Computer Physics Communications 183 (2012) 2107–2115

Contents lists available at SciVerse ScienceDirect

Computer Physics Communications

journal homepage: www.elsevier.com/locate/cpc

An efficient and accurate numerical algorithm for the vector-valued Allen–Cahn equations

Hyun Geun Lee, Junseok Kim *

Department of Mathematics, Korea University, Seoul 136-701, Republic of Korea

ARTICLE INFO

Article history:
Received 7 February 2012
Received in revised form
11 May 2012
Accepted 13 May 2012
Available online 19 May 2012

Keywords:
Vector-valued Allen–Cahn equations
Operator splitting
Linear geometric multigrid
Grain growth
Multiple crystals growth

ABSTRACT

In this paper, we consider the vector-valued Allen–Cahn equations which model phase separation in N-component systems. The considerations of solving numerically the vector-valued Allen–Cahn equations are as follows: (1) the use of a small time step is appropriate to obtain a stable solution and (2) a sufficient number of phase-field variables is required to capture the correct dynamics. However, stability restrictions on the time step and a large number of phase-field variables cause huge computational costs and make the calculation very inefficient. To overcome this problem, we present an efficient and accurate numerical algorithm which is based on an operator splitting technique and is solved by a fast solver such as a linear geometric multigrid method. The algorithm allows us to convert the vector-valued Allen–Cahn equations with N components into a system of $N-1$ binary Allen–Cahn equations and drastically reduces the required computational time and memory. We demonstrate the efficiency and accuracy of the algorithm with various numerical experiments. Furthermore, using our algorithm, we can simulate the growth of multiple crystals with different orientation angles and fold numbers on a single domain. Finally, the efficiency of our algorithm is validated with an example that includes the growth of multiple crystals with consideration of randomness effects.

1. Introduction

The Allen–Cahn (AC) equation was originally introduced as a phenomenological model for antiphase domain coarsening in a binary alloy [1]. The AC equation and its various modified forms have been applied to a wide range of problems, such as phase transitions [1], image analysis [2,3], motion by mean curvature [4–10], two-phase fluid flows [11], crystal growth [12–14], and grain growth [15–19]. Several techniques have been developed, including boundary integral [20,21], cellular automata [22,23], front-tracking [24,25], level-set [26–28], and phase-field [28–33] methods to solve numerically the AC equation and its various modified forms. Among these various methods, the phase-field method is popular and widely used since it does not require explicit tracking of the interface.

An important class of phase-field models is the multi-phase field model [34,35], in which several phase-field variables are used to describe components. In this paper, we employ a multi-phase field model to solve numerically the vector-valued formulation of the AC equation. The vector-valued AC equations were introduced in Garcke et al. [34] and have been extensively used [36–38], allowing us to consider an arbitrary number of components. In multi-phase field simulations, one has the following considerations: (1) the use of a small time step in a numerical algorithm is appropriate to obtain a stable solution; (2) the grid resolution must be fine enough to accurately resolve the interfaces; and (3) for a problem such as grain growth, a sufficient number of phase-field variables is required to represent discretized grain orientations. However, stability restrictions on the time step and a large number of phase-field variables cause huge computational costs and make the calculation very inefficient.

In many previous papers, implicit Euler's or unconditionally gradient stable (given by Eyre [39]) methods are used to obtain a sufficiently large time step. However, Eyre [39] showed that an implicit Euler's method suffers from instability if a large time step is used. Also, in [40], the authors addressed the unconditional stability of Eyre's scheme and showed that the equivalent time step is bounded by a small value. Moreover, except for an explicit Euler's method, we need to solve nonlinear systems with a large number of phase-field variables and the systems become bigger and bigger for an increasing number of phase-field variables. To overcome this problem, a variety of numerical approaches have been developed, including adaptive mesh refinement [41–45], parallel computing [46,47], and moving meshes [48]. However, such approaches can be difficult to implement or handle only a few phase-field variables.

* Corresponding author. Tel.: +82 2 3290 3077; fax: +82 2 929 8562.
E-mail address: cfdkim@korea.ac.kr (J. Kim).
URL: http://math.korea.ac.kr/~cfdkim/ (J. Kim).

0010-4655/$ – see front matter © 2012 Elsevier B.V. All rights reserved.
doi:10.1016/j.cpc.2012.05.013

In this paper, we present an efficient and accurate numerical algorithm, which is based on an operator splitting technique and is solved by a fast solver such as the linear geometric multigrid method. The algorithm allows us to convert the vector-valued Allen–Cahn equations with N components into a system of $N-1$ binary Allen–Cahn equations and drastically reduces the computational time and memory requirements. Further, the convergence rate of the computing time of our algorithm is linear with respect to the number of phase-field variables.

Although the interaction between multiple crystals has been demonstrated in many studies [26,49–51], the randomness of crystal orientation and fold number on a single domain has not, to our knowledge, been considered. However, using our algorithm, we can simulate the growth of multiple crystals with different orientation angles and fold numbers on a single domain.

This paper is organized as follows. In Section 2, we briefly review a derivation of the vector-valued AC equations. In Section 3, we present an efficient and accurate numerical algorithm for the vector-valued AC equations and its computational advantage is explained. The phase-field model of multiple crystals growth is summarized in Section 4. Numerical experiments showing the efficiency and accuracy of the algorithm are presented in Section 5. Finally, conclusions are drawn in Section 6.

2. The vector-valued Allen–Cahn equations

We consider the evolution of multi-component systems on a polygonal (polyhedral) domain $\Omega \subset \mathbb{R}^d$, $d = 1, 2, 3$. Let $\mathbf{c} = (c_1, \ldots, c_N)$ be a vector-valued phase-field. The components c_i for $i = 1, \ldots, N$ represent mole fractions of different components in the system. Clearly the total mole fractions must sum to 1, i.e.,

$$c_1 + \cdots + c_N = 1, \tag{1}$$

so that admissible states will belong to the Gibbs N-simplex

$$G := \left\{ \mathbf{c} \in \mathbb{R}^N \,\middle|\, \sum_{i=1}^{N} c_i = 1, 0 \le c_i \le 1 \right\}.$$

Without loss of generality, we postulate that the free energy can be written as follows:

$$\mathcal{F}(\mathbf{c}) = \int_\Omega \left(\frac{F(\mathbf{c})}{\epsilon^2} + \frac{1}{2} \sum_{i=1}^{N} |\nabla c_i|^2 \right) d\mathbf{x},$$

where $F(\mathbf{c}) = 0.25 \sum_{i=1}^{N} c_i^2 (1 - c_i)^2$ and $\epsilon > 0$ is the gradient energy coefficient. The natural boundary condition for the vector-valued AC equations is the zero Neumann boundary condition:

$$\nabla c_i \cdot \mathbf{n} = 0 \quad \text{on } \partial\Omega, \tag{2}$$

where $\mathbf{n}$ is the unit normal vector to $\partial\Omega$.

Now we review a derivation of the vector-valued AC equations as a gradient flow under the additional constraint (1), which has to hold everywhere at any time. It is natural to seek a law of evolution in the form

$$\frac{\partial \mathbf{c}}{\partial t} = -\text{grad}\mathcal{F}(\mathbf{c}). \tag{3}$$

The symbol "grad" here denotes the gradient on the manifold in L^2 space.

