INSTITUTE OF PHYSICS PUBLISHING MODELLING AND SIMULATION IN MATERIALS SCIENCE AND ENGINEERING

Modelling Simul. Mater. Sci. Eng. **14** (2006) 1189–1195 doi:10.1088/0965-0393/14/7/007

Sparse data structure and algorithm for the phase field method

J Gruber[1], N Ma[2], Y Wang[2], A D Rollett[1] and G S Rohrer[1]

[1] Department of Materials Science and Engineering, Carnegie Mellon Univresity, 5000 Forbes Avenue, Pittsburgh, PA 15213, USA
[2] Department of Materials Science and Engineering, The Ohio State University, 2041 College Road, Columbus, OH 43210, USA

Received 7 March 2006, in final form 25 July 2006
Published 19 September 2006
Online at stacks.iop.org/MSMSE/14/1189

Abstract

The concepts of sparse data structures and related algorithms for phase field simulations are discussed. Simulations of polycrystalline grain growth with a conventional phase field method and with sparse data structures are compared. It is shown that memory usage and simulation time scale with the number of nodes but are independent of the number of order parameters when a sparse data structure is used.

1. Introduction

Modelling moving boundary problems is the most typical application of the phase field method [1–3]. In such systems the regions of most activity are those around interfaces, which account for a relatively small portion of the simulation domain. Near interfaces the values of order parameters change rapidly, while elsewhere the order parameters remain at one of a few 'ambient' values.

In conventional phase field simulations, the spatial domain Ω is discretized into a set of N_Ω nodes, and the values of N_η order parameters are tracked at each node [4, 5]. Both the memory usage and simulation time in a conventional approach scale as $N_\Omega N_\eta$. There is then a tradeoff between allowable system size and complexity due to hardware limitations and time constraints.

Efforts have been taken in the literature to overcome these limitations. Krill and Chen [6] dynamically reassigned the spatial distribution of order parameters to reduce the number of order parameters required to avoid frequent domain coalescence. Such an approach is limited to systems where the only use of the order parameters is to differentiate unique domains. Kobayashi and Warren [7, 8] used one order parameter coupled with an orientation field to describe polycrystalline materials. Although this approach leads to fewer calculations related to the order parameters, it requires additional equations of motion for the orientation fields. Ma and Wang [9] developed an algorithm that accounts for time evolution of order parameters at only those regions where finite gradients of the order parameters are present. The approach significantly reduces the simulation time required, but demands the same amount of memory

usage as the conventional phase field algorithm. Adaptive meshing [10–14] has been used in phase field simulations to increase computational efficiency, but implementation of such schemes could be difficult.

Many applications of the phase field method, however, present opportunities for more efficient data structures and algorithms. As an example, we consider phase field models of polycrystalline grain growth. In typical simulations of grain growth, each order parameter represents a unique grain orientation [4]. In a sufficiently coarse polycrystalline microstructure, most of the domain nodes lie within the interiors of the grains. Within a grain only one orientation is non-zero and, therefore, only one order parameter value need be stored for each node. We will call methods which exploit this fact sparse phase field methods, by analogy to the sparse matrix methods of numerical linear algebra [15]. One storage scheme for this problem is discussed in the following section, and it will be shown that its memory usage scales as N_Ω, and so is independent of the number of order parameters.

Additionally, many computations required by the phase field method can be reformulated to take advantage of a sparse storage scheme. In particular, the updated value for each order parameter at a node depends on the values of all order parameters in a small neighbourhood. Usually, the update computation only needs to be performed for order parameters which differ from the ambient value; for the case of grain growth, only the non-zero order parameters in any neighbourhood must be updated. An algorithm for phase field simulations of grain growth is presented and shown to scale as N_Ω.

We note that the concepts presented in this paper could easily be applied to a number of other moving boundary problems, including anisotropic grain growth, recrystallization, particle coarsening, solidification, dislocation motion and order/disorder transformations in any number of dimensions.

2. Data structure

In phase field methods for simulating polycrystalline grain growth, a crystal orientation is described by a unique order parameter η_i such that $0 \leqslant \eta_i \leqslant 1$. Within a grain of the orientation specified by η_i, the order parameter is unity. The value of the order parameter changes continuously from 1 to 0 in interface regions separating grains of different orientation. The width of interface regions is relatively narrow, such that the total area (volume) associated with interface regions is only a small fraction of the total domain area (volume). Therefore, at most nodes only one order parameter is non-zero.

