned among processors. This strategy gives a near optimal allocation to cores at all multigrid levels. The communication on any level is also close to optimal but communication *between* levels is compromised. As seen from Figure 6, the partition of the blocks shown in Figure 4 is different to the Morton ordered partition of Figure 5. The Level Morton partitioning has an excellent distribution of blocks at each level, however it creates more interprocessor parent-child communications (9 edges for Morton order versus 13 edges for Level Morton order).

3.3. Recursive Coordinate Bisection(RCB). The recursive coordinate bisection (RCB) algorithm [12] partitions a graph according to the coordinates of the

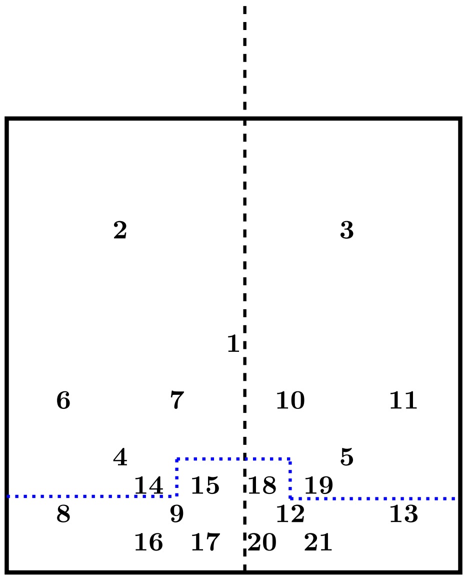


FIGURE 7. Two consecutive coordinate bisections on the group of 2D blocks shown in Fig. 4

vertices in physical space. The algorithm divides a group of geometric objects into two by a cut orthogonal to one of the coordinate axes so that half the work load is in each of the sub-groups. The splitting direction is determined by computing in which coordinate direction the set of objects is most elongated, based upon the geometric locations of the objects. The splitting of each subgroup proceeds by recursive application of the same division algorithm until the number of subgroups equals the number of processes (which needs to be power of 2 in this original form of the algorithm). Figure 7 demonstrates the RCB algorithm on the same group of 2D blocks illustrated in Figure 4, where each number represents a single block, whose center is at the location of the number. The group of blocks are divided first by the dashed cut, then each subgroup is further partitioned into two smaller subgroups by a dotted line. Note that the right-hand half is partitioned into two 5-point parts, while the left-hand half is divided into two parts with 6 points and 5 points respectively. In [17] a generalization of this approach has been implemented that allows meshes to be partitioned into an arbitrary number of subdomains (not just powers of 2).

3.4. Graph partitioning(GP). In graph partitioning the data objects (in our case blocks of mesh) are represented as graph vertices, and pairwise data dependencies as graph edges. The graph partitioning problem is then to partition the vertices into equal-weighted sub-graphs, while minimizing the weight of edges with endpoints in different subsets (known as the "cut weight"). Zoltan includes interfaces to two external graph partitioning libraries: PT-Scotch [19] and ParMETIS [20]. However in this work we use the native parallel hypergraph partitioner based on the scheme by Borzdag et al. ([21]) to perform the graph partitioning.

In all of the graph partitioning experiments that we undertake the nodes of the hypergraph are given an equal weight, to reflect the fact that all blocks contain an equal number of cells. The weights given to the edges need not necessarily be identical however. In fact we examine two variants: the first gives equal weight to all graph edges, whilst the second gives ten times the weight to parent-child edges than to graph edges between neighbouring blocks at the same level. The latter approach seeks to encourage parent-child co-location in the final partition.

4. Test Problems

In this section we describe the phase-field models, related to alloy solidification, that we use in order to assess the different dynamic load-balancing algorithms introduced in the previous section. These models takes the form of coupled nonlinear PDEs, each of which is used to describe physical problems associated with metallic alloy solidification: isothermal alloy solidification [9] and thermal-solute alloy solidification [6].

In phase-field models, the sharp interface between phases is replaced by a diffuse interface with very small thickness. This diffusion layer is described by the phase-field parameter ϕ which is a function of space $\mathbf{x}$ and time t. This phase-field parameter $\phi(\mathbf{x}, t)$, varies from 0 (or -1) to 1 between the liquid phase and the solid phase, respectively. The intermediate values between 0 (or -1) and 1 capture the phase transition. It is the position and geometry of this phase transition region that we seek to resolve to high accuracy through the use of grid adaptation as the interface evolves. Convergence at each implicit time step is achieved through the FAS multigrid algorithm described previously.

In each of the examples that follow the spatial discretization is based upon a cell centred finite difference scheme for spatial derivatives, and BDF2 (e.g. [22]) is used in time. The grid transfer operators are based upon linear interpolation and restriction with full weighting, and the coarse grid solve is approximated by repeated application of the smoother on the coarsest grid.