There are two main approaches in which the constraint (1) can be ensured, either by the use of a variable Lagrange multiplier [32,34,52,53] or by specifying explicitly how the phases vary with respect to one another [16,54,55]. In recent work [56], Bollada et al. proposed two multi-phase formulations that have no N dependence and do not generate spurious additional phases at binary interfaces. Here, in order to ensure the constraint (1), we use a variable Lagrange multiplier $\beta(\mathbf{c})$ [34] and set $\frac{\partial F}{\partial \mathbf{c}} = (\frac{\partial F}{\partial c_1}, \frac{\partial F}{\partial c_2}, \ldots, \frac{\partial F}{\partial c_N}) = \mathbf{f}(\mathbf{c}) = (f(c_1), f(c_2), \ldots, f(c_N))$, where $f(c) = c(c - 0.5)(c - 1)$. Let $\mathbf{1} = (1, \ldots, 1) \in \mathbb{R}^N$. Using a general smooth vector-valued function $\boldsymbol{\xi}$, we set

$$\mathbf{d} = (d_1, d_2, \ldots, d_N) = \boldsymbol{\xi} - \frac{1}{N} \sum_{i=1}^{N} \xi_i \mathbf{1}, \quad \text{then } \sum_{i=1}^{N} d_i = 0.$$

Let $\beta(\mathbf{c}) = (-1/N) \sum_{i=1}^{N} f(c_i)$. Then we have the following:

$$\begin{aligned}
(\text{grad}\mathcal{F}(\mathbf{c}), \mathbf{d})_{L^2} &= \frac{d}{d\eta} \mathcal{F}(\mathbf{c} + \eta \mathbf{d}) \Big|_{\eta=0} \\
&= \frac{d}{d\eta} \int_\Omega \sum_{i=1}^{N} \left(\frac{1}{4\epsilon^2} (c_i + \eta d_i)^2 (1 - (c_i + \eta d_i))^2 \right. \\
&\quad \left. + \frac{1}{2} |\nabla (c_i + \eta d_i)|^2 \right) d\mathbf{x} \Big|_{\eta=0} \\
&= \int_\Omega \sum_{i=1}^{N} \left(\frac{1}{\epsilon^2} d_i f(c_i) + \nabla d_i \cdot \nabla c_i \right) d\mathbf{x} \\
&= \int_\Omega \left[\frac{1}{\epsilon^2} \left(\mathbf{f}(\mathbf{c}) \cdot \boldsymbol{\xi} - \mathbf{f}(\mathbf{c}) \cdot \frac{1}{N} \sum_{i=1}^{N} \xi_i \mathbf{1} \right) \right. \\
&\quad \left. + \sum_{i=1}^{N} \nabla \left(\xi_i - \frac{1}{N} \sum_{j=1}^{N} \xi_j \right) \cdot \nabla c_i \right] d\mathbf{x} \\
&= \int_\Omega \left[\frac{1}{\epsilon^2} \left(\mathbf{f}(\mathbf{c}) \cdot \boldsymbol{\xi} + \beta(\mathbf{c}) \mathbf{1} \cdot \boldsymbol{\xi} \right) \right. \\
&\quad \left. + \sum_{i=1}^{N} \left(\nabla \xi_i \cdot \nabla c_i - \frac{1}{N} \nabla \sum_{j=1}^{N} \xi_j \cdot \nabla c_i \right) \right] d\mathbf{x} \\
&= \int_\Omega \frac{1}{\epsilon^2} \left(\mathbf{f}(\mathbf{c}) + \beta(\mathbf{c}) \mathbf{1} \right) \cdot \boldsymbol{\xi} \, d\mathbf{x} \\
&\quad + \sum_{i=1}^{N} \left[\int_{\partial\Omega} \left(\xi_i \nabla c_i - \frac{1}{N} \sum_{j=1}^{N} \xi_j \nabla c_i \right) \cdot \mathbf{n} \, ds \right. \\
&\quad \left. - \int_\Omega \left(\xi_i \Delta c_i - \frac{1}{N} \sum_{j=1}^{N} \xi_j \Delta c_i \right) d\mathbf{x} \right] \\
&= \int_\Omega \frac{1}{\epsilon^2} \left(\mathbf{f}(\mathbf{c}) + \beta(\mathbf{c}) \mathbf{1} \right) \cdot \boldsymbol{\xi} \, d\mathbf{x} \\
&\quad - \int_\Omega \sum_{i=1}^{N} \left(\xi_i \Delta c_i - \frac{1}{N} \sum_{j=1}^{N} \xi_j \Delta c_i \right) d\mathbf{x} \\
&= \int_\Omega \left[\frac{1}{\epsilon^2} \left(\mathbf{f}(\mathbf{c}) + \beta(\mathbf{c}) \mathbf{1} \right) - \Delta \mathbf{c} \right] \cdot \boldsymbol{\xi} \, d\mathbf{x} \\
&= \int_\Omega \left[\frac{1}{\epsilon^2} \left(\mathbf{f}(\mathbf{c}) + \beta(\mathbf{c}) \mathbf{1} \right) - \Delta \mathbf{c} \right] \cdot \left(\mathbf{d} + \frac{1}{N} \sum_{i=1}^{N} \xi_i \mathbf{1} \right) d\mathbf{x} \\
&= \int_\Omega \left[\frac{1}{\epsilon^2} \left(\mathbf{f}(\mathbf{c}) + \beta(\mathbf{c}) \mathbf{1} \right) - \Delta \mathbf{c} \right] \cdot \mathbf{d} \, d\mathbf{x} \\
&\quad + \left(\frac{1}{N} \sum_{i=1}^{N} \xi_i \right) \int_\Omega \left[\frac{1}{\epsilon^2} \sum_{i=1}^{N} \left(f(c_i) + \beta(\mathbf{c}) \right) - \sum_{i=1}^{N} \Delta c_i \right] d\mathbf{x} \\
&= \int_\Omega \left[\frac{1}{\epsilon^2} \left(\mathbf{f}(\mathbf{c}) + \beta(\mathbf{c}) \mathbf{1} \right) - \Delta \mathbf{c} \right] \cdot \mathbf{d} \, d\mathbf{x} \\
&= \left(\frac{1}{\epsilon^2} \left(\mathbf{f}(\mathbf{c}) + \beta(\mathbf{c}) \mathbf{1} \right) - \Delta \mathbf{c}, \mathbf{d} \right)_{L^2},
\end{aligned}$$

where we have used the boundary condition (2). We identify $\text{grad}\mathcal{F}(\mathbf{c}) = (\mathbf{f}(\mathbf{c}) + \beta(\mathbf{c})\mathbf{1})/\epsilon^2 - \Delta\mathbf{c}$, then Eq. (3) becomes the vector-valued AC equations

$$\frac{\partial \mathbf{c}}{\partial t} = -\left(\frac{\mathbf{f}(\mathbf{c}) + \beta(\mathbf{c})\mathbf{1}}{\epsilon^2} - \Delta\mathbf{c}\right). \tag{4}$$

3. Numerical algorithm

In this section, we describe an operator splitting algorithm for solving the vector-valued AC equations. For simplicity and clarity of exposition, we shall discretize the vector-valued AC equations in one-dimensional space, i.e., $\Omega = (a, b)$. Two- and three-dimensional discretizations are defined analogously. Note that we only need to solve equations with $c_1, c_2, \ldots, c_{N-1}$, since $c_N = 1 - c_1 - c_2 - \cdots - c_{N-1}$. Let $\mathbf{c} = (c_1, c_2, \ldots, c_{N-1})$.

Let N_x be a positive even integer, $h = (b-a)/N_x$ be a uniform grid size, and $\Omega_h = \{x_i = (i - 0.5)h, 1 \le i \le N_x\}$ be the set of cell-centers. Let $\mathbf{c}_i^n$ be the approximations of $\mathbf{c}(x_i, n\Delta t)$, where $\Delta t = T/N_t$ is the time step, T is the final time, and N_t is the total number of time steps. The zero Neumann boundary condition (2) is first implemented by requiring that for each n,

$$\nabla_h \mathbf{c}^n_{\frac{1}{2}} = \nabla_h \mathbf{c}^n_{N_x+\frac{1}{2}} = \mathbf{0},$$

where the discrete differentiation operator is $\nabla_h \mathbf{c}^n_{i+\frac{1}{2}} = (\mathbf{c}^n_{i+1} - \mathbf{c}^n_i)/h$. We then define the discrete Laplacian as $\Delta_h \mathbf{c}^n_i = (\nabla_h \mathbf{c}^n_{i+\frac{1}{2}} - \nabla_h \mathbf{c}^n_{i-\frac{1}{2}})/h$ and the discrete L^2 inner product as

$$(\mathbf{c}, \mathbf{d})_h = h \sum_{k=1}^{N-1} \sum_{i=1}^{N_x} c_{ki} d_{ki}. \tag{5}$$

