Computational Materials Science 71 (2013) 89–96

Contents lists available at SciVerse ScienceDirect

Computational Materials Science

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

A parallel multigrid method of the Cahn–Hilliard equation

Jaemin Shin, Sungki Kim, Dongsun Lee, Junseok Kim *

Department of Mathematics, Korea University, Seoul 136-713, Republic of Korea

ARTICLE INFO

Article history:
Received 11 October 2012
Received in revised form 19 December 2012
Accepted 14 January 2013

Keywords:
Parallel computing
Cahn–Hilliard equation
Multigrid
Phase separation
Linearly stabilized splitting scheme

ABSTRACT

We present a parallel finite difference scheme and its implementation for solving the Cahn–Hilliard equation, which describes the phase separation process. Our numerical algorithm employs an unconditionally gradient stable splitting discretization method. The resulting discrete equations are solved using a parallel multigrid method. This parallel scheme facilitates the solution of large-scale problems. We provide numerical results related to the speed-up, efficiency, and scalability to demonstrate the high performance of our proposed method. We also propose a linearly stabilized splitting scheme for the Cahn–Hilliard equation with logarithmic free energy.

1. Introduction

The Cahn–Hilliard (CH) equation was originally proposed by Cahn and Hilliard to model the spinodal decomposition and coarsening phenomena observed in binary alloys [1,2]. If m_1 and m_2 are the local masses of the material components 1 and 2 their mass concentrations are defined by $c_1 = m_1/(m_1 + m_2)$ and $c_2 = m_2/(m_1 + m_2)$, respectively. The distributions of components in a binary mixture are described by the mass concentrations c_1 and c_2. We use the difference of the concentrations $\phi = c_1 - c_2$ as an order parameter. The CH equation is

$$\frac{\partial \phi(\mathbf{x},t)}{\partial t} = \nabla \cdot [\mathbf{M}(\phi(\mathbf{x},t))\nabla \mu(\phi(\mathbf{x},t))], \quad \mathbf{x} \in \Omega, \ 0 < t \leqslant T, \tag{1}$$

$$\mu(\phi(\mathbf{x},t)) = F'(\phi(\mathbf{x},t)) - \epsilon^2 \Delta \phi(\mathbf{x},t). \tag{2}$$

The typical bulk free energy $F(\phi) = 0.25(\phi^2 - 1)^2$, is considered to be a double well potential [3] (Fig. 1). For simplicity, the mobility tensor $\mathbf{M}$ is assumed to be isotropic and constant, i.e., the identity matrix $\mathbf{M} = \mathbf{I}$. The homogenous Neumann boundary condition is set as $\mathbf{n} \cdot \nabla \phi = \mathbf{n} \cdot \nabla \mu = 0$, where $\mathbf{n}$ is the unit vector normal to $\partial\Omega$.

The CH equation arises from the Ginzburg–Landau free energy

$$\mathcal{E}(\phi) := \int_\Omega \left(F(\phi) + \frac{\epsilon^2}{2} |\nabla \phi|^2 \right) d\mathbf{x},$$

where $\Omega \subset \mathbb{R}^d$ $(d = 1, 2, 3)$, $F(\phi)$ is a free energy function, and ϵ is the gradient interfacial energy coefficient. A chemical potential μ is the variational derivative of $\mathcal{E}$ with respect to ϕ

$$\mu = \frac{\delta \mathcal{E}}{\delta \phi} = F'(\phi) - \epsilon^2 \Delta \phi.$$

Based on the law of mass conservation, we can derive the CH equation

$$\frac{\partial \phi}{\partial t} = -\nabla \cdot \mathcal{F},$$

where the flux is given by $\mathcal{F} = -\mathbf{M}\nabla\mu$ with a mobility tensor $\mathbf{M}$. We refer readers to [4] for the detailed mathematical derivation of the CH equation.

The CH equation is a prototypical model in theoretical materials science and it has subsequently been adopted to model many physical phenomena, including phase transitions and interface dynamics in multiphase fluids [5–12]. Other important applications of the CH equation are flow visualization [13], image processing [14], and the formation of quantum dots [15,16]. It has also been applied to studies of practical problems such as thermal decomposition in polymer blends [17] and spinodal decomposition in solder balls [18]. Thus, efficient and accurate numerical solutions of the equation are required to simulate large-scale dynamics.

Numerical approximations of the CH equation have been studied and developed by many authors using various numerical methods [3,7,11,19–21] including the finite difference, finite element, and spectral methods. However, the CH equation is notoriously difficult to solve numerically because the equation is rigid due to its biharmonic and nonlinear terms [22,23]. Because these terms

* 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).

0927-0256/$ - see front matter © 2013 Elsevier B.V. All rights reserved.
http://dx.doi.org/10.1016/j.commatsci.2013.01.008

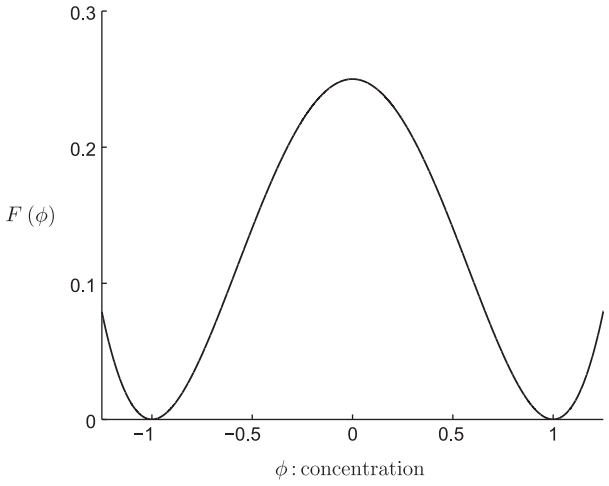


Fig. 1. Free energy $F(\phi) = 0.25(\phi^2 - 1)^2$.

degrade the numerical stability, it is computationally expensive to reach the desired time scale where interesting dynamics occur [3]. Recently, an advanced discontinuous Galerkin method [24] and isogeometric analysis [25] were developed to deal with the high-order spatial derivatives present in the CH equation.