4.1. 3D isothermal binary alloy solidification. In this example we use the approach described in detail in [9] to simulate the isothermal growth of a dendrite from an under-cooled binary alloy. The dimensionless governing equations include the evolution of the phase field denoted by ϕ and dimensionless concentration field of the alloy, denoted by U:

$$(1) \qquad A^2\dot{\phi} = \nabla \cdot \frac{\partial}{\partial \nabla\phi}\left(\tfrac{1}{2}A^2\nabla\phi\cdot\nabla\phi\right) + \phi(1-\phi^2) + \lambda(1-\phi^2)^2(\Delta + Mc_\infty U),$$

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

In equation (1) A is an anisotropy function given by

$$(3) \qquad A = 1 - 3\epsilon + 4\epsilon\left(\frac{\phi_x^4 + \phi_y^4 + \phi_z^4}{|\nabla\phi|^4}\right)$$

where ϵ governs the strength of the anisotropy and $\phi_x = \frac{\partial\phi}{\partial x}$, etc. The parameters Δ and Mc_∞ in equation (1) are the under cooling and the scaled magnitude of the liquidus slope, respectively. They are related by

$$(4) \qquad Mc_\infty = 1 + (1-k_E)\Delta,$$

where k_E is known as the equilibrium partition coefficient. In equation (2) D is the dimensionless diffusivity. The non-dimensional concentration field U is related to the concentration c via

$$U = \frac{\left(\dfrac{2c/c_\infty}{1 + k_E - (1 - k_E)\phi}\right)}{1 - k_E}, \tag{5}$$

where c_∞ is the solute concentration far from the interface.

Following discretization, the resulting algebraic equations are nonlinear and can be written as a nonlinear vector function $A(\mathbf{v})$, where $\mathbf{v}$ represents the vector of the unknown cell-centred values at the end of current time step. For the FAS smoother the values of variables are updated via pointwise Newton iterations in which a 2×2 linear system with the local Jacobian matrix is solved. In this implementation the off diagonal terms of the Jacobian matrix are found to be insignificant, thus the Newton update only requires inversion of a 2×2 diagonal matrix for each cell on the current mesh level. This is very cheap to apply.

4.2. 3D non-isothermal binary alloy solidification. In this example we use the phase-field model, discretization scheme and nonlinear solver described in detail in [6]. In this thermal-solute model dendrite growth is driven by the alloy concentration and temperature fields, which are governed by diffusion parameters D and κ, respectively. The ratio of these two parameters yields the Lewis number, denoted by $Le = \kappa/D$.

The dimensionless governing equations describe the evolution of the phase field denoted by ϕ, concentration of the alloy denoted by c and the temperature field denoted by T. We start with the equations involve ϕ and c, as shown in Eq. (6) - (7):

$$\dot{\phi} = -M(c)\frac{\delta F}{\delta \phi}, \tag{6}$$

$$\dot{c} = \nabla \cdot D(\phi, c)\nabla \frac{\delta F}{\delta c}, \tag{7}$$

where the free energy functional $F(\phi, c, T)$, over a volume V of free energy density f, is given by

$$F = \int_V f dV. \tag{8}$$

The mobility M in (6) interpolates the mobility of each pure metal (M_0 and M_1):

$$M = (1 - c)M_0 + cM_1. \tag{9}$$

The diffusivity D in (7) is given by

$$D = \frac{[\phi D_{\text{Liq}} + (1 - \phi)D_{\text{Sol}}]c(1 - c)}{RT}, \tag{10}$$

with the constants $D_{\text{Sol}} \ll D_{\text{Liq}}$.

The evolution of the temperature field is given by

$$C_p\dot{T} = \nabla \cdot (\kappa\nabla T + Df_c\nabla f_c) - \left(1 - T\frac{\partial}{\partial T}\right) f_\phi\, \dot{\phi} - \left(1 - T\frac{\partial}{\partial T}\right) f_c\, \dot{c}, \tag{11}$$

where

$$C_p = -T\frac{\partial^2 f}{\partial T^2} \tag{12}$$

and we use the notation

$$f_\phi \equiv \frac{\partial f}{\partial \phi},\ f_c \equiv \frac{\partial f}{\partial c}. \tag{13}$$

This temperature equation follows the theory proposed by [23].