A simple data structure that exploits this fact is one that maintains a dynamic list or vector of non-zero order parameters at each node. Such a list must contain items that hold both the value and index of an order parameter. A requirement of such a storage scheme is to have some indexing functions to both set and retrieve the values of an arbitrary order parameter anywhere in the domain. For the special algorithms described below, we must also be able to iterate through the list of non-zero order parameters at any node. The implementation described in this paper uses the vector class from the C++ STL [16] for maintaining the order parameter data. The source code is available by contacting the authors. When using such a data structure, order parameter updates require different treatment depending on the magnitude of the update value. This could mean adding new storage to the data structure when the number of non-zero order parameters at a node increases, changing the value of an existing non-zero order parameter or deleting data for an order parameter which has dropped below some threshold value. It would be possible to use a threshold of exactly zero, but this is both inefficient and unnecessary. We choose a small value ε as the threshold, and any order parameter with value less than ε is considered to be zero and is not retained. The effects of our choice of ε are described in detail later.

For sufficiently coarse systems, the average number of non-zero order parameters at a point is approximately one. Thus the total memory used by such a data structure scales only with the number of nodes N_Ω.

3. Algorithm

The evolution of the system for each order parameter is described by the time-dependent Ginzburg–Landau (Allen–Cahn) equation:

$$\frac{\partial \eta_i}{\partial t} = -L\frac{\delta F}{\delta \eta_i}, \tag{1}$$

where L is the kinetic coefficient that characterizes grain boundary mobility,

$$F = F_0 + \int_V \left[f_0 + \kappa \sum_{i=1}^{N_\eta} (\nabla \eta)^2 \right] \mathrm{d}V \tag{2}$$

is the total free energy expressed on a coarse-grained level, κ is the gradient energy coefficient, and f_0 is the local free energy. The exact form of f_0 is not important as long as it provides degenerate minima corresponding to each grain orientation. A simple form that satisfies this requirement is

$$f_0 = \frac{a}{2}\sum_{i=1}^{N_\eta}\left(-\eta_i^2 + \frac{1}{2}\eta_i^4\right) + \sum_{i=1}^{N_\eta}\sum_{j>i}^{N_\eta}\frac{b}{2}\eta_i^2\eta_j^2, \tag{3}$$

where N_η is total number of order parameters and a and b are phenomenological constants with their values determined by grain boundary energy and width. It can be shown that if $b > a$, minimization of the total free energy yields an equilibrium order parameter profile η_i that changes continuously from 1 inside grain i to 0 outside. In the conventional phase field method, (1) is solved for all order parameters at all nodes.

With a sparse data structure we can simplify the computation of (2) and (3) by iterating through only non-zero order parameters at the node. More importantly, if the value of any order parameter is zero at all nodes in a neighbourhood, the value of the same order parameter at the centre node will remain zero during the next time step, according to equations (1)–(3). Therefore an update value for an order parameter at any node need be calculated only if it is non-zero at some neighbour. The revised algorithm is then to first generate a list of unique non-zero order parameters in the neighbourhood of each node, perform the update calculation for only those order parameters in the list and repeat for all other nodes. In a sufficiently coarse system, the number of non-zero nodes in any neighbourhood is usually one in a grain interior and rarely larger than three or four at a 2D triple junction. The use of a sparse data structure therefore allows the computation of node updates to scale independently of the number of order parameters, and hence simulation time scales with N_Ω.

4. Simulations

An initial microstructure was created as follows. One integer Q, where $1 \leqslant Q \leqslant 1000$, was assigned randomly to each point on a 512×512 square two-dimensional lattice. This grid was coarsened using a standard Monte Carlo model for isotropic grain growth [17] until 383 grains remained. For simulations with the conventional phase field method, the number of unique orientations was reduced to 100 by the reassignment of $Q' = Q \pmod{100}$.

Table 1. Performance and accuracy (similarity factor) of different phase field algorithms. The simulations were carried out on workstations with Intel Xeon 2.4 GHz processors and 512 MB physical memory. The results correspond to 10,000 iterations.