We also define a discrete norm associated with (5) as $\|\mathbf{c}\|^2 = (\mathbf{c}, \mathbf{c})_h$. We redefine $\mathbf{f}(\mathbf{c})$ and $\mathbf{1}$ to $\mathbf{f}(\mathbf{c}) = (f(c_1), f(c_2), \ldots, f(c_{N-1}))$ and $\mathbf{1} = (1, 1, \ldots, 1) \in \mathbb{R}^{N-1}$. We discretize Eq. (4) in time by an operator splitting algorithm:

$$\frac{\mathbf{c}_i^* - \mathbf{c}_i^n}{\Delta t} = \Delta_h \mathbf{c}_i^* - \frac{\beta(\mathbf{c}_i^n)\mathbf{1}}{\epsilon^2}, \tag{6}$$

$$\frac{\mathbf{c}_i^{n+1} - \mathbf{c}_i^*}{\Delta t} = -\frac{\mathbf{f}(\mathbf{c}_i^{n+1})}{\epsilon^2}. \tag{7}$$

Eq. (6) is an implicit Euler's method for $\mathbf{c}_t = \Delta\mathbf{c} - \frac{\beta(\mathbf{c})\mathbf{1}}{\epsilon^2}$ with an initial condition $\mathbf{c}^n$. The resulting implicit discrete system of equations can be solved by a fast solver, such as the linear geometric multigrid method [57,58]. Also, a pointwise Gauss–Seidel relaxation scheme is used as the smoother in the multigrid method. Eq. (7) can be considered as an approximation of the equation

$$\mathbf{c}_t = -\frac{\mathbf{f}(\mathbf{c})}{\epsilon^2} \tag{8}$$

by an implicit Euler's method with an initial condition $\mathbf{c}^*$. We can solve Eq. (8) analytically by the method of separation of variables [59]. The solution is given as follows:

$$c_{k_i}^{n+1} = 0.5 + \frac{c_{k_i}^* - 0.5}{\sqrt{e^{\frac{-\Delta t}{2\epsilon^2}} + (2c_{k_i}^* - 1)^2 (1 - e^{\frac{-\Delta t}{2\epsilon^2}})}}$$

for $k = 1, 2, \ldots, N-1$.

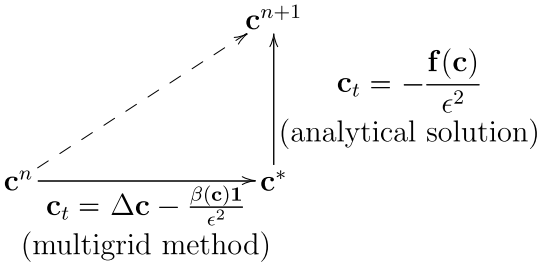


Fig. 1. Computational algorithm for solving the vector-valued AC equations.

The numerical algorithm is shown schematically in Fig. 1.

In Eq. (6), the variable Lagrange multiplier $\beta(\mathbf{c})$ is determined by the solutions at time level n. By treating $\beta(\mathbf{c})$ explicitly, there is no relation between the solutions at time level $*$. Thus the vector-valued AC equations can be solved in a decoupled way, i.e.,

$$\frac{c_{k_i}^* - c_{k_i}^n}{\Delta t} = \Delta_h c_{k_i}^* - \frac{\beta(\mathbf{c}_i^n)}{\epsilon^2},$$

$$c_{k_i}^{n+1} = 0.5 + \frac{c_{k_i}^* - 0.5}{\sqrt{e^{\frac{-\Delta t}{2\epsilon^2}} + (2c_{k_i}^* - 1)^2 \left(1 - e^{\frac{-\Delta t}{2\epsilon^2}}\right)}}$$

for $k = 1, 2, \ldots, N-1$.

This means that we only solve the binary AC equation $N-1$ times to solve the vector-valued AC equations. Our algorithm is accurate to second order in space and first order in time (this will be confirmed numerically in Section 5.1). The accuracy in time can be improved by using high-order multistep methods [60–64].

4. The growth of multiple dendrites with k-fold symmetry

One of applications of the vector-valued AC equations is the growth of multiple dendrites. Extending the algorithm presented in Section 3, we can simulate the growth of multiple dendrites with different orientation angles and fold numbers on a single domain. In this section, the phase-field model for the growth of multiple dendrites is summarized and the numerical solution is presented.

4.1. The phase-field model

A two-dimensional phase-field model for the growth of multiple k-fold symmetric dendrites is as follows:

$$\begin{aligned}\epsilon^2(c_i)\frac{\partial c_i}{\partial t} &= \nabla \cdot (\epsilon^2(c_i)\nabla c_i) \\ &\quad + [c_i - 0.5 - \lambda U c_i(1 - c_i)]c_i(1 - c_i) \\ &\quad + \left(|\nabla c_i|^2 \epsilon(c_i)\frac{\partial \epsilon(c_i)}{\partial c_{ix}}\right)_x + \left(|\nabla c_i|^2 \epsilon(c_i)\frac{\partial \epsilon(c_i)}{\partial c_{iy}}\right)_y,\end{aligned} \tag{9}$$

$$\frac{\partial U}{\partial t} = D\Delta U + \sum_{i=1}^{N} \frac{\partial c_i}{\partial t} \quad \text{for } i = 1, 2, \ldots, N,$$

where c_i is the phase-field of the ith dendrite, which varies from unity in the solid phase to zero in the liquid phase, $\epsilon(c_i)$ is an anisotropy function, λ is a dimensionless parameter that controls the strength of the coupling between the phase and diffusion fields, and $U = (T - T_M)/(L/c_p)$ is a dimensionless temperature field. Here T is the temperature, T_M is the melting temperature, L is the latent heat of melting, and c_p is the specific heat at constant pressure. $D = \alpha\tau_0/W_0^2$ is the dimensionless thermal diffusivity, α is the thermal diffusivity of the solid, τ_0 is the characteristic time, and W_0 is the characteristic length, which is typically the diffuse interface width. The relations $W_0 = \lambda d_0/a_1$ and $\tau_0 = (d_0^2/\alpha)a_2\lambda^3/a_1^2$ follow

from the thin-interface analysis of Karma and Rappel [65], where d_0 is the capillary length and a_1 and a_2 are constants.

For each dendrite, the angle between the direction normal to the interface and the x-axis is calculated from the phase-field via $\theta_i = \arctan(c_{iy}/c_{ix})$. Then, by replacing $\epsilon(c_i)$ with $\epsilon(\theta_i) = 1 + \epsilon_k \cos(k\theta_i)$, where ϵ_k is the strength of anisotropy and k is a fold number, we can simplify the third term on the right-hand side of Eq. (9):

$$\begin{aligned}&\left(|\nabla c_i|^2\epsilon(c_i)\frac{\partial\epsilon(c_i)}{\partial c_{ix}}\right)_x\\&= \left((c_{ix}^2 + c_{iy}^2)\epsilon(\theta_i)\epsilon'(\theta_i)\left(-\frac{c_{iy}}{c_{ix}^2 + c_{iy}^2}\right)\right)_x\\&= -(\epsilon'(\theta_i)\epsilon(\theta_i)c_{iy})_x.\end{aligned}$$

In a similar way, we get

$$\left(|\nabla c_i|^2\epsilon(c_i)\frac{\partial\epsilon(c_i)}{\partial c_{iy}}\right)_y = (\epsilon'(\theta_i)\epsilon(\theta_i)c_{ix})_y.$$

Therefore the phase-field equations for multiple k-fold symmetric dendrites can be rewritten as

$$\begin{aligned}\epsilon^2(\theta_i)\frac{\partial c_i}{\partial t} &= \nabla\cdot(\epsilon^2(\theta_i)\nabla c_i)\\&\quad + [c_i - 0.5 - \lambda U c_i(1-c_i)]c_i(1-c_i)\\&\quad - (\epsilon'(\theta_i)\epsilon(\theta_i)c_{iy})_x + (\epsilon'(\theta_i)\epsilon(\theta_i)c_{ix})_y,\end{aligned} \tag{10}$$

$$\frac{\partial U}{\partial t} = D\Delta U + \sum_{i=1}^{N}\frac{\partial c_i}{\partial t} \quad \text{for } i = 1, 2, \ldots, N. \tag{11}$$