A parallel algorithm was also applied to solving many scientific simulations, i.e., dendrite crystal growth [26–28], wetting phenomena [27], phase separation [29], and block copolymer structure [30,31] simulations. Many applied industrial sciences have overcome the problem of large-scale applications using parallel computing. Multigrid methods are generally accepted as being the fastest numerical methods for solving elliptic partial differential equations [32]. The multigrid method has been applied extensively to solving the CH equation [7–9,19]. Only few studies have been performed for solving the CH equation using parallel algorithms with multigrid methods [33]. On the other hand, the authors in [34] studied an efficient parallel multigrid method for solving the biharmonic problem based on the finite element techniques. In this paper, we present an unconditional stable scheme and implicit parallel multigrid method for solving the CH equation based on the finite difference scheme.

This paper is organized as follows. In Section 2, we present the finite difference scheme, multigrid method, and parallel strategies including the grid partitioning, data communication, and coarse level procedures. Numerical experiments that demonstrate the superiority of our parallel method are presented in Section 3. We present our conclusions in Section 4.

2. Numerical scheme

We consider finite difference approximations to obtain a linear discrete system from the CH equation. An unconditionally gradient stable scheme, which was proposed by Eyre [20,23], is applied for time discretization. A multigrid method is used to solve the resulting system at an implicit time level. A detailed description is provided in this section. In addition, we parallelize the multigrid method to solve the system.

2.1. Discretization

We discretize the CH Eqs. (1) and (2) in two-dimensional space $\Omega = (a,b) \times (c,d)$. Let $N_x = 2^m$ and $N_y = 2^n$ be the numbers of mesh points with integers m and n. The mesh sizes are defined as $\Delta x = (b-a)/N_x$ and $\Delta y = (d-c)/N_y$. We denote a discrete computational domain by $\Omega_{m,n} = \{(x_i, y_j)\colon\ x_i = a + (i - 0.5)\Delta x,\ \ y_j = c + (j - 0.5)\Delta y,\ \ 1 \leqslant i \leqslant N_x,\ \ 1 \leqslant j \leqslant N_y\}$, which is the set of cell-centered points. In addition, ϕ_{ij}^n and μ_{ij}^n are approximations of $\phi(x_i, y_j, t_n)$ and $\mu(x_i, y_j, t_n)$, where $t_n = n\Delta t$ and Δt is a temporal step. For simplicity, we assume that the unit domain $\Omega = (0,1) \times (0,1)$ and $N_x = N_y = 2^m$ is used, i.e., the uniform mesh grid $h = \Delta x = \Delta y$. We define the domain by $\Omega_m = \Omega_{m,n}$ if $m = n$.

Using the linearly stabilized splitting scheme [20], we implement an implicit time and centered difference space discretization of the CH Eqs. (1) and (2).

$$\frac{\phi_{ij}^{n+1} - \phi_{ij}^n}{\Delta t} = \Delta_d \mu_{ij}^{n+1}, \tag{3}$$

$$\mu_{ij}^{n+1} = 2\phi_{ij}^{n+1} - \epsilon^2 \Delta_d \phi_{ij}^{n+1} + \left(\phi_{ij}^n\right)^3 - 3\phi_{ij}^n, \tag{4}$$

where the discrete Laplacian operator is defined by $\Delta_d \phi_{ij}^{n+1} = \left(\phi_{i+1,j}^{n+1} + \phi_{i-1,j}^{n+1} + \phi_{i,j+1}^{n+1} + \phi_{i,j-1}^{n+1} - 4\phi_{ij}^{n+1}\right)/h^2$.

2.2. Multigrid V-cycle algorithm

In this section, we present a multigrid method for solving the discrete system (3) and (4) at the implicit time level. First, we represent the discrete CH system

$$L_h(\phi^{n+1}, \mu^{n+1}) = (\xi^n, \psi^n),$$

where the linear operator L_h is defined as

$$L_h(\phi^{n+1}, \mu^{n+1}) = \left(\frac{\phi^{n+1}}{\Delta t} - \Delta_d \mu^{n+1}, -2\phi^{n+1} + \epsilon^2 \Delta_d \phi^{n+1} + \mu^{n+1}\right),$$

and the source term is denoted by $(\xi^n, \psi^n) = (\phi^n/\Delta t, (\phi^n)^3 - 3\phi^n)$. Next, we describe the multigrid method, which includes the presmoothing, coarse grid correction, and postsmoothing steps. We denote a mesh grid Ω_k as the discrete domain for each multigrid level k. Note that a mesh grid Ω_k has $2^k \times 2^k$ grid points. Let k_{min} be the coarsest multigrid level. We now introduce the SMOOTH and V-cycle functions. Given the number ν_1 of pre-smoothing and ν_2 of post-smoothing relaxation sweeps, the V-cycle is used as an iteration step in the multigrid method.

2.2.1. Smoothing

Compute $(\bar{\phi}_k, \bar{\mu}_k)$ by applying ν smoothing procedures to (ϕ_k, μ_k).

$$(\bar{\phi}_k, \bar{\mu}_k) = \mathrm{SMOOTH}^{\nu}(\phi_k, \mu_k, L_h, \xi_k, \psi_k),$$

on a mesh grid Ω_k where $h = 1/2^k$. The SMOOTH^{ν} function means that it performs a SMOOTH relaxation operator with approximations ϕ_k and μ_k, and source terms ξ_k and ψ_k. The superscript ν denotes how many times the given relaxation operator is applied to obtain the updated approximations $(\bar{\phi}_k, \bar{\mu}_k)$. Eqs. (3) and (4) are rewritten to apply a relaxation operator

$$\frac{1}{\Delta t}\bar{\phi}_{ij} - \frac{4}{h^2}\bar{\mu}_{ij} = \xi_{ij} + \frac{\mu_{i-1,j} + \mu_{i+1,j} + \mu_{i,j-1} + \mu_{i,j+1}}{h^2}, \tag{5}$$

$$-\left(2 + \frac{4\epsilon^2}{h^2}\right)\bar{\phi}_{ij} + \bar{\mu}_{ij} = \psi_{ij} - \epsilon^2 \frac{\phi_{i-1,j} + \phi_{i+1,j} + \phi_{i,j-1} + \phi_{i,j+1}}{h^2}. \tag{6}$$

Here, the subscript k is omitted. One SMOOTH relaxation operator step is completed by solving the system (5) and (6) by a 2×2 matrix inversion for each i and j. This relaxation step is evaluated using pointwise Jacobi or Red–Black iterative methods. The degrees of parallelism for the Jacobi and Red–Black relaxations are $N_x N_y$ and $N_x N_y/2$, respectively.