The free energy density f splits into two parts: the surface and bulk

$$f = f_S + f_B, \tag{14}$$

where the surface part controls the interface width and the anisotropy and is given by

$$f_S = W\left(\frac{1}{2}A|\nabla\phi|^2 + \phi^2(1-\phi)^2\right) \tag{15}$$

with anisotropy again defined by

$$A = 1 - 3\epsilon + 4\epsilon\left(\frac{\phi_x^4 + \phi_y^4 + \phi_z^4}{|\nabla\phi|^4}\right). \tag{16}$$

In (15), the energy of the double well barrier height is given in terms of the barrier heights associated with each alloy component by interpolation:

$$W = (1-c)W_0 + cW_1. \tag{17}$$

The bulk free energy is formed from given database functions and interpolation:

$$f_B = g(\phi)f_{\mathrm{Liq}}(c,T) + (1-g(\phi))f_{\mathrm{Sol}}(c,T), \tag{18}$$

where the particular interpolation function chosen, $g(\phi)$ is

$$g(\phi) = \phi^2(3-2\phi). \tag{19}$$

The values for the free energies of the solid and liquid phases, $f_{\mathrm{Liq}}(c,T)$ and $f_{\mathrm{Sol}}(c,T)$, may be found in CALPHAD [24] or from SGTE v5.0 [25]. The values for the undefined parameters are all constant.

As for the previous test problem the spatial variables are discretized via cell centred finite differences and the time variable is discretized using BDF2 [22]. In this case however a much more expensive smoother is required: at each smoothing step a 24×24 block smoother is used, corresponding to the part of the Jacobian associated with the 3 unknowns on a cube containing 8 cells including the current cell.

5. Numerical Experiments and Comparisons

In this section we compare the performance of the different dynamic load balancing strategies for managing the block distribution arising from mesh adaptivity described in section 2. All the tests are carried out on the same system ("ARC3" at the University of Leeds) which is equipped with 252 standard nodes, each with 24 cores and 128GB memory. The compute nodes are connected with Infiniband FDR of 56Gbit/s in a 2:1 blocking fat-tree topology, built up from non-blocking islands of 24 nodes. Our experiments use multiples of 24 to investigate the scaling of the performance of the solver in order to minimize communications between nodes, which is more costly than communication within a node. Latency and bandwidth of internode communications depend upon whether the nodes are on the same island, which is not generally the case.

For comparing the solver's performance using different load-balancing strategies, a set of restart files were generated. Then the simulations are all initiated from these restart files and run for a fixed number of time steps. In the 3D simulations the running times for completing 10, 20 or 40 (depending on cases) time steps are recorded.

Besides the running times for completing the simulations, the corresponding maximum number of blocks among processors at each multigrid level are recorded. This information indicates the degree of load-balancing at each multigrid step. For example, if a block assignment strategy results in large maximum number of blocks in the finer multigrid levels, it is to be expected that the waiting time for the other processors will be significant before moving onto the next multigrid level. Communications between processors can also be a bottleneck so, to consider this issue, we represent the volume of inter processor communications by counting the maximum (among processes) number of edges between parent and children blocks not on the same processor (x), plus the edges between neighbouring blocks at the same refinement level which are on different processes (y).

5.1. 3D Phase isothermal field binary alloy solidification. For this test problem three block sizes are considered in simulations ($8 \times 8 \times 8$, $16 \times 16 \times 16$, $32 \times 32 \times 32$). The parameters used for simulations are listed in Table 1 and these lead to the creation of a dendrite shape which is developed as a start point for all tests (see Figure 8). In order to make a fair comparison between runs the restart files are checked so that the tip locations for these three cases are nearly the same in each cases (i.e., for each block size). Furthermore, the minimum mesh size is fixed at 0.78125. Thus the number of refinement levels, and hence the multigrid levels, are different in the three cases. Also note that the time step sizes and running time for each time step are different in each case, hence the number of time steps used for comparing the performance in each case is different.

TABLE 1. Parameters for simulations: 3D Phase field isothermal binary alloy solidification.

Parameters for 3D Phase field isothermal binary alloy solidification				
Domain	Δ	k_E	ϵ	Mc_∞
800^3	-0.525	0.3	0.02	0.6325

In the first case, the first block size considered is $8 \times 8 \times 8$. The maximum refinement level is 8. The tip location at the start time step is 287.41 and the initial time is 205.60. The performance tests run for 40 time steps. At the end of each test run, the tip of the dendrite has moved to location 287.95. The starting total number of blocks is 132089 and this increases to 132737 blocks at the end of the run, where there were 9 grid adaptations carried out. The initial partitioning of blocks in the restart file is obtained from Morton ordering however the partitioning of blocks is carried out again by the new partitioning strategies before the first time step is executed.