4.2. Numerical algorithm

We discretize Eqs. (10) and (11) in time by an operator splitting algorithm:

$$\begin{aligned}&\epsilon^2(\theta_i^n)\frac{c_i^{n+1} - c_i^n}{\Delta t}\\&= \epsilon^2(\theta_i^n)\Delta_h c_i^{n+1,2}\\&\quad + 2\epsilon(\theta_i^n)\nabla_h\epsilon(\theta_i^n)\cdot\nabla_h c_i^n\\&\quad - F'(c_i^{n+1}) - 4\lambda U^n F(c_i^{n+1,1})\\&\quad - (\epsilon'(\theta_i)\epsilon(\theta_i)c_{iy})_x^n + (\epsilon'(\theta_i)\epsilon(\theta_i)c_{ix})_y^n,\end{aligned} \tag{12}$$

$$\frac{U^{n+1} - U^n}{\Delta t} = D\Delta_h U^{n+1} + \sum_{i=1}^{N}\frac{c_i^{n+1} - c_i^n}{\Delta t}$$

for $i = 1, 2, \ldots, N$,

where $F(c) = 0.25c^2(1-c)^2$ and $F'(c) = c(c-0.5)(c-1)$. Here $c_i^{n+1,1}$ and $c_i^{n+1,2}$ are defined in the operator splitting algorithm. Eq. (12) can be split into three equations:

$$\begin{aligned}&\epsilon^2(\theta_i^n)\frac{c_i^{n+1,1} - c_i^n}{\Delta t}\\&= 2\epsilon(\theta_i^n)\nabla_h\epsilon(\theta_i^n)\cdot\nabla_h c_i^n\\&\quad - (\epsilon'(\theta_i)\epsilon(\theta_i)c_{iy})_x^n + (\epsilon'(\theta_i)\epsilon(\theta_i)c_{ix})_y^n,\\&\epsilon^2(\theta_i^n)\frac{c_i^{n+1,2} - c_i^{n+1,1}}{\Delta t} = \epsilon^2(\theta_i^n)\Delta_h c_i^{n+1,2}\\&\qquad - 4\lambda U^n F(c_i^{n+1,1}),\\&\epsilon^2(\theta_i^n)\frac{c_i^{n+1} - c_i^{n+1,2}}{\Delta t} = -F'(c_i^{n+1}).\end{aligned} \tag{13}$$

Eq. (13) can be considered as an approximation of the equation

$$c_t = -\frac{c(c-0.5)(c-1)}{\epsilon^2} \tag{14}$$

by an implicit Euler's method with an initial condition $c_i^{n+1,2}$. We can solve Eq. (14) analytically by the method of separation of variables. The solution is given as follows:

$$c_i^{n+1} = 0.5 + \frac{c_i^{n+1,2} - 0.5}{\sqrt{e^{-\frac{\Delta t}{2\epsilon^2(\theta_i^n)}} + (2c_i^{n+1,2} - 1)^2\left(1 - e^{-\frac{\Delta t}{2\epsilon^2(\theta_i^n)}}\right)}}.$$

Finally, the numerical algorithm can be written as follows: for $i = 1, 2, \ldots, N$

$$\begin{aligned}&\epsilon^2(\theta_i^n)\frac{c_i^{n+1,1} - c_i^n}{\Delta t}\\&= 2\epsilon(\theta_i^n)\nabla_h\epsilon(\theta_i^n)\cdot\nabla_h c_i^n\\&\quad - (\epsilon'(\theta_i)\epsilon(\theta_i)c_{iy})_x^n + (\epsilon'(\theta_i)\epsilon(\theta_i)c_{ix})_y^n,\\&\epsilon^2(\theta_i^n)\frac{c_i^{n+1,2} - c_i^{n+1,1}}{\Delta t} = \epsilon^2(\theta_i^n)\Delta_h c_i^{n+1,2}\\&\qquad - 4\lambda U^n F(c_i^{n+1,1}),\\&c_i^{n+1} = 0.5 + \frac{c_i^{n+1,2} - 0.5}{\sqrt{e^{-\frac{\Delta t}{2\epsilon^2(\theta_i^n)}} + (2c_i^{n+1,2} - 1)^2\left(1 - e^{-\frac{\Delta t}{2\epsilon^2(\theta_i^n)}}\right)}},\end{aligned} \tag{15}$$

$$\frac{U^{n+1} - U^n}{\Delta t} = D\Delta_h U^{n+1} + \sum_{i=1}^{N}\frac{c_i^{n+1} - c_i^n}{\Delta t}. \tag{16}$$

Eqs. (15) and (16) can be solved by a linear geometric multigrid method. Also, a pointwise Gauss–Seidel relaxation scheme is used as the smoother in the multigrid method.

5. Numerical experiments

5.1. Convergence test

We consider a quaternary system in one-dimensional space, $\Omega = [0, 1]$. An estimate of the convergence rate is obtained by performing a number of simulations for a sample initial problem on a set of increasingly finer grids. The initial conditions are

$$\begin{aligned}c_1(x, 0) &= 0.25 + 0.01\cos(3\pi x) + 0.04\cos(5\pi x),\\c_2(x, 0) &= 0.25 - 0.02\cos(2\pi x) + 0.01\cos(4\pi x),\\c_3(x, 0) &= 0.25 + 0.03\cos(5\pi x) + 0.015\cos(3\pi x).\end{aligned}$$

The numerical solutions are computed on the uniform grids $h = 1/2^n$ and with $\epsilon = 0.01$ for $n = 5, 6, 7, 8$, and 9. For each grid, we integrate to time $T = 0.001$ with $\Delta t = 0.001h^2$. We define the error to be the discrete l_2-norm of the difference between that grid and the average of the next finer grid cells covering it: $\mathbf{e}_{h/\frac{h}{2}_i} := \mathbf{c}_{h_i} - (\mathbf{c}_{\frac{h}{2}_{2i}} + \mathbf{c}_{\frac{h}{2}_{2i-1}})/2$. The rate of convergence is defined as: $\log_2(\|\mathbf{e}_{h/\frac{h}{2}}\|/\|\mathbf{e}_{\frac{h}{2}/\frac{h}{4}}\|)$. The errors and rates of convergence are given in Table 1. The results suggest that the scheme is accurate to second order in space and first order in time.

5.2. Linear stability analysis

In this section, we study the short-time behavior of a quaternary mixture. The partial differential equation (4) we wish to solve can be written as

$$\frac{\partial\mathbf{c}(x, t)}{\partial t} = -\left(\frac{\psi(\mathbf{c})}{\epsilon^2} - \Delta\mathbf{c}\right), \quad \text{for } (x, t) \in \Omega\times(0, T], \tag{17}$$

Table 1
l_2 convergence result.