2.2.2. V-cycle

One V-cycle step comprises the presmoothing, coarse grid correction, and postsmoothing steps. Please refer to the reference text for additional details and background [32,35].

$$\left(\phi_k^{n+1,m+1}, \mu_k^{n+1,m+1}\right) = \text{V-cycle}\left(k, \phi_k^{n+1,m}, \mu_k^{n+1,m}, L_h, \xi_k^n, \psi_k^n, v_1, v_2\right),$$

where $\left(\phi_k^{n+1,m+1}, \mu_k^{n+1,m+1}\right)$ and $\left(\phi_k^{n+1,m}, \mu_k^{n+1,m}\right)$ are the approximations of ϕ_k^{n+1} and μ_k^{n+1} before and after the V-cycle. Next, we define the V-cycle.

2.2.3. Presmoothing

Compute $\left(\bar{\phi}_k^{n+1,m}, \bar{\mu}_k^{n+1,m}\right)$ by applying v_1 smoothing procedures to $\left(\phi_k^{n+1,m}, \mu_k^{n+1,m}\right)$.

$$\left(\bar{\phi}_k^{n+1,m}, \bar{\mu}_k^{n+1,m}\right) = \text{SMOOTH}^{v_1}\left(\phi_k^{n+1,m}, \mu_k^{n+1,m}, L_h, \xi_k^n, \psi_k^n\right).$$

2.2.4. Coarse grid correction

(1) Find the defect:

$$\left(\bar{d}_{1,k}^m, \bar{d}_{2,k}^m\right) = \left(\xi_k^n, \psi_k^n\right) - L_k\left(\bar{\phi}_k^{n+1,m}, \bar{\mu}_k^{n+1,m}\right).$$

(2) Restrict the defect:

$$\bar{d}_{1,k-1}^m = I_k^{k-1}\bar{d}_{1,k}^m, \quad \bar{d}_{2,k-1}^m = I_k^{k-1}\bar{d}_{2,k}^m,$$

where the restriction operator I_k^{k-1}, mapping the k-level grid to the $(k-1)$-level grid, is defined by

$$\begin{aligned} d_{k-1}(x_i, y_j) &= I_k^{k-1} d_k \\ &= \frac{1}{4}[d_k(x_{2i-1}, y_{2j-1}) + d_k(x_{2i-1}, y_{2j}) + d_k(x_{2i}, y_{2j-1}) \\ &\quad + d_k(x_{2i}, y_{2j})]. \end{aligned}$$

(3) Evaluate approximations $\left(\hat{v}_{1,k-1}^{n+1,m}, \hat{v}_{2,k-1}^{n+1,m}\right)$ of the following coarse grid system on Ω_{k-1}:

$$L_{2h}\left(\hat{v}_{1,k-1}^{n+1,m}, \hat{v}_{2,k-1}^{n+1,m}\right) = \left(\bar{d}_{1,k-1}^m, \bar{d}_{2,k-1}^m\right).$$

If $k > k_{\min} + 1$, then we can solve the coarse the grid system using the zero grid functions as initial approximations and the defect functions as source terms

$$\left(\hat{v}_{1,k-1}^{n+1,m}, \hat{v}_{2,k-1}^{n+1,m}\right) = \text{V} - \text{cycle}\left(k-1, 0, 0, L_{2h}, \bar{d}_{1,k-1}^m, \bar{d}_{2,k-1}^m, v_1, v_2\right).$$

If $k = k_{min}$, then we apply the smoothing procedure to obtain the approximations.

(4) Interpolate (or prolongate) the correction:

$$\hat{v}_{1,k}^{n+1} = I_{k-1}^k \hat{v}_{1,k-1}^{n+1,m}, \hat{v}_{2,k}^{n+1,m} = I_{k-1}^k \hat{v}_{2,k-1}^{n+1,m},$$

where the prolongation operator I_{k-1}^k, mapping the $(k-1)$-level grid to the k-level grid, is defined by

$$\left.\begin{array}{l} v_k(x_{2i-1}, y_{2j-1}) \\ v_k(x_{2i-1}, y_{2j}) \\ v_k(x_{2i}, y_{2j-1}) \\ v_k(x_{2i}, y_{2j}) \end{array}\right\} = I_{k-1}^k v_{k-1} = v_{k-1}(x_i, y_j).$$

(5) Compute the corrected approximation on Ω_k:

$$\left(\tilde{\phi}_k^{n+1,m}, \tilde{\mu}_k^{n+1,m}\right) = \left(\bar{\phi}_k^{n+1,m}, \bar{\mu}_k^{n+1,m}\right) + \left(\hat{v}_{1,k}^{n+1,m}, \hat{v}_{2,k}^{n+1,m}\right),$$

where $\left(\tilde{\phi}_k^{n+1,m}, \tilde{\mu}_k^{n+1,m}\right)$ is the approximation after coarse grid correction.

2.2.5. Postsmoothing

Compute $\left(\phi_k^{n+1,m+1}, \mu_k^{n+1,m+1}\right)$ by applying the v_2 smoothing procedures to $\left(\tilde{\phi}_k^{n+1,m}, \tilde{\mu}_k^{n+1,m}\right)$.

$$\left(\phi_k^{n+1,m+1}, \mu_k^{n+1,m+1}\right) = \text{SMOOTH}^{v_2}\left(\tilde{\phi}_k^{n+1,m}, \tilde{\mu}_k^{n+1,m}, L_h, \xi_k^n, \psi_k^n\right).$$

This completes the description of the V-cycle.