Table 2 presents the execution time of the solver using the different dynamic load-balancing strategies, with block size 8^3. As seen from the table, recursive coordinated bisection (RCB) is the best load-balancing approach when the number

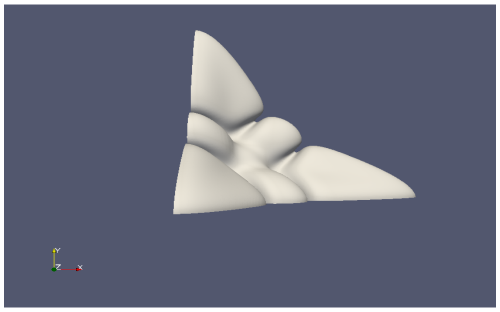


FIGURE 8. Isosurface $\phi = 0$ at the start of each test.

of the tasks is small (≤ 48), while for a medium to large number of tasks Morton ordering performs the best.

TABLE 2. Case I: Performance (i.e., execution time in seconds) for 40 time steps with block size 8^3.

3D Phase field isothermal binary alloy solidification						
No.of processors	24	48	96	192	384	768
Load-balancing						
MO	4116	2422	1373	803	564	491
LMO	4741	2724	1772	1249	1034	758
Weighted GP	4100	2338	1476	1000	770	645
Unweighed GP	4257	2422	1517	1107	833	700
RCB	3969	2263	1429	1037	791	663

The results in Table 2 can be understood through close inspection of the partitions created after the final time step, as illustrated in Tables 3 - 4. As discussed in Section 2, there are two main activities in the computational phase of the multigrid algorithm: restriction/interpolation and smoothing. In restriction and interpolation operations data must be transferred between processes if a parent and one or more of its children are not in the same process. We represent the volume of this interprocessor communication between parents and children by finding the number of parent-child edges which cut across the processes (denoted as x in the introduction to this section). On the other hand, for the smoothing step each block exchanges data with its neighbours through the use of its guard cells. If some neighbouring blocks are not on the same process, this data exchange happens between processes. The volume of this communication is represented by the number of neighbouring blocks which are on the different processes (defined as y in the introduction to this section). The "interprocessor communications" column of Table 3 (and subsequent tables) denotes these values $x + y$ respectively.

When the number of cores/subdomains is 24, the load-balancing at every level is best for Level Morton ordering. However, its volume of interprocessor communications is vastly greater than that of the other strategies. RCB results in the second best load-balancing situation. Although the volume of communications arising from the RCB ordering is larger than that of the remaining partitioning strategies, the

Table 3. Case I: Maximum number of blocks at each multigrid level, block size 8^3, 24 processors. Interprocessor communications are represented as: parent-child cut edges + sibling or neighbour cut edges at the same level.

3D Phase field binary alloy solidification									
multigrid Level Load-balancing	1	2	3	4	5	6	7	8	interprocessor communications
MO	1	7	57	159	334	697	1413	4871	54+2770
LMO	1	1	3	9	30	142	778	4543	12808+2680
Weighted GP	1	7	51	121	229	483	1153	5296	39+2973
Unweighted GP	1	8	58	125	129	266	1000	5129	309+3087
RCB	1	8	27	92	164	394	1126	4851	1450+2489

Table 4. Case I: Maximum number of blocks at each multigrid level, block size 8^3. 768 processors. Interprocessor communications are represented as: parent-child cut edges + sibling or neighbour cut edges at the same level.

3D Phase field binary alloy solidification									
multigrid Level Load-balancing	1	2	3	4	5	6	7	8	interprocessor communications
MO	1	3	31	61	54	86	112	152	52+322
LMO	1	1	1	1	1	5	25	142	430+344
Weighted GP	1	4	32	60	56	123	139	167	37+385
Unweighted GP	1	4	32	64	77	105	136	168	71+408
RCB	1	2	27	46	56	68	110	163	228+316

performance gain from superior load-balancing at every multigrid level more than compensates for this.

For a large number of tasks, such as 768 cores/domains, Morton ordering has the advantage over other algorithms. Notice from Table 4 that the maximum number of blocks at each multigrid level is relatively small, as is the interprocessor communication (especially the parent-child communication used in grid transfer operations). Although Level Morton results in perfect load-balancing, the associated volume of interprocessor communications is too large and causes poor performance. The above results show that for large numbers of cores the interprocessor communications dominate the performance of the multigrid solver, especially when the smoother is relatively inexpensive.

In the second case the block size considered in simulations is 16^3. The tip location in the restart file is 287.01 and the initial time is 203.26. In this case we run for 20 time steps to compare the performance. At the end of each test run, the tip location of the dendrite has moved to location 287.31. The initial number of the blocks is 18929, and there were no grid adaptations during simulations. The maximum number of the refinement levels is 7, which means that the finest mesh is the same size as for the previous example. However the total number of degrees of freedom is greater due to the fact that the larger block size leads to less sharp refinement regions.