32–64	Rate	64–128	Rate	128–256	Rate	256–512
5.6048e−4	1.9997	1.4015e−4	2.0000	3.5037e−5	2.0000	8.7594e−6

where $\psi(\mathbf{c}) = \mathbf{f}(\mathbf{c}) + \beta(\mathbf{c})\mathbf{1}$. Let the mean concentration take the form $\mathbf{m} = (m_1, m_2, m_3)$. We seek a solution of the form

$$\mathbf{c}(x, t) = \mathbf{m} + \sum_{k=1}^{\infty} \cos(k\pi x)(\alpha_k(t), \beta_k(t), \gamma_k(t)), \tag{18}$$

where $|\alpha_k(t)|, |\beta_k(t)|$, and $|\gamma_k(t)| \ll 1$. After linearizing $\psi(\mathbf{c})$ about $\mathbf{m}$, we have

$$\psi(\mathbf{c}) \approx \psi(\mathbf{m}) + (\mathbf{c} - \mathbf{m}) \begin{pmatrix} \partial_{c_1}\psi_1(\mathbf{m}) & \partial_{c_1}\psi_2(\mathbf{m}) & \partial_{c_1}\psi_3(\mathbf{m}) \\ \partial_{c_2}\psi_1(\mathbf{m}) & \partial_{c_2}\psi_2(\mathbf{m}) & \partial_{c_2}\psi_3(\mathbf{m}) \\ \partial_{c_3}\psi_1(\mathbf{m}) & \partial_{c_3}\psi_2(\mathbf{m}) & \partial_{c_3}\psi_3(\mathbf{m}) \end{pmatrix}. \tag{19}$$

Substituting (19) into (17) and letting $m_1 = m_2 = m_3 = m = 0.25$ for simplicity, then, up to first order, we have

$$\frac{\partial \mathbf{c}}{\partial t} = -\frac{\mathbf{c} - \mathbf{m}}{2\epsilon^2} \times \begin{pmatrix} 18m^2 - 9m + 1 & 3m(4m-1) & 3m(4m-1) \\ 3m(4m-1) & 18m^2 - 9m + 1 & 3m(4m-1) \\ 3m(4m-1) & 3m(4m-1) & 18m^2 - 9m + 1 \end{pmatrix} + \Delta \mathbf{c}. \tag{20}$$

After substituting $\mathbf{c}(x, t)$ from Eq. (18) into (20), we get

$$\frac{d}{dt}\begin{pmatrix} \alpha_k(t) \\ \beta_k(t) \\ \gamma_k(t) \end{pmatrix} = \mathbf{A}\begin{pmatrix} \alpha_k(t) \\ \beta_k(t) \\ \gamma_k(t) \end{pmatrix}, \quad \mathbf{A} = \begin{pmatrix} a & b & b \\ b & a & b \\ b & b & a \end{pmatrix}, \tag{21}$$

where $a = -(18m^2 - 9m + 1)/(2\epsilon^2) - k^2\pi^2$ and $b = -3m(4m - 1)/(2\epsilon^2)$. The eigenvalues of $\mathbf{A}$ are

$$\lambda_1 = -\frac{42m^2 - 15m + 1 + 2\epsilon^2 k^2 \pi^2}{2\epsilon^2},$$
$$\lambda_2 = \lambda_3 = -\frac{6m^2 - 6m + 1 + 2\epsilon^2 k^2 \pi^2}{2\epsilon^2}.$$

The solution to the system of ordinary differential equations (21) is given by

$$\begin{pmatrix} \alpha_k(t) \\ \beta_k(t) \\ \gamma_k(t) \end{pmatrix} = \frac{\alpha_k(0) + \beta_k(0) + \gamma_k(0)}{3}\begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix} e^{\lambda_1 t} + \frac{-\alpha_k(0) - \beta_k(0) + 2\gamma_k(0)}{3}\begin{pmatrix} -1 \\ 0 \\ 1 \end{pmatrix} e^{\lambda_2 t} + \frac{-\alpha_k(0) + 2\beta_k(0) - \gamma_k(0)}{3}\begin{pmatrix} -1 \\ 1 \\ 0 \end{pmatrix} e^{\lambda_2 t}.$$

In Fig. 2, we plot the evolution of the amplitudes as a function of time. The symbols '-○-', '-◇-', and '-△-' are numerical results, which are compared with the theoretical values $\alpha_k(t)$ (point), $\beta_k(t)$ (star), and $\gamma_k(t)$ (plus), respectively:

$$c_1(x, 0) = 0.25 + 0.001\cos(3\pi x), \tag{22}$$
$$c_2(x, 0) = 0.25 + 0.002\cos(3\pi x), \tag{23}$$
$$c_3(x, 0) = 0.25 + 0.003\cos(3\pi x). \tag{24}$$

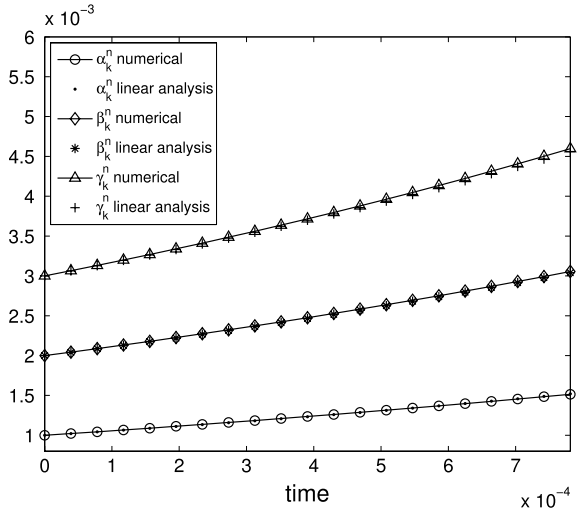


Fig. 2. The symbols '-○-', '-◇-', and '-△-' are numerical results, which are compared with the theoretical values $\alpha_k(t)$ (point), $\beta_k(t)$ (star), and $\gamma_k(t)$ (plus), respectively, with the initial conditions of Eqs. (22)–(24).

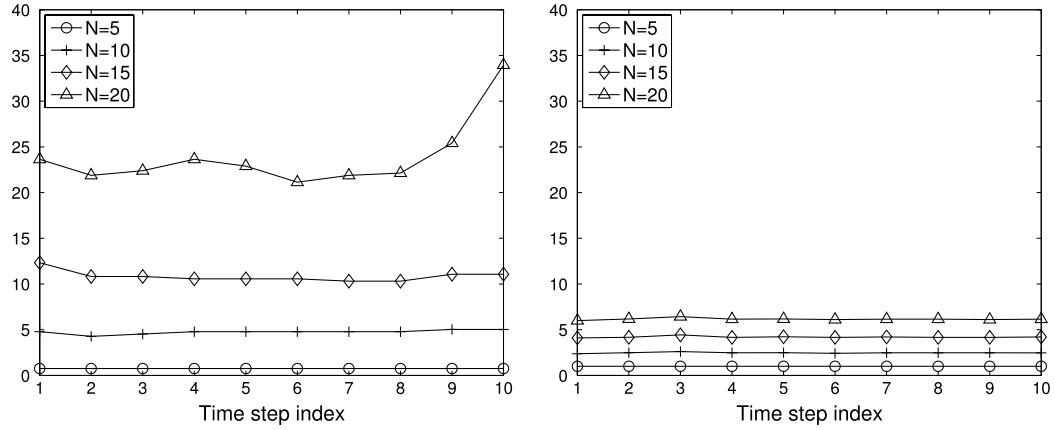
Here, we used $k = 3$, $m = 0.25$, $\epsilon = 0.01$, $h = 1/256$, $\Delta t = 0.001h$, and $T = 200\Delta t$. The numerical amplitudes are defined by

$$\alpha_k^n = \left(\max_{1\le i\le N_x} c_1^n(x_i) - \min_{1\le i\le N_x} c_1^n(x_i)\right)\Big/2,$$
$$\beta_k^n = \left(\max_{1\le i\le N_x} c_2^n(x_i) - \min_{1\le i\le N_x} c_2^n(x_i)\right)\Big/2,$$
$$\gamma_k^n = \left(\max_{1\le i\le N_x} c_3^n(x_i) - \min_{1\le i\le N_x} c_3^n(x_i)\right)\Big/2.$$

The results in Fig. 2 show that the linear stability analysis and numerical solutions are in good agreement in a linear regime.