2.3. Parallel computation

Large-scale problems are solved using a fast and accurate solver, and by parallelizing the solver as efficiently as possible. In this section, we describe the parallel multigrid algorithm used to avoid a long running time and to reduce the memory requirements on an individual processor without losing any of the multigrid's advantages. This method is based on grid partitioning of the discrete computational domain and data communication between adjacent processors. These processes are at the heart of parallel multigrid problem. In order to develop an efficient multigrid parallel algorithm, we divide the domain into smaller blocks. This reduces the unprocessed pending events when applying the multigrid method.

2.3.1. Grid partitioning

Grid partitioning is a strategy for parallelizing a mesh grid. To improve the efficiency of the parallel algorithm, we use a uniform distribution of processors based on the single program multiple data (SPMD) structure [36]. Let P be the total number of processors. The x and y directional numbers of the processors are denoted by P_x and P_y such that $P = P_x \times P_y$. In addition, Ω_k is the discrete domain having $2^k \times 2^k$ grid points and let Ω_k^r be its rth subdomain for $r = 0, 1, \ldots, P-1$, where the subscript k is the multigrid level. The discrete domain is divided and allocated to processors for column- or row-wise ordering (known as lexicographical ordering, which starts at the upper left corner).

Fig. 2 shows examples of the grid partitioning strategy when P is 2, 4, 8, and 16. If the number of processors is $P = 2^{2p}$ where p is a non-negative integer, we select the strategy $P_x \times P_y = 2^p \times 2^p$ as shown in Fig. 2b and d. The coarsest multigrid level k_{min} is greater than or equal to p, because the discrete domain at the coarsest level must contain at least $2^p \times 2^p$ grid points. However, if the number of processors is $P = 2^{2p+1}$, we select the strategy $P_x \times P_y = 2^{\{p+1\}} \times 2^p$ as shown in Fig. 2a and c. The coarsest multigrid level k_{min} is greater than or equal to p+1, because the discrete domain at the coarsest level must contain $2^{p+1} \times 2^{p+1}$ grid points. At the coarsest level, the grid points for each processor are 1×2. For each processor, the ratio of the discrete subdomain Ω_k^r is constant for all multigrid levels k.

2.3.2. Data communication

Data communication, i.e., exchange of data among adjacent processors, is achieved by the synchronization process. This implies that all communications are performed before the smoothing and defect procedures because of the requirement for implementing the discrete Laplacian operator. On the other hand, no data communication is needed before restriction and prolongation in the V-cycle. For example, if we take $P = 2^{2p}$, Fig. 3 shows the rth processor and its adjacent processors, $r \pm P_x$ and $r \pm 1$, which exchange data between each other. The shaded layer in Fig. 3 is known as the sub-boundary layer. This layer is used for exchanging data between the adjacent processors. Note that the Message Passing Interface (MPI) [37] is used for interprocessor communications. After finishing the data communications, the smoothing procedure can sweep over the subdomains and the defects can be calculated for each processor.

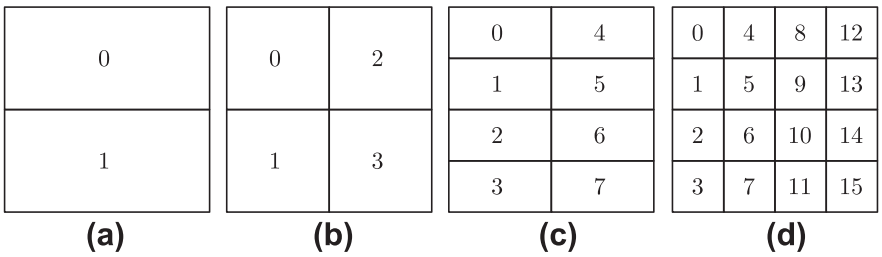


Fig. 2. Examples of strategies: (a) two subdomains $P_x \times P_y = 2 \times 1$, (b) four subdomains $P_x \times P_y = 2 \times 2$, (c) eight subdomains $P_x \times P_y = 4 \times 2$, and (d) 16 subdomains $P_x \times P_y = 4 \times 4$.

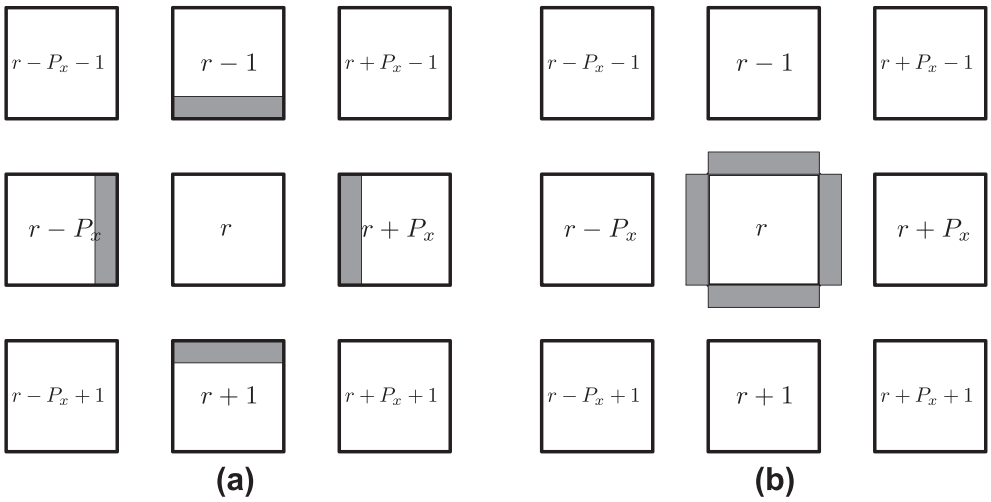


Fig. 3. Data communication on the rth processor; (a) sending data to and (b) receiving data from adjacent processors.