TABLE 5. Case II:Performance (i.e., execution time in seconds) for 20 time steps, block size 16^3.

3D Phase field binary alloy solidification						
No.of processors	24	48	96	192	384	768
Load-balancing						
MO	3107	1766	959	597	379	280
LMO	3348	1665	1145	853	489	380
Weighted GP	3112	1773	983	668	447	311
Unweighed GP	3280	1809	1008	710	479	340
RCB	2926	1692	1064	723	491	374

TABLE 6. Case II: Maximum number of blocks at each multigrid level, block size 16^3. 24 processors. Interprocessor communications are represented as: parent-child cut edges + sibling or neighbour cut edges at the same level.

3D Phase field binary alloy solidification.								
multigrid Level	1	2	3	4	5	6	7	interprocessor
Load-balancing								communications
MO	1	7	57	155	279	278	692	43+760
LMO	1	1	3	9	29	124	625	2058+795
Weighted GP	1	8	56	143	242	224	752	21+708
Unweighted GP	1	8	59	133	154	294	756	106+814
RCB	1	8	27	92	140	190	704	430+683

Compared with the 8^3 case, for a single block the amount of computations is eight time larger and the communication four times larger, thus the computation/communication ratio increases. Furthermore, the total number of blocks is decreased to 18929, about 1/7 of that in the 8^3 case. Nonetheless, from Table 5 we see a similar comparison to the 8^3 case. That is, RCB performance is the best when the number of tasks is small ($\leq$ 48) and Morton ordering outperforms the others when the number of tasks is larger. The performance with Level Morton ordering still suffers from the large amount of interprocessor communications (see Tables 6 - 7). As for the 8^3 case, the good load balance (at all levels) of RCB is most important for small core counts (where computation dominates), but lower communication costs (especially at grid transfer) of MO are more important for larger core counts.

In the third case the block size considered in this problem is 32^3. The tip location at the start time step is 287.77 and the initial time is 195.78. The number of time steps taken for performance comparison is 10. The tip location of the dendrite at the end of the 10th time step has moved to 288.02. The maximum refinement level is 6, so the finest mesh size is consistent with the prior examples. The starting total number of blocks is 3073 and it increases to 3145 at the end of the run, where 2 grid adaptations were carried out. In this case the computation/communication ratio for a single block is again twice of that in the previous case. Consequently, load-balancing is expected to dominate the performance more than the communication costs do, even as the core count increases.

TABLE 7. Case II: Maximum number of blocks at each multigrid level, block size 16^3. 768 processors. Interprocessor communications are represented as: parent-child cut edges + sibling or neighbour cut edges at the same level.

3D Phase field binary alloy solidification.								
multigrid Level / Load-balancing	1	2	3	4	5	6	7	interprocessor communications
MO	1	1	15	16	22	22	23	37+88
LMO	1	1	1	1	1	4	20	74+88
Weighted GP	1	3	20	24	17	24	26	39+96
Unweighted GP	1	4	22	22	24	24	25	50+92
RCB	1	1	12	14	14	22	25	57+96

TABLE 8. Case III: Performance (i.e., execution time in seconds) for 10 time steps, block size 32^3.

3D Phase field binary alloy solidification						
No.of processors / Load-balancing	24	48	96	192	384	768
MO	5583	3069	1759	1087	752	357
LMO	4940	2688	1621	1065	794	612
Weighted GP	5073	2788	1677	937	545	330
Unweighed GP	5295	2971	1682	1004	599	348
RCB	4992	2696	1661	1042	699	476

TABLE 9. Case III: Maximum number of blocks among processors at each multigrid level, block size 32^3. 24 processors. Interprocessor communications are represented as: parent-child cut edges + sibling or neighbour cut edges at the same level.

3D Phase field binary alloy solidification.							
multigrid Level / Load-balancing	1	2	3	4	5	6	interprocessor communications
MO	1	3	31	70	93	112	34+230
LMO	1	1	3	9	24	93	398+322
Weighted GP	1	4	32	72	80	122	14+199
Unweighted GP	1	6	40	88	114	124	37+244
RCB	1	4	27	52	52	116	120+207

The simulation results show a different outcome from the previous cases. From Table 8 we see that Level Morton ordering wins for small to medium numbers ($\leq$ 96) of tasks, while the weighted graph partitioning is the best strategy for a large number of cores/subdomains. Table 9 shows the load-balancing situation and volume of communications when the number of tasks is 24. For small the number of cores the superior load balance of Level Morton at each level out-weighs the significantly greater communication requirements in this case.