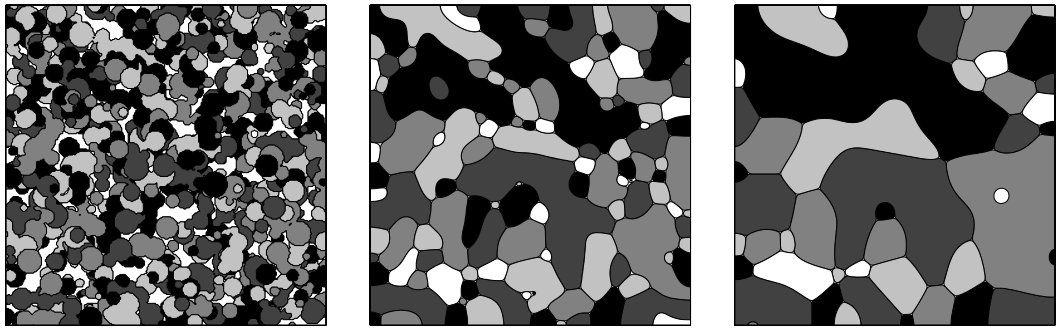
5.3. The efficiency of the proposed algorithm

As mentioned in Section 3, we can solve the vector-valued AC equations with N components in a decoupled way by using our algorithm. In order to show the efficiency of the proposed algorithm, we consider grain growth on the unit square domain $\Omega = (0, 1)\times(0, 1)$ with $N = 5, 10, 15$, and 20. The initial condition is a randomly chosen superposition of 1000 circular grains. The randomly chosen radii are ranging from 0.01 to 0.04. The other parameter values are chosen as $h = 1/256$, $\epsilon = 0.0019$, and $\Delta t = 0.1\epsilon^2$. Simulations are performed on a 3.2 GHz Intel Core i3 CPU with 2 GB of RAM. Table 2 provides the execution time for different values of N, averaged over the first 10 time steps. Fig. 3 (a) and (b) show the relative computing time per time step for the first 10 time steps of Vanherpe et al. [32] and our study, respectively. Here, the execution time for $N = 5$ is taken as a reference point. It can be seen that the convergence rate of the average CPU time of the proposed algorithm is linear with respect to a number of phase-field variables. For $N = 5$, Fig. 4 shows the initial condition and evolutions of the phases.



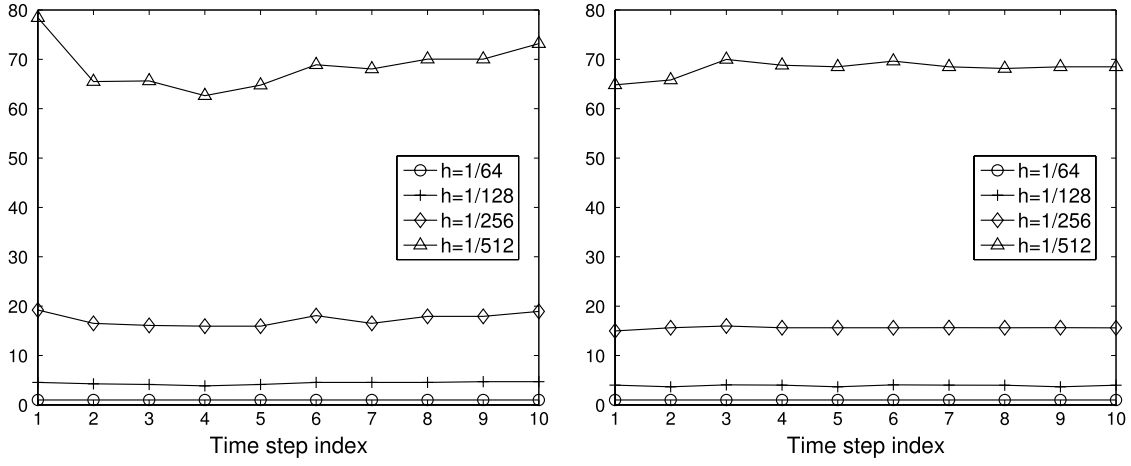
(a) Vanherpe et al.'s result [32]. (b) Our result.

Fig. 3. CPU time during the first 10 time steps for $N = 5$, 10, 15, and 20. Here, the CPU time for $N = 5$ is taken as a reference point.



(a) Initial condition. (b) $t = 3000\Delta t$. (c) $t = 10000\Delta t$.

Fig. 4. Evolution of the phases for $N = 5$.



(a) Vanherpe et al.'s result [32]. (b) Our result.

Fig. 5. CPU time during the first 10 time steps for $h = 1/64$, $1/128$, $1/256$, and $1/512$. Here, the CPU time for $h = 1/64$ is taken as a reference point.

Table 2
Average CPU time (s) for different numbers of phase-field variables N during the first 10 time steps. $V(5, 5)$-cycles are applied. The numbers in parentheses are the average CPU times (s) given in Ref. [32].

N	System size	Average CPU time
5	$5 \times 256 \times 256$	0.2953 (8.0620)
10	$10 \times 256 \times 256$	0.7297 (38.5610)
15	$15 \times 256 \times 256$	1.2391 (89.0100)
20	$20 \times 256 \times 256$	1.8188 (193.3280)

Next, we study the execution time as a function of the grid size, h. Table 3 shows the execution time averaged over the first 10 time steps of simulations on a domain $\Omega = (0, 1) \times (0, 1)$ with $h = 1/64$, $1/128$, $1/256$, and $1/512$, for $\epsilon = 0.0019$, $\Delta t = 0.1\epsilon^2$, and $N = 10$. Fig. 5(a) and (b) show the relative computing time per time step for the first 10 time steps of Vanherpe et al. [32] and our study, respectively. Here, the execution time for $h = 1/64$ is taken as a reference point. The results suggest that both Vanherpe et al.'s and our algorithms achieve multigrid efficiency and, in particular, our algorithm reaches the tenth time step in a very small CPU time.

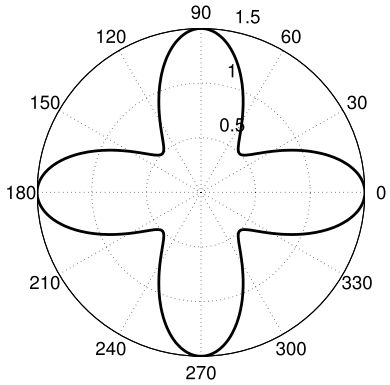


Fig. 6. Polar plot of the anisotropy function $\epsilon(\theta) = 1 + \epsilon_4 \cos(4\theta)$, with $\epsilon_4 = 0.5$.

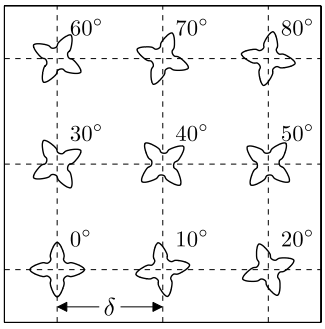


Fig. 7. Schematic of the growth of nine crystals.

Table 3
Average CPU time (s) for different numbers of spatial unknowns during the first 10 time steps. $V(5, 5)$-cycles are applied. The numbers in parentheses are the average CPU times (s) given in Ref. [32].

h	System size	Average CPU time
1/64	$10 \times 64 \times 64$	0.0468 (2.2250)
1/128	$10 \times 128 \times 128$	0.1828 (9.5110)
1/256	$10 \times 256 \times 256$	0.7297 (38.5610)
1/512	$10 \times 512 \times 512$	3.1875 (153.3250)

5.4. The growth of nine crystals: effect of crystal spacing

With the anisotropy function $\epsilon(\theta) = 1 + \epsilon_4 \cos[4(\theta - I)]$, where I is an orientation angle, a four-fold crystal grows fastest in directions with angles $0° + I$, $90° + I$, $180° + I$, and $270° + I$ to the x-axis, and slowest in directions with angles $45° + I$, $135° + I$, $225° + I$, and $315° + I$ to the x-axis [66] (see Fig. 6). As I varies for different crystals, different crystals will grow with different preferred orientations.

In this section, we consider the growth of nine four-fold crystals with orientations $I = 0°, 10°, 20°, 30°, 40°, 50°, 60°, 70°$, and $80°$, as shown in Fig. 7. A simulation of this problem has previously been performed by Tan and Zabaras using a level-set method [66]. The computational domain is $\Omega = (-450, 450) \times (-450, 450)$. The nine crystals are uniformly spaced with distance δ. The initial shape of each crystal is circular, with radius $14d_0$. The capillary length d_0 is defined as $d_0 = a_1 W_0/\lambda$ [67–69], with $a_1 = 5\sqrt{2}/8$ [65,67, 70], $W_0 = 1$ [67–69], and $\lambda = 3.1913$ [67]. Initially, the domain is undercooled at temperature -0.55, while inside the nine initial crystals the initial temperature is taken as 0. We take $\epsilon_4 = 0.05$, $h = 0.8789$, and $\Delta t = 0.15$.