2.3.3. Coarse level procedure

The multigrid method is one of the fastest numerical algorithms for solving elliptic partial differential equations. However, its efficient parallelization is hampered by the poor computation-to-computation ratio on coarse grids [38]. The efficiency of parallelism becomes worse as lower multigrid levels k. In the worst case, the discrete domain of each processor is a 1×1 mesh grid and the communication data volume is larger than that of the computational data.

We propose a coarse level procedure that uses the residual error. Before progressing to a coarser level (after the defect procedure during the coarse grid correction), the algorithm does not move down to the next level if the maximum values of both the residuals $\bar{d}^m_{1,k}$ and $\bar{d}^m_{2,k}$ are less than 0.001. This additional reduction step avoids wasteful communications at the coarse levels and it does not seriously affect the number of V-cycles required for convergence.

3. Numerical results

We present typical numerical test results for the phase separation of the CH equation using large grid sizes, and demonstrate the speed-up, efficiency, and scalability. We also propose a linearly stabilized splitting method for the logarithmic free energy, and present numerical results.

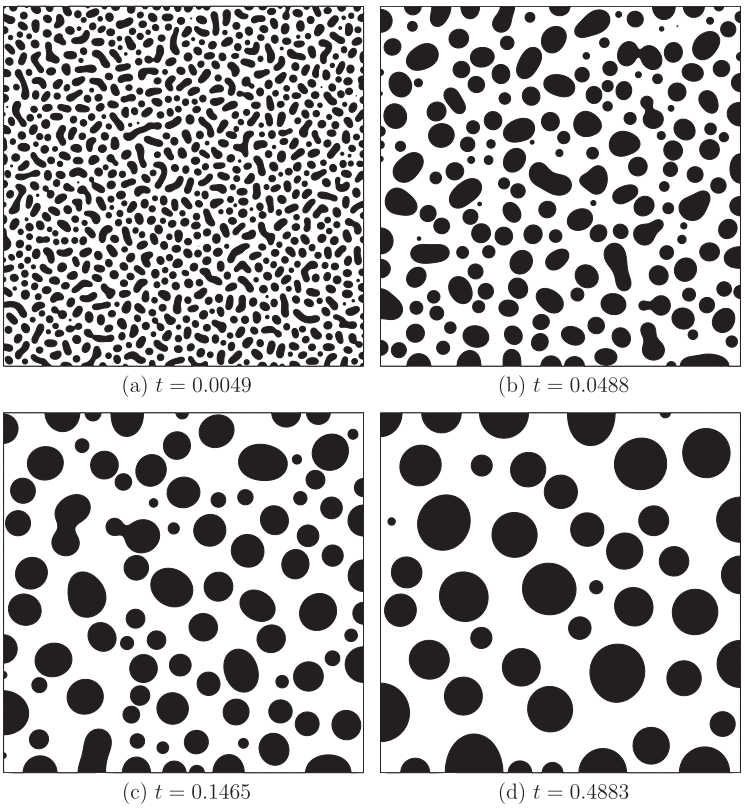
3.1. Phase separations

We simulate the dynamics of a droplet pattern and co-continuous network, which are typical cases in the evolution of the CH equation. The mesh size $2^{11} \times 2^{11}$, spatial step size $h = 1/2^{11}$, $\epsilon = 0.0014$, and temporal step size $\Delta t = 0.1h$ are used as parameters. First, the initial condition is a random perturbation where the average is -0.3 and the amplitude is 0.01: $\phi(x,y,0) = -0.3 + 0.01\text{rand}(x,y)$, where rand$(x,y)$ is a random value between -1 and 1. Second, the initial condition is a random perturbation where the average is zero and the amplitude is 0.01: $\phi(x,y,0) = 0.01\text{rand}(x,y)$. We use 2^8 processors for the simulations.

Figs. 4 and 5 correspond to the first and second cases, respectively. The two general features of the CH equation are a rapid phase separation followed by a slower coarsening process. Fig. 4 shows the droplet pattern, i.e., dark regions, in which one of the components, is nucleated and the regions grow. Fig. 5 shows the highly inter-connected pattern; after the phase separation, where the dark and light regions have a co-continuous phase, the coarsening process starts.

3.2. Speed-up and efficiency

The parallel performance is usually evaluated based on speed-up and the efficiency. The value of the speed-up is used to measure the ratio of the time spent in the serial mode and that spent in the parallel mode. Let $T(P)$ be the execution time, i.e., the computational time, using P processors. The speed-up $S(P)$ is defined as $S(P) = T(1)/T(P)$. However, the efficiency of the parallel algorithm is measured based on the processor utilization. The efficiency $E(P)$ is defined as $E(P) = S(P)/P$, which is the speed-up divided by the number of processors. Ideally, the algorithm aim to achieve $S(P) \approx P$, or equivalently $E(P) \approx 1$.