Table 10. Case III: Maximum number of blocks among processors at each multigrid level, block size 32^3. 768 processors. Interprocessor communications are represented as: parent-child cut edges + sibling or neighbour cut edges at the same level.

3D Phase field binary alloy solidification.							
multigrid Level / Load-balancing	1	2	3	4	5	6	interprocessor communications
MO	1	1	4	4	4	4	28+24
LMO	1	1	1	1	1	3	22+33
Weighted GP	1	1	4	4	4	4	18+26
Unweighted GP	1	2	4	5	5	5	22+26
RCB	1	1	4	4	4	5	20+24

On the other hand, for a large number of cores the difference on the number of blocks among algorithms is reduced. From Table 10, weighted graph partitioning is seen to perform best due to the smallest volume of communications whilst also having a moderately good load balance. Although Level Morton ordering exhibits excellent load-balancing, the relatively large volume of communication again slows down the solver's performance.

5.2. 3D non-isothermal binary alloy solidification. The second test problem (Case IV) simulates the dendrite growth of a non-isothermal alloy, as described in Section 4.2. In this model three variables ϕ (phase), c (solute) and T (temperature) are solved and updated as the time step evolves. Due to the more costly smoothing outlined in Section 4.2, we expect that the load-balancing issue to contribute more than the interprocessor communication to the performance of the multigrid solver. The parameters used in this case are listed in Table 11 and the crystal shape, which is used as a start point for our simulation is shown in Figure 9.

Table 11. Parameters for simulations: 3D non-isothermal binary alloy solidification.

Parameters for 3D non-isothermal binary alloy solidification				
Domain	ϵ	Le	M_0	M_1
800^3	0.02	100	0.009	0.00729

In this example we only present results for the 8^3 block case since the 16^3 cases that we considered yield identical conclusions (we did not undertake experiments for 32^3 blocks in this case). The initial condition corresponds to a tip location along each coordinate axis at 157.82 and the experiments run the simulation for 10 time steps. At the end of the 10th step the tip location of the dendrite has moved to location 157.90. The maximum number of the refinement levels is 7 and there is no mesh refinement during the simulation. Hence the number of blocks is 7801 throughout and no mesh adaptation overhead issue is considered here. Therefore the results provide a pure comparison of the load-balancing algorithms alone.

The results exhibited in Tables 12 - 14 show that Level Morton ordering provides the optimal block arrangement for all of the cases considered. This is partly

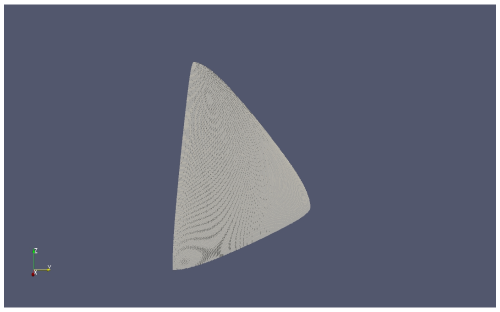


FIGURE 9. Isosurface $\phi = 0.5$ for initial condition used in the non-isothermal phase-field case.

TABLE 12. Case IV: Performance (i.e., execution time in seconds) for 10 time steps, block size 8^3.

3D non-isothermal binary alloy solidification.						
No.of processors	24	48	96	192	384	768
Load-balancing						
MO	8109	4358	2435	1969	1327	776
LMO	7006	3543	2132	1800	944	609
Weighted GP	11436	6072	3866	3259	2085	1150
Unweighed GP	11992	7115	4224	3337	2219	1145
RCB	10307	5801	3040	2924	1832	953

TABLE 13. Case IV: Maximum number of blocks among processors at each multigrid level, block size 8^3. 24 processors. Interprocessor communications are represented as: parent-child cut edges + sibling or neighbour cut edges at the same level.

3D non-isothermal binary alloy solidification.								
multigrid Level	1	2	3	4	5	6	7	interprocessor communications
Load-balancing								
MO	6	6	8	18	32	85	275	622+215
LMO	2	2	3	7	16	56	242	970+448
Weighted GP	24	24	60	127	120	136	312	17+425
Unweighted GP	25	32	59	109	98	162	309	53+441
RCB	25	32	36	76	60	102	293	233+413

explained by the load-balancing data in Tables 13 and 14, which show that Level Morton ordering provides by far the best load balance at each level. As also expected, the volume of interprocessor communications is the largest for the Level Morton ordering, however the much greater computation to communication ratio in this example means that this weakness has less impact than previously. That is, the load-balancing issue at each multigrid level is a more important impact factor to the solver's performance than interprocessor communications in this problem.

Table 14. Case IV: Maximum number of blocks at each multigrid level, block size 8^3. 768 processors. Interprocessor communications are represented as: parent-child cut edges + sibling or neighbour cut edges at the same level.