Method	# of order parameters	Memory (MB)	CPU Time (min)	Similarity factor
Conventional PF	100	407	1234	Base
MW algorithm [9] $\Delta\varepsilon = 0.0001$	100	407	165	0.005 306
Sparse PF, $\varepsilon = 0.01$	100	21	42	0.116 920
Sparse PF, $\varepsilon = 0.001$	100	28	53	0.001 297
Sparse PF, $\varepsilon = 0.0001$	100	30	55	0.000 019
Sparse PF, $\varepsilon = 0.001$	383	28	53	0.044 666

In the current study, the parameters in equations (1–3) are chosen as the following: $\kappa = 1$, $a = 1$, $b = 2$, $L = 1$ and $\mathrm{d}x = 2.0$, all in dimensionless units. The typical width of a grain boundary is approximately $3\mathrm{d}x$, which is sufficiently diffuse for smooth grain boundary migration driven by curvature and parabolic grain growth kinetics [18, 19]. The Laplacian is calculated using a centred method up to second nearest neighbours.

Simulations with 100 order parameters were performed using the conventional phase field method, the MW algorithm [9], and the sparse phase field method with various values assigned to ε. Additionally, a simulation in which all grains (383) had unique orientations was performed using the sparse phase field method.

5. Results and discussion

The memory usage and real time for 10 000 iterations of each method are summarized in table 1. It can be seen that the sparse phase field method with $\varepsilon = 0.001$ used about 7% of the memory and ran approximately 4% of the time used in the conventional phase field method. The sparse phase field method required about 32% of the time used for the MW algorithm.

To compare the accuracy of the sparse phase field algorithm with that of the conventional phase field algorithm, we use the similarity factor introduced by Demirel *et al* [20]

$$S = \frac{1}{A}\sum_{i,j}\left[\delta\left(I_{ij} - I'_{ij}\right) - 1\right]L, \tag{4}$$

where A is the total area of a system considered, I_{ij} and I'_{ij} are the index numbers of node (i, j) for the two microstructures, respectively, and δ is the Kronecker delta. The index number at a given node is defined as the index of the largest order parameter at that node. Complete agreement and complete disagreement are denoted by $S = 0$ and $S = 1$, respectively. The final microstructure obtained by the conventional phase field algorithm is used as the reference for comparison. The similarity factors for each case at 10 000 iterations are listed in table 1. When the parameter $\varepsilon = 0.001$, the similarity factor reaches 0.001 297, indicating that almost identical final microstructures are obtained by the conventional and sparse phase field methods. This can be seen directly from the superposition of the two microstructures shown in figure 1.

The sparse phase field simulation with 383 unique orientations was carried out with $\varepsilon = 0.001$. It is clear that the performance of the sparse phase field method is independent of the number of order parameters used. This is a significant advancement over the conventional phase field algorithm because a limited number of orientations may lead to an artificially high coalescence event. To evaluate this effect quantitatively, we superimposed the final