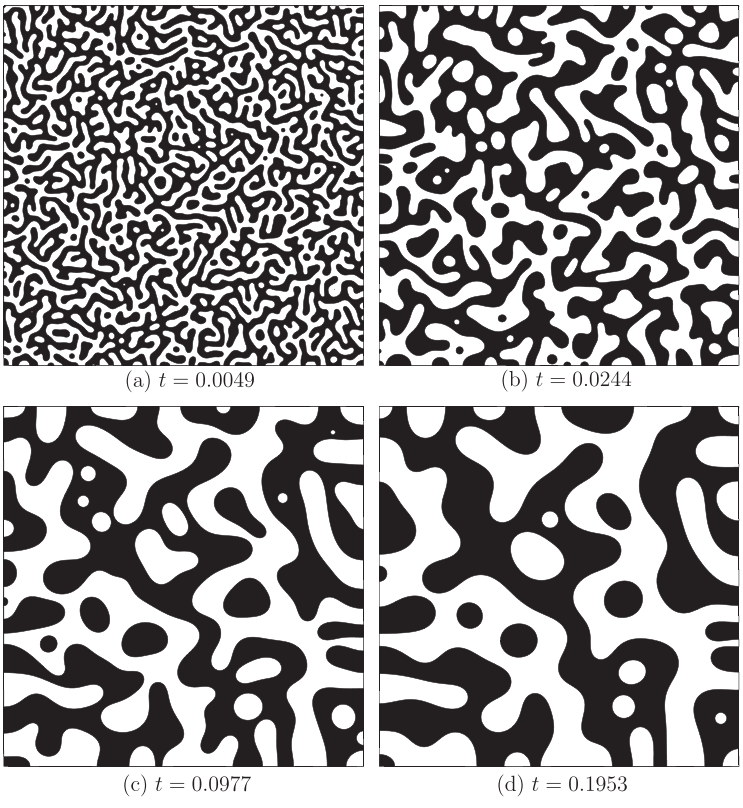


(a) $t = 0.0049$ (b) $t = 0.0488$

(c) $t = 0.1465$ (d) $t = 0.4883$

Fig. 4. Evolution of the average concentration 35% using a $2^{11} \times 2^{11}$ mesh grid.

Tables 1–3 show the execution time $T(P)$, speed-up $S(P)$, and efficiency $E(P)$ with different mesh sizes. The discrete computational domains $N_x \times N_y = 2^p \times 2^p$ are simulated using $\epsilon = 0.0038 \times 2^{9-p}$ for $p = 9,\ 10,\ 11$. The execution time is the elapsed time taken to reach $t = 100\Delta t$. All of the calculations are performed relative to with the initial data $\phi(x,y,0) = -0.3 + \text{rand}$



(a) $t = 0.0049$ (b) $t = 0.0244$

(c) $t = 0.0977$ (d) $t = 0.1953$

Fig. 5. Evolution of the average concentration 50% using a $2^{11} \times 2^{11}$ mesh grid.

(x,y). The spatial step size $h = 1/N_x$ and temporal step size $\Delta t = h$ are the other parameters. The number of processors P is varied from 1 to 64 and increased by a factor of 2.

The speed-up increases as the number of processors increases. On the other hand, the efficiency decreases as the number of processors increases. If $E(P) > 0.9$, we can regard the parallel computing as performing properly. Tables 1–3 demonstrate that the speed-up $S(P)$ and efficiency $E(P)$ are the best when the number of processors P is relatively small. For $2^{11} \times 2^{11}$ mesh grid, in Table 3, the execution time decreases almost linearly as the number of processors increases. The parallel performance is good until using 64 processors are used. For the mesh grids of $2^{10} \times 2^{10}$ and $2^9 \times 2^9$, however, the parallel computing performance is good until 32 and 8 processors are used (Tables 1 and 2). The relatively small size after grid partitioning degrades the performance.

3.3. Scalability

The parallel performance is also evaluated based on scalability. Fig. 6 shows the execution time versus the number of mesh sizes. The solid lines shows the ideal trends which are determined by the execution time of the discrete domain $2^{11} \times 2^{11}$ multiplied by $N_x \times N_y/2^{22}$. Symbols indicate the number of processors. The scalability is high because the execution times are similar to the associated ideal lines. When the number of processors P range from 1 to 8, the execution times are almost ideal. When the number of processors range from 16 to 64, there are differences between the execution time and the ideal line in some cases. However, this is expected because the parallel method is not improved with a large number of processors when simulating a relatively small mesh size.

Table 1
Execution time (s), speed-up, and efficiency using a mesh size $2^9 \times 2^9$.

P	1	2	4	8	16	32	64
$T(P)$	115.3	58.4	30.0	15.2	8.1	4.5	2.8
$S(P)$	1	1.97	3.85	7.59	14.27	25.90	41.63
$E(P)$	1	0.99	0.96	0.95	0.89	0.81	0.65

Table 2
Execution time (s), speed-up, and efficiency using a mesh size $2^{10} \times 2^{10}$.

P	1	2	4	8	16	32	64
$T(P)$	479.6	239.7	120.1	60.6	30.4	15.9	8.8
$S(P)$	1	2.00	3.99	7.91	15.77	30.08	54.47
$E(P)$	1	1.00	0.99	0.98	0.98	0.94	0.85

Table 3
Execution time (s), speed-up, and efficiency using a mesh size $2^{11} \times 2^{11}$.

P	1	2	4	8	16	32	64
$T(P)$	1964.0	974.7	495.1	245.9	123.3	63.4	33.9
$S(P)$	1	2.02	3.97	7.99	15.93	30.97	57.96
$E(P)$	1	1.00	0.99	0.99	0.99	0.97	0.91

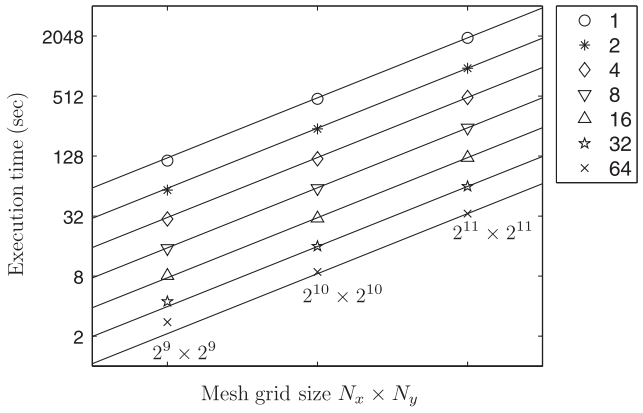


Fig. 6. Parallel scalability. Execution time versus the number of mesh sizes with different numbers of processors.

3.4. Logarithmic free energy

We now consider the symmetric logarithmic free energy

$$F(\phi) = \theta((1-\phi)\ln(1-\phi) + (1+\phi)\ln(1+\phi)) - \chi\phi^2,$$

where θ and χ are positive constant values. Fig. 7 shows an example of the logarithmic free energy $F(\phi)$ for $\theta = 5$ and $\chi = 8$. We denote the local minima by ϕ_α and ϕ_β for $\phi_\alpha < \phi_\beta$.