3D non-isothermal binary alloy solidification								
multigrid Level Load-balancing	1	2	3	4	5	6	7	interprocessor communications
MO	2	2	3	6	9	10	11	70+59
LMO	1	1	1	1	1	1	8	45+53
Weighted GP	11	8	8	9	9	10	11	20+52
Unweighted GP	10	8	8	9	10	10	11	33+51
RCB	10	6	8	8	10	10	11	34+50

Note that we have also applied a similar set of tests to a third phase-field problem involving the multiphase modelling of eutectic growth in two space dimensions based upon [26, 27]. This problem also involved very expensive smoother and therefore a large computation to communication ratio. The conclusions derived were the same as in this test case: that the improved load-balancing of the Level Morton scheme out-weighed the inferior communication volume. In future work we will assess load balancing strategies on yet more phase-field examples, such as the 6th order, three dimensional functionalized Chan-Hilliard (FCH) equation described in [28]. This applies a modification to the FAS algorithm to ensure mass conservation, which is not yet included in our current work.

6. Conclusion

From the experimental results, we observe that the relative performance of the different load-balancing algorithms depends strongly on the computational features of the tested problems. These features are two fold: the computational cost of s-moothing and the communication cost of smoothing and grid transfer operations (interpolation/restriction). Smoothing consists of local Jacobi matrix inversion and communications (guard cell information exchanging) between sibling block-s, while interpolation/restriction requires communications between parent-child blocks. Both components are $O(n)$ operations (n as the degrees of freedom) but their relative cost depends on the amount of work within the smoother (i.e., the size of the local Jacobian). For all cases, Level Morton ordering causes the largest volume of interprocessor communications, of which a large portion involves communication between parent-child blocks. On the other hand, Morton ordering preserves good locality and provides the block distribution with the smallest amount of interprocessor communications.

According to the test results, the impact of the load-balancing algorithms on the performance of the nonlinear multigrid solver may be summarized as follows. First-ly, for simple smoothers where a larger proportion of time is spent in communication between neighbouring blocks (due to the simpler smoother and smaller block size, hence lower computation to communication ratio), Morton ordering is recommend-ed: e.g. the isothermal problem with block sizes 8^3 and 16^3. As the block size (and therefore the computation to communication ratio) increases, such as in case III (32^3), either weighted graph partitioning (for 768 process) or RCB (for 24 processes) is the most competitive load-balancing strategy. In this case the computation time contributes a significant part of the solution time, thus both load-balancing and interprocess communications are important. Conversely, for problems with expensive smoothers, such as used in 3D non-isothermal alloy solidification, our newly proposed Level Morton strategy is recommended. In this situation load balance is critical to the solver's performance and Level Morton ordering provides nearly perfect load balance at all multigrid levels.

In every case, there is trade off on the performance of the multigrid solver between load-balancing and processor communications. For computationally intensive problems Level Morton ordering is a more suitable partitioning where good load-balancing at each multigrid level is more desirable than optimal inter-processor communications.

Acknowledgments

The work undertaken by P. C. Bollada and P. J. Jimack was funded by EPSRC Innovative Manufacturing Research Hub in Liquid Engineering (LiME), Grant No. EP/N007638/1.

References

[1] G.D. Smith. *Numerical solution of partial differential equations: finite difference methods.* Oxford University Press, 3rd ed., 1985.

[2] J. Strikwerda. *Finite Difference Schemes and Partial Differential Equations.* SIAM., 2nd ed., 2004.

[3] J. N. Reddy. *An Introduction to the Finite Element Method.* Third ed.) McGraw-Hill., 3rd ed., 2006.

[4] D. Komatitsch, J. Labarta, and D. Mich ?ea. *A simulation of seismic wave propagation at high resolution in the inner core of the Earth on 2166 processors of MareNostrum.* High Performance Computing for Computational Science-VECPAR, 364C377, 2008.

[5] H. Bake, M. V. Jayaraman, P. D. Richardson, G. E. Karniadakis, *Flow instability and wall shear stress variation in intracranial aneurysms.* Journal of Royal Society, Interface, 7(47), 967-88, 2010.

[6] P.C. Bollada, C.E. Goodyer, P.K. Jimack, A. Mullis and F.W. Yang. *Three dimensional thermal-solute phase field simulation of binary alloy solidification.* Journal of Computational Physics, 287: 130C150 2015.

[7] N. Provatas, N. Goldenfeld, J. Dantzig *Adaptive Mesh Refinement Computation of Solidification Microstructures Using Dynamic Data Structures.* Journal of Computational Physics, 148, 265-290, 1999.