Fig. 8(a) and (b) show evolutions of the interface with spacing 300 and 200, respectively. In the case of $\delta = 300$, each crystal grows almost independently, since the thermal boundary layers very slightly overlap (see Figs. 8 and 9(a)). However, in the case of $\delta = 200$, the interaction between the crystals is very obvious, as shown in Fig. 8(b). This is caused by overlapping of the thermal boundary layers (see Fig. 9 (b)). These results are in qualitative agreement with the previous results of Tan and Zabaras [66]. Also, in both cases, we can see that the two crystals with $I = 30°$ and $I = 60°$ (or $I = 40°$ and $I = 50°$) are symmetrical to each other at places close to their center line $y = \frac{\delta}{2}$ (or $x = \frac{\delta}{2}$). This partial symmetry comes from the symmetry in the crystal orientations, as shown in Fig. 7.

5.5. Crystals with different fold numbers and orientations on a single domain

With the anisotropy functions $\epsilon(\theta_i) = 1 + \epsilon_k \cos[k(\theta_i - I_i)]$ for each crystal, our algorithm can simulate the growth of multiple crystals with different orientation angles and fold numbers on a single domain. To show this, we simulate the growth of eight crystals with different fold numbers and orientations on a single domain. A 1024×1024 mesh is used on the computational domain $\Omega = (-450, 450) \times (-450, 450)$. The initial shape of each crystal is circular, with radius $14d_0$ where $d_0 = 0.2770$. Initially, the domain is undercooled at temperature -0.55, while inside the nine initial crystals the initial temperature is taken as 0. We choose $\Delta t = 0.15$ and $\epsilon_k = 1/(k^2 - 1)$. The evolutions are shown in Fig. 10.

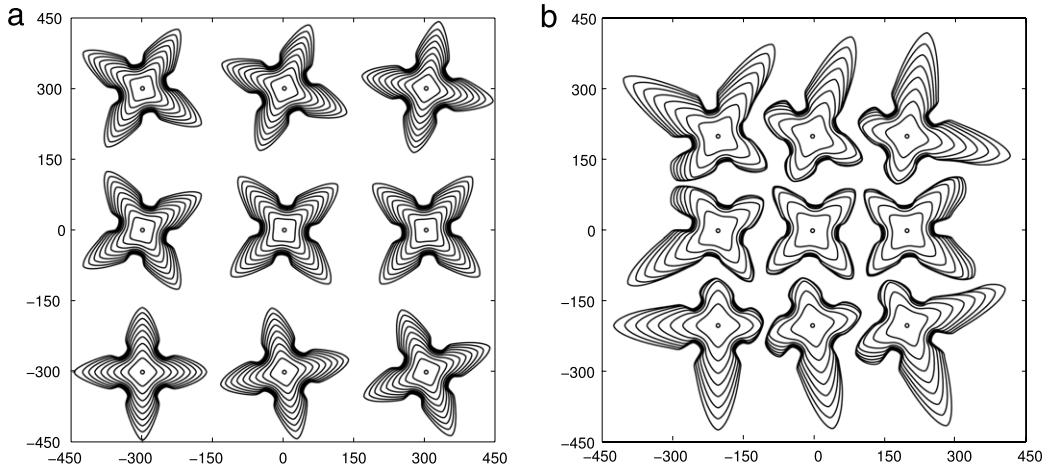
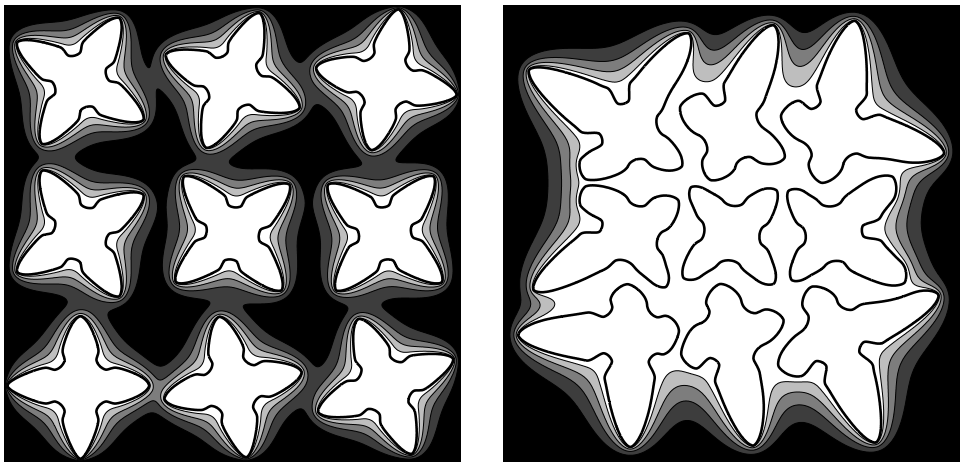


Fig. 8. (a) and (b) show sequences of interfaces with spacing 300 and 200, respectively. The times are (a) $t = 0$, 113, 225, 338, 450, 563, 675, 788, 900, and 1013 and (b) $t = 0$, 225, 450, 675, 900, 1125, 1350, 1575, and 1800 (from inside to outside).



(a) $t = 1013$. (b) $t = 1800$.

Fig. 9. Temperature field for the interaction between nine crystals with spacing (a) 300 and (b) 200. The temperature fields are shown with filled contours from −0.55 to 0 increasing in steps of 0.11.

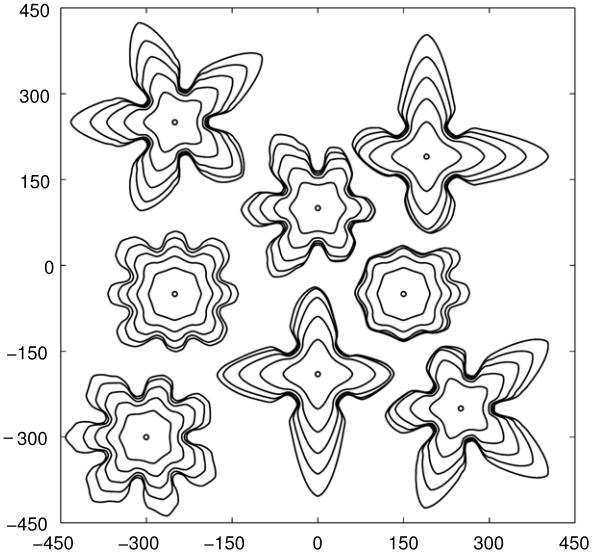


Fig. 10. The growth of eight crystals with different fold numbers and orientations.

6. Conclusions

In this paper, we considered the vector-valued Allen–Cahn equations which model phase separation in N-component systems. The numerical algorithm for the vector-valued AC equations was based on an operator splitting technique. The linear equation was solved using an implicit Euler's method and a linear geometric multigrid method with Gauss–Seidel smoother, and the nonlinear equation was solved analytically. The accuracy of our algorithm was demonstrated using a convergence test and it was shown that our algorithm is accurate to second order in space and first order in time. The algorithm allowed us to convert the vector-valued AC equations with N components into a system of $N - 1$ binary AC equations and drastically reduced the required computational time and memory. Through the efficiency test, we observed that the convergence rate of the computing time of the algorithm is linear with respect to the number of phase-field variables. Finally, by extending our algorithm, we simulated the growth of multiple crystals with different orientation angles and fold numbers on a single domain.

Note that to compute the vector-valued Allen–Cahn equations in arbitrarily shaped domains, we required the use of unstructured meshes of triangles. For unstructured meshes, an algebraic multigrid method is more suitable than a geometric multigrid method. In future work, we will investigate our algorithm with an algebraic multigrid to allow more flexibility in the choice of mesh.