Eyre reported that, if the free energy functional of the CH equation is split into contractive and expansive parts, the splitting method becomes an unconditionally gradient stable scheme [20]. Thus, we propose a splitting method for the logarithmic free energy as follows: $\mathcal{E}(\phi) = \mathcal{E}_c(\phi) - \mathcal{E}_e(\phi)$ and $F(\phi) = F_c(\phi) - F_e(\phi)$, where

$$\mathcal{E}_c(\phi) = \int_\Omega \frac{\epsilon^2}{2}|\nabla\phi|^2 + F_c(\phi)dx = \int_\Omega \frac{\epsilon^2}{2}|\nabla\phi|^2 + \alpha\chi\phi^2 dx,$$

$$\mathcal{E}_e(\phi) = \int_\Omega F_e(\phi)dx,$$

$$= -\int_\Omega \theta((1-\phi)\ln(1-\phi) + (1+\phi)\ln(1+\phi)) - (1+\alpha)\chi\phi^2 dx.$$

$\mathcal{E}_c(\phi)$ and $\mathcal{E}_e(\phi)$ are the contractive and expansive parts, respectively. For a sufficiently large number α, $F_c(\phi)$ and $F_e(\phi)$ are convex functions on the interval $[\phi_\alpha, \phi_\beta]$. We then treat the contractive part $\mathcal{E}_c(\phi)$ implicitly and the expansive part $\mathcal{E}_e(\phi)$ explicitly. The linear operator L_h is defined as

$$L_h(\phi^{n+1}, \mu^{n+1}) = \left(\frac{\phi^{n+1}}{\Delta t} - \Delta_d\mu^{n+1}, -2\alpha\chi\phi^{n+1} + \epsilon^2\Delta_d\phi^{n+1} + \mu^{n+1}\right),$$

and the source term is denoted by

$$(\xi^n, \psi^n) = \left(\frac{\phi^n}{\Delta t}, \theta\ln\left(\frac{1+\phi^n}{1-\phi^n}\right) - 2(1+\alpha)\chi\phi^n\right).$$

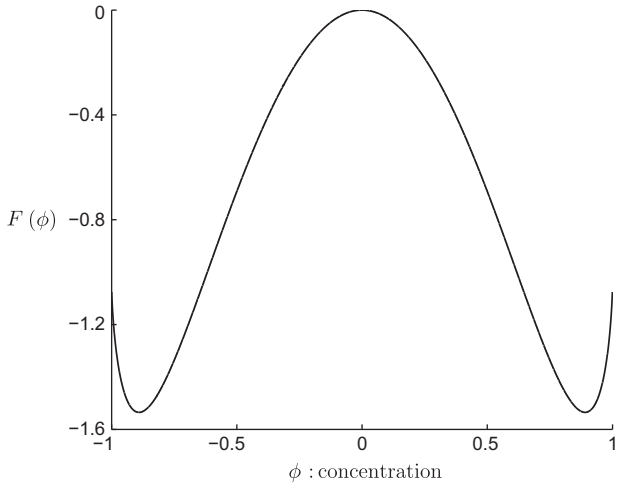


Fig. 7. Logarithmic free energy $F(\phi) = 5((1-\phi)\ln(1-\phi) + (1+\phi)\ln(1+\phi)) - 8\phi^2$.

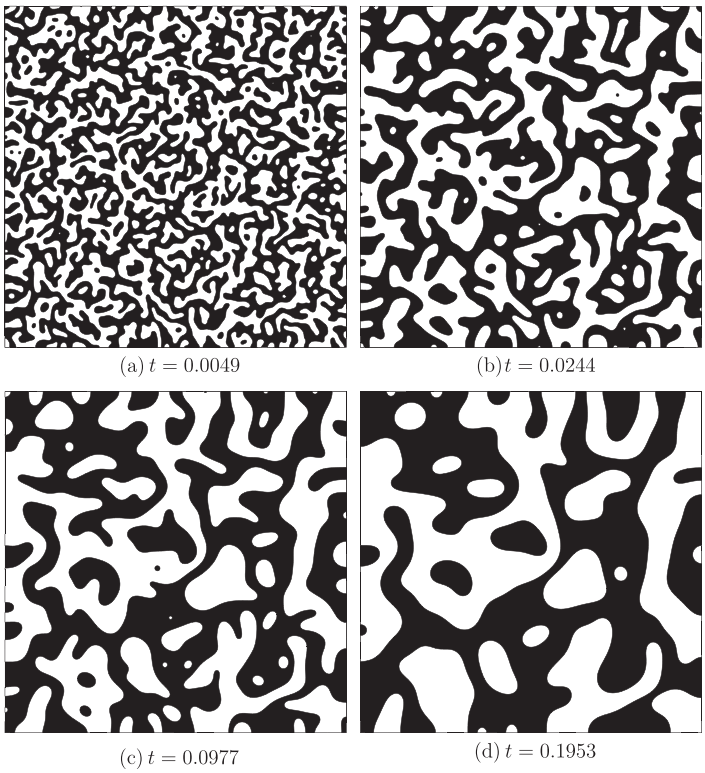


Fig. 8. Evolution of the average concentration 50% using a $2^{11} \times 2^{11}$ mesh grid mesh grid with the logarithmic free energy.

We then solve the linear system $L_h(\phi^{n+1}, \mu^{n+1}) = (\xi^n, \psi^n)$ by using the multigrid method. We refer the reader to [39] for a nonlinear splitting scheme of a logarithmic free energy.

Fig. 8 shows the spinodal decomposition of a binary mixture with the logarithmic free energy. The initial condition is $\phi(x,y,0) = 0.01\text{rand}(x,y)$. In this simulation, we use a $2^{11} \times 2^{11}$ mesh grid with spatial step size $h = 1/2^{11}$, and temporal step size $\Delta t = 0.1h$. The other parameters are $\epsilon = 0.0019$, $\theta = 5$, $\chi = 8$, and $\alpha = 2$. Fig. 8 shows the evolution at different times. The results of Fig. 8 also show the inter-connected patterns of the spinodal decomposition, which are similar to the results shown in Fig. 5.