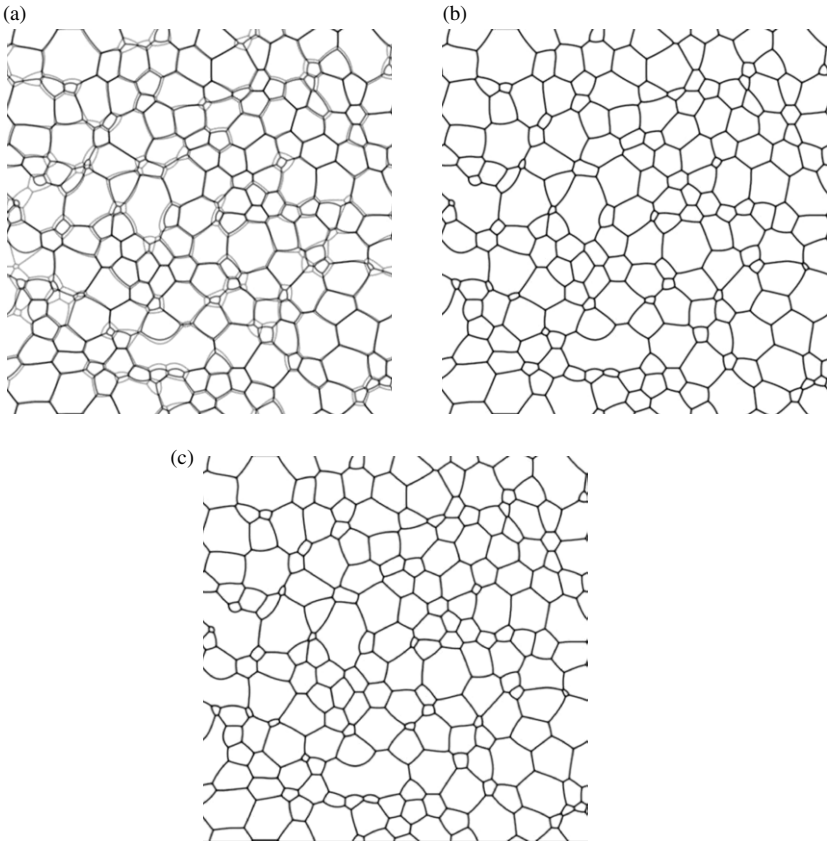

Figure 1. Superpositions of two final microstructures obtained by the conventional phase field algorithm (dark lines) and the sparse phase field algorithm (light grey lines) with different ε values. (*a*) $\varepsilon = 0.01$, $S = 0.116\,920$; (*b*) $\varepsilon = 0.001$, $S = 0.001\,297$; and (*c*) $\varepsilon = 0.0001$, $S = 0.000\,019$. When $\varepsilon \leqslant 0.001$, the two microstructures become indistinguishable.

microstructure obtained in this simulation on the one obtained using only 100 order parameters, (figure 2). The grain size distributions of the two microstructures are also calculated and the results are plotted in figure 3. Significant differences in grain morphology, average grain size and grain size distribution are observed. This is reflected in the similarity factor calculation for the two microstructures, i.e. $S = 0.045$. These differences are expected to increase for polycrystalline systems with anisotropic boundary properties and for three-dimensional systems.

The algorithm presented here becomes increasingly efficient as grain size increases since the average number of non-zero order parameters per node decreases. Note that there is overhead associated with storing order parameter indices. Simulations with a very small total number of order parameters should use the conventional phase field method. For simulations starting with random noise or where the grain size is comparable to the boundary thickness, the conventional phase field algorithm should be applied until a well defined polycrystalline microstructure is developed. At this point the sparse phase field method could be used. Such a sequence is often employed in Monte Carlo simulations of grain growth where 'brute force' is most efficient at short times but 'continuous time' becomes more efficient once a sufficiently sparse microstructure develops [21].

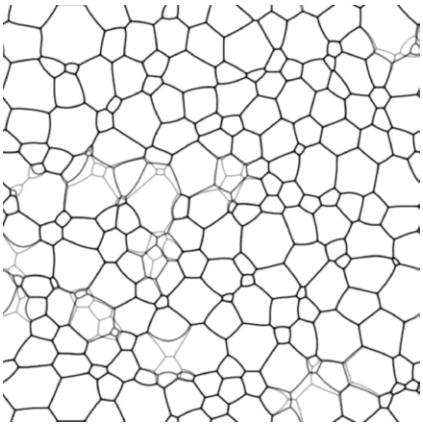

Figure 2. Comparison of the final microstructure obtained using the sparse phase field algorithm with unique orientation for each grain (light grey lines) with that obtained using the conventional phase field algorithm with 100 grain orientations.

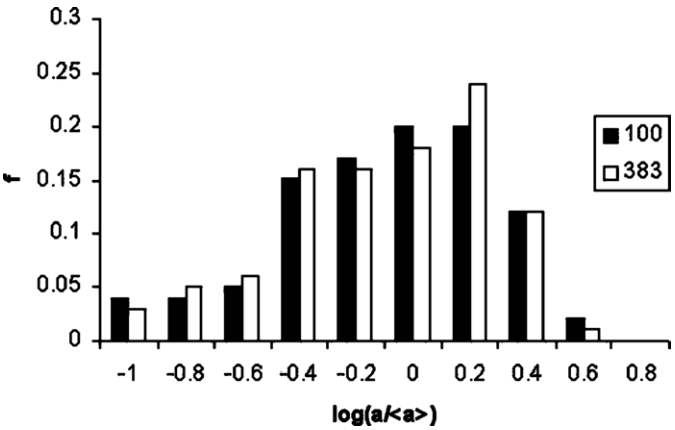

Figure 3. Comparison of grain size distribution obtained by the sparse phase field algorithm with 383 orientations (one for each grain) (open bars) with the one obtained by the conventional phase field algorithm with 100 grain orientations (filled bars). The average grain area is 1181 pixels for the former and 1225 for the latter.