[8] J. Rosam, P. K. Jimack, A. M. Mullis. *An adaptive, fully implicit multigrid phase-field model for the quantitative simulation of non-isothermal binary alloy solidification.* Acta Materialia, 56, 4559-4569, 2008.

[9] C.E. Goodyer, P.K. Jimack, A.M. Mullis, H.B. Dong, and Y. Xie. *On the fully implicit solution of a phase-field model for binary alloy solidification in three dimensions.* Advances in Applied Mathematics and Mechanics, 4: 665C684, 2012.

[10] F. W. Yang, C. E. Goodyer, M. E. Hubbard, P. K. Jimack *Parallel implementation of an adaptive, multigrid solver for the implicit solution of nonlinear parabolic systems, with application to a multi-phase-field model for tumour growth.* Civil Comp Proceedings (CCP 107), 2015.

[11] N. Touheed, P. Selwood, P.K. Jimack, and M. Berzins, *A comparison of some dynamic load-balancing algorithms for a parallel adaptive flow solver.* Parallel Computing, 26, 1535C1554, 2000.

[12] R. D. Williams. *Performance of dynamic load balancing algorithms for unstructured mesh calculations.* Concurrency: Practice and Experience, 3:457-481, 1991.

[13] G. M. Morton. *A computer Oriented Geodetic Data Base; and a New Technique in File Sequencing.* Technical Report, Ottawa, Canada: IBM Ltd, 1966.

[14] P. MacNeice, K.M.Olson, C. Mobarry, R. de Fainchtein and C. Packer. *PARAMESH: A parallel adaptive mesh refinement community toolkit.* Computer Physics Communications, 126: 330-354, 2000.

[15] A. Brandt. *Multi-level adaptive solutions to boundary-value problems*, Mathematics of Computation 31: 333C390, 1977.

[16] K. Devine, E. Boman, R. Heaphy, B. Hendrickson, C. Vaughan. *Zoltan data management services for parallel dynamic applications.* Computing in Science and Engineering 4: 90C97, 2002.

[17] K. Devine, E. Boman, U. Catalyurek, V. Leung, S. Rajamanickam, M. Wolf. *Zoltan: Parallel Partitioning, Load Balancing and Data-Management Services.*.

[18] F.W. Yang, C.E. Goodyear, P.K. Jimack, M.E. Hubbard. *An Optimally Efficient Technique for the Solution of Systems of Nonlinear Parabolic Partial Differential Equations.* Advances in Engineering Software, 103: 65-84, 2017.

[19] F. Pelligrini. *PT-SCOTCH 5.1 users guide.* Technical report, LaBRI, 2008.

[20] G. Karypis, K. Schloegel, V. Kumar. *Parmetis: Parallel graph partitioning and sparse matrix ordering library, version 3.1.* Technical report, Dept. Computer Science, University of Minnesota (2003)

[21] D. Bozdag, A. Gebremedhin, F. Manne, E. Boman, U. Catalyurek. *A framework for scalable greedy coloring on distributed-memory parallel computers.* Journal of Parallel and Distributed Computing 68: 515C535, 2008.

[22] A. Iserles. *A First Course in the Numerical Analysis of Differential Equations.* Cambridge University Press, 1996.

[23] P.C. Bollada, P.K. Jimack, A.M. Mullis. *Bracket formalism applied to phase field models of alloy solidification.* Computational Materials Science 126: 426-437, 2017

[24] H. L. Lukas, S. G. Fries and B. Sundman, *Computational Thermodynamics*, the Calphad Method, Cambridge University Press, 2007.

[25] A. T. Dinsdale, *SGTE Data for Pure Elements.* Edition 2, ASM International, pp T13-T15.

[26] P.C. Bollada, P.K. Jimack. *New approach to multi-phase formulation for the solidification of alloys.* Physica D: Nonlinear Phenomena 241, issue 8: 816-829, 2012

[27] G. I. Toth, T. Pusztai, L. Granasy. *Consistent multiphase-field theory for interface driven multidomain dynamics* Physical Review, B 92, 184105, 2015.

[28] W. Feng, Z. Guo, J.S. Lowengrub, S.M. Wise. *A mass-conservative adaptive FAS multigrid solver for cell-centred finite difference methods on block-structured, locally-cartesian grids.* Journal of Computational Physics, 352: 463-497, 2018.

School of Computing, University of Leeds, Leeds, LS2 9JT, UK
E-mail: `m.h.chen@leeds.ac.uk`

School of Computing, University of Leeds, Leeds, LS2 9JT, UK
E-mail: `p.c.bollada@leeds.ac.uk`

School of Computing, University of Leeds, Leeds, LS2 9JT, UK
E-mail: `p.k.jimack@leeds.ac.uk`