4. Conclusion

In this paper, we presented a parallel multigrid scheme for solving the CH equation. We partitioned the discrete domain to minimize interprocessor communications when the number of processors is known and each partitioned domain is assigned to a different processor. We reduced computational costs by using an unconditionally stable splitting scheme and a parallel multigrid method. With a larger computational discrete domain, the proposed method is efficient and saves computational time. In addition, we proposed the linearly stabilized splitting scheme for solving the CH equation with logarithmic free energy. Our future studies will focus on extending this parallel scheme to complex applications such as multi-component multi-phase systems or the dynamics of red blood cells in blood vessels.

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 simulations were performed using the GAIA supercomputing cluster at the KISTI Supercomputing Center. The authors thank the reviewers for their useful and constructive comments on this article.

References

[1] J.W. Cahn, Acta Mater. 9 (1961) 795–801.
[2] J.W. Cahn, J.E. Hilliard, J. Chem. Phys. 28 (1958) 258–267.
[3] E.V.L. Mello, O.T.S. Filho, Physica A 347 (2005) 429–443.
[4] P.C. Fife, Electron. J. Differ. Equ. 48 (2000) 1–26.
[5] M. Gurtin, Physica D 92 (1996) 178–192.
[6] J.E. Hilliard, Spinodal decomposition, in: Phase Transformations, ASM, Cleveland, Ohio, 1970, pp. 497–560 (Chapter 12).
[7] J. Kim, K. Kang, J. Lowengrub, J. Comput. Phys. 193 (2004) 511–543.
[8] J. Kim, J. Comput. Phys. 204 (2005) 784–804.
[9] J. Kim, J. Lowengrub, Interface Free Bound. 7 (2005) 435–466.
[10] D. Anders, K. Weinberg, Angew. Math. Mech. 91 (2011) 609–629.
[11] C.M. Elliott, D.A. French, IMA J. Appl. Math. 38 (1987) 97–128.
[12] D. Anders, K. Weinberg, Comput. Mater. Sci. 50 (2011) 1359–1364.
[13] H. Garcke, T. Preusser, M. Rumpf, A. Telea, U. Weikard, J.J. van Wijk, IEEE Trans. Vis. Comput. Graph. 7 (2001) 230–241.
[14] A.L. Bertozzi, S. Esedoglu, A. Gillette, IEEE Trans. Image Process. 16 (2007) 285–291.
[15] S.M. Wise, J.S. Lowengrub, J.S. Kim, K. Thornton, P.W. Voorhees, W.C. Johnson, Appl. Phys. Lett. 87 (2005) 133102-1–133102-3.
[16] X.D. Liang, Y. Ni, L.H. He, Comput. Mater. Sci. 48 (2010) 871–874.
[17] D. Anders, K. Weinberg, Mech. Mater. 47 (2012) 33–50.
[18] D. Anders, C. Hesch, K. Weinberg, Int. J. Solids Struct. 49 (2012) 1557–1572.
[19] D. Kay, R. Welford, J. Comput. Phys. 212 (2006) 288–304.
[20] D.J. Eyre, An Unconditionally Stable One-Step Scheme for Gradient Systems, 1998. <http://www.math.utah.edu/~eyre/research/methods/stable.ps> (unpublished article).
[21] J. Zhu, L.Q. Chen, J. Shen, V. Tikare, Phys. Rev. 60 (1999) 3564–3572.
[22] C.M. Elliott, The Cahn–Hilliard model for the kinetics of phase separation, in: J.F. Rodrigues (Ed.), Mathematical Models for Phase Change Problems, International Series of Numerical Mathematics, vol. 88, Birkhüser Verlag, Basel, 1989, pp. 35–73.
[23] D.J. Eyre, Mater. Res. Soc. Sympos. Proc. 529 (1998) 39–46.
[24] G.N. Wells, E. Kuhl, K. Garikipati, J. Comput. Phys. 218 (2006) 860–877.
[25] H. Gomez, V.M. Calo, Y. Bazilevs, T.J.R. Hughes, Comput. Meth. Appl. Mech. Eng. 197 (2008) 4333–4352.
[26] J.H. Jeong, N. Goldenfeld, J.A. Dantzig, Phys. Rev. E 64 (2001) 1–14.
[27] M. Do-Quang, W. Villanueva, G. Amberg, I. Loginova, Bull. Pol. Acad. Sci. Technol. Sci. 55 (2007) 229–237.
[28] Z. Guo, J. Mi, P.S. Grant, J. Comput. Phys. 231 (2011) 1781–1796.

[29] D. Playne, K. Hawick, Data parallel three-dimensional Cahn–Hilliard field equation simulation on GPUs with CUDA, in: Proc. 2009 International Conference on Parallel and Distributed Processing Techniques and Applications, Las Vegas, USA, 2009.
[30] S.W. Sides, G.H. Fredrickson, Polymer 44 (2003) 5859–5866.
[31] X. Guo, M. Pinna, A.V. Zvelindovsky, Macromol. Theor. Simul. 16 (2007) 779–784.
[32] U. Trottenberg, C. Oosterlee, A. Schüller, Multigrid, Academic Press, London, 2001.
[33] S. Vey, A. Voigt, Computing 81 (2007) 53–75.
[34] M.D. Mihajlović, D.J. Silvester, Parallel Comput. 30 (2004) 35–55.
[35] MGNET. <http://www.mgnet.org>.
[36] A. Grama, A. Gupta, G. Karypis, V. Kumar, Introduction to Parallel Computing, second ed., Addison-Wesley, London, 2003.
[37] Message Passing Interface Forum, MPI, A Message Passing Interface Standard, 1994.
[38] F. Hulsemann, M. Kowarschik, M. Mohr, U. Rude, Parallel geometric multigrid, in: A.M. Bruaset, A. Tveito (Eds.), Numerical Solution of Partial Differential Equations on Parallel Computers, Springer-Verlag, 2005.
[39] H. Gomez, T.J.R. Hughes, J. Comput. Phys. 230 (2011) 5310–5327.