6. Summary

A sparse phase field algorithm for phase field simulations has been developed. Both the memory usage and simulation time of the new algorithm are independent of the number of order parameters used in the simulation. Results obtained using the new algorithm were compared with those obtained by conventional phase field algorithms. For a two-dimensional system of 512×512 nodal points with 100 order parameters, the sparse phase field algorithm used approximately 7% of the physical memory of the conventional algorithms. After 10 000 iterations, the sparse phase field algorithm had used only 4% of the total simulation time of the conventional phase field algorithm and 32% of that of the fastest algorithm available. The possible effects caused by limited number of grain orientations affordable for the conventional algorithms on grain size distribution are investigated. Significant differences are observed

between two simulations with 100 and 383 order parameters. The sparse phase field algorithm presented here could be easily extended to grain growth in three dimensions and to systems with anisotropic grain boundary properties. Similar algorithms could also be developed for a number of other moving boundary problems.

Acknowledgments

The work at Carnegie Mellon University was supported by the MRSEC programme of the National Science Foundation under Award Number DMR-0520425 and by the Computational Materials Science Network programme of the Office of Basic Energy Sciences, Department of Energy. The work at The Ohio State University was supported by the US Air Force through the Metals Affordability Initiative programme, Grant Number F33615-99-2-5215, and by the Office of Naval Research through the D3D program, Grant Number N00014-05-1-0504.

Notes added in proof.

It has been brought to our attention that the method we described in this paper is essentially the same as that described in a recently published paper by Vedantam and Patnaik [22].

References

[1] Wang Y Z and Chen L Q 2000 Simulation of microstructural evolution using the phase field method *Methods in Materials Research* ed E N Kaufmann (New York: Wiley) chapters 2a.3.1-2a.3.23
[2] Chen L Q 2002 *Ann. Rev. Mater. Res.* **32** 113
[3] Karma A 2001 Phase field methods *Encyclopedia of Materials Science and Technology* (Oxford: Elsevier)
[4] Chen L Q 1995 *Scr. Metall.* **32** 115
[5] Kazaryan A, Wang Y, Dregia S A and Patton B R 2000 *Phys. Rev.* B **61** 14275
[6] Krill C E III and Chen L Q 2002 *Acta Mater.* **50** 3059
[7] Kabayashi R, Warren J A and Carter W C 2000 *Phys.* D **140** 141
[8] Warren, J A, Kobayashi R, Lobkovsky A E and Carter W C 2003 *Acta Mater.* **51** 6035
[9] Ma N and Wang Y unpublished
[10] Wang S L and Sekerka R F 1996 *Phys. Rev.* E **53** 3760
[11] Braun R J, Murray B T and Soto J 1997 *Modelling Simul. Mater. Sci. Eng.* **5** 365
[12] Provatas N, Goldenfeld N and Dantzig J 1999 *J. Comput. Phys.* **148** 265
[13] Plapp M and Karma A 2000 *J. Comput. Phys.* **165** 592
[14] Lan C W, Hsu C M, Liu C C and Chang Y C 2002 *Phys. Rev.* E **65** 061601
[15] Saad Y 2003 *Iterative Methods for Sparse Linear Systems* 2nd edn (Philadelphia, USA: SIAM)
[16] Stroustrup B 1997 *The C++ Programming Language* 3rd edn (Boston, USA: Addison-Wesley)
[17] Anderson M P, Srolovitz D J, Grest G S and Sahni P S 1984 *Acta Metall.* **32** 783
[18] Fan D and Chen L Q 1997 *Phil. Mag. Lett.* **75** 187
[19] Fan D and Chen L Q 1997 *Acta Mater.* **45** 611
[20] Demirel M C, Kuprat A P, George D C and Rollett A D 2003 *Phys. Rev. Lett.* **90** 016106
[21] Hassold G N and Holm E A 1993 *Comput. Phys.* **7** 97
[22] Vedantam S and Patnaik B S V 2006 *Phys. Rev.* E **73** 016703