Acknowledgments

This research was supported by the Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education, Science and Technology (No. 2011-0023794). The authors thank the reviewers for their constructive and valuable comments.

References

[1] S.M. Allen, J.W. Cahn, Acta Metall. 27 (1979) 1085.
[2] M. Beneš, V. Chalupecký, K. Mikula, Appl. Numer. Math. 51 (2004) 187.
[3] J.A. Dobrosotskaya, A.L. Bertozzi, IEEE Trans. Image Process. 17 (2008) 657.
[4] L.C. Evans, H.M. Soner, P.E. Souganidis, Comm. Pure Appl. Math. 45 (1992) 1097.
[5] T. Ilmanen, J. Differ. Geom. 38 (1993) 417.
[6] M. Katsoulakis, G.T. Kossioris, F. Reitich, J. Geom. Anal. 5 (1995) 255.
[7] L.Q. Chen, J. Shen, Comput. Phys. Commun. 108 (1998) 147.
[8] M. Beneš, K. Mikula, Acta Math. Univ. Comenian. 67 (1998) 17.
[9] X. Feng, A. Prohl, Numer. Math. 94 (2003) 33.
[10] T. Ohtsuka, Asymptot. Anal. 56 (2008) 87.
[11] X. Yang, J.J. Feng, C. Liu, J. Shen, J. Comput. Phys. 218 (2006) 417.
[12] A.A. Wheeler, W.J. Boettinger, G.B. McFadden, Phys. Rev. A 45 (1992) 7424.
[13] M. Cheng, J.A. Warren, J. Comput. Phys. 227 (2008) 6241.
[14] Y. Li, H.G. Lee, J. Kim, J. Cryst. Growth 321 (2011) 176.
[15] L.-Q. Chen, W. Yang, Phys. Rev. B 50 (1994) 15752.
[16] I. Steinbach, F. Pezzolla, B. Nestler, M. Seeßelberg, R. Prieler, G.J. Schmitz, J.L.L. Rezende, Physica D 94 (1996) 135.
[17] D. Fan, C. Geng, L.-Q. Chen, Acta Mater. 45 (1997) 1115.
[18] M.T. Lusk, Proc. R. Soc. Lond. Ser. A 455 (1999) 677.
[19] R. Kobayashi, J.A. Warren, W.C. Carter, Physica D 140 (2000) 141.
[20] J.A. Sethian, J. Strain, J. Comput. Phys. 98 (1992) 231.
[21] S. Li, J.S. Lowengrub, P.H. Leo, Physica D 208 (2005) 209.
[22] D. Li, R. Li, P. Zhang, Appl. Math. Model. 31 (2007) 971.
[23] H. Yin, S.D. Felicelli, Modell. Simul. Mater. Sci. Eng. 17 (2009) 075011.
[24] D. Juric, G. Tryggvason, J. Comput. Phys. 123 (1996) 127.
[25] D. Stafford, M.J. Ward, B. Wetton, Eur. J. Appl. Math. 12 (2001) 1.
[26] S. Chen, B. Merriman, S. Osher, P. Smereka, J. Comput. Phys. 135 (1997) 8.
[27] L.-L. Wang, Y. Gu, Commun. Comput. Phys. 9 (2011) 859.
[28] Z. Xu, H. Huang, X. Li, P. Meakin, Comput. Phys. Commun. 183 (2012) 15.
[29] A.E. Lobkovsky, J.A. Warren, J. Cryst. Growth 225 (2001) 282.
[30] B. Nestler, A.A. Wheeler, Comput. Phys. Commun. 147 (2002) 230.
[31] J.-W. Choi, H.G. Lee, D. Jeong, J. Kim, Phys. A 388 (2009) 1791.
[32] L. Vanherpe, F. Wendler, B. Nestler, S. Vandewalle, Math. Comput. Simul. 80 (2010) 1438.
[33] Y. Li, J. Kim, Comput. Math. Appl. 62 (2011) 737.
[34] H. Garcke, B. Nestler, B. Stoth, Physica D 115 (1998) 87.
[35] B. Nestler, H. Garcke, B. Stinner, Phys. Rev. E 71 (2005) 041609.

[36] H. Garcke, V. Styles, Interfaces Free Bound. 6 (2004) 271.
[37] R. Kornhuber, R. Krause, Comput. Visual. Sci. 9 (2006) 103.
[38] D.A. Kay, A. Tomasi, IEEE Trans. Image Process. 18 (2009) 2330.
[39] D.J. Eyre, An unconditionally stable one-step scheme for gradient systems, http://www.math.utah.edu/~eyre/research/methods/stable.ps.
[40] Y. Li, H.G. Lee, D. Jeong, J. Kim, Comput. Math. Appl. 60 (2010) 1591.
[41] S.-L. Wang, R.F. Sekerka, Phys. Rev. E 53 (1996) 3760.
[42] R.J. Braun, B.T. Murray, J. Soto Jr., Modell. Simul. Mater. Sci. Eng. 5 (1997) 365.
[43] N. Provatas, N. Goldenfeld, J. Dantzig, J. Comput. Phys. 148 (1999) 265.
[44] M. Plapp, A. Karma, J. Comput. Phys. 165 (2000) 592.
[45] C.W. Lan, C.M. Hsu, C.C. Liu, Y.C. Chang, Phys. Rev. E 65 (2002) 061601.
[46] B. Nestler, J. Cryst. Growth 275 (2005) e273.
[47] Y. Suwa, Y. Saito, H. Onodera, Scr. Mater. 55 (2006) 407.
[48] W.M. Feng, P. Yu, S.Y. Hu, Z.K. Liu, Q. Du, L.Q. Chen, J. Comput. Phys. 220 (2006) 498.
[49] P. Zhao, J.C. Heinrich, J. Comput. Phys. 173 (2001) 765.
[50] F. Gibou, R. Fedkiw, R. Caflisch, S. Osher, J. Sci. Comput. 19 (2002) 183.
[51] A. Badillo, C. Beckermann, Acta Mater. 54 (2006) 2015.
[52] B. Nestler, A.A. Wheeler, Physica D 138 (2000) 114.
[53] B. Nestler, A.A. Wheeler, L. Ratke, C. Stöcker, Physica D 141 (2000) 133.
[54] J. Tiaden, B. Nestler, H.J. Diepers, I. Steinbach, Physica D 115 (1998) 73.
[55] I. Steinbach, F. Pezzolla, Physica D 134 (1999) 385.
[56] P.C. Bollada, P.K. Jimack, A.M. Mullis, Physica D 241 (2012) 816.
[57] W.L. Briggs, A Multigrid Tutorial, SIAM, Philadelphia, PA, 1987.
[58] U. Trottenberg, C. Oosterlee, A. Schüller, Multigrid, Academic Press, London, 2001.
[59] A. Stuart, A.R. Humphries, Dynamical System and Numerical Analysis, Cambridge University Press, Cambridge, 1998.
[60] S. Gottlieb, C.-W. Shu, E. Tadmor, SIAM Rev. 43 (2001) 89.
[61] R.J. Spiteri, S.J. Ruuth, SIAM J. Numer. Anal. 40 (2003) 469.
[62] S. Gottlieb, D.I. Ketcheson, C.-W. Shu, J. Sci. Comput. 38 (2009) 251.
[63] C. Huang, Appl. Numer. Math. 59 (2009) 891.
[64] S.J. Ruuth, W. Hundsdorfer, J. Comput. Phys. 209 (2005) 226.
[65] A. Karma, W.-J. Rappel, Phys. Rev. E 57 (1998) 4323.
[66] L. Tan, N. Zabaras, J. Comput. Phys. 226 (2007) 131.
[67] J. Rosam, P.K. Jimack, A. Mullis, J. Comput. Phys. 225 (2007) 1271.
[68] G. Caginalp, Phys. Rev. A 39 (1989) 5887.
[69] J.S. Langer, Directions in Condensed Matter Physics, World Scientific, Singapore, 1986, pp. 165–186.
[70] A. Karma, W.-J. Rappel, Phys. Rev. E 53 (1996) R3017.