Available online at www.sciencedirect.com

SCIENCE DIRECT®

Journal of Computational Physics 212 (2006) 288–304

JOURNAL OF
COMPUTATIONAL
PHYSICS

www.elsevier.com/locate/jcp

A multigrid finite element solver for the Cahn–Hilliard equation

David Kay, Richard Welford [*,1]

Department of Mathematics, Mantell Building, University of Sussex, Falmer, Brighton, BN1 9RF, UK

Received 23 September 2004; received in revised form 19 May 2005; accepted 4 July 2005
Available online 15 August 2005

Abstract

A multigrid finite element solver for the Cahn–Hilliard equation is presented that has mesh-independent convergence rates for any time-step size, including in the important limit $\epsilon \to 0$ which is examined via numerical examples. Numerics are performed for a number of test problems which show that the features of the Cahn–Hilliard equation (minimising interface measure, Lyapunov energy functional etc.) are preserved. We also explore the use of this solver in conjunction with adaptive time-stepping and adaptive mesh strategies.

PACS: 65M55; 65Z05; 74S05; 82B26

Keywords: Cahn–Hilliard equation; Finite element; Non-linear multigrid method

1. Introduction

The Cahn–Hilliard equation [6,7] was originally introduced as a phenomenological model of phase separation in a binary alloy. More recently, it has been used to study phase transitions and interface dynamics, related free boundary problems, multiphase fluids and polymer solutions, see [5,17,21] and the references therein.

The equation, together with appropriate initial and boundary conditions,

$$\frac{\partial c}{\partial t} - \nabla \cdot (B(c)\nabla w) = 0, \tag{1a}$$

[*] Corresponding author.
E-mail address: r.r.welford@sussex.ac.uk (R. Welford).

[1] Supported by EPSRC Grant GR/R8951/01.

0021-9991/$ - see front matter © 2005 Elsevier Inc. All rights reserved.
doi:10.1016/j.jcp.2005.07.004

$$w - \frac{1}{\epsilon}\Phi'(c) + \epsilon\Delta c = 0 \qquad \text{in } \Omega_T := \Omega \times (0,T), \tag{1b}$$

$$\frac{\partial c}{\partial \hat{n}} = B\frac{\partial w}{\partial \hat{n}} = 0 \qquad \text{on } \partial\Omega_T := \partial\Omega \times (0,T), \tag{1c}$$

$$c(\mathbf{x},0) = c_0(\mathbf{x}), \tag{1d}$$

where $\Omega \subset \mathbb{R}^d (d = 1, \ldots, 3)$ is a bounded polygonal domain and $\hat{n}$ is the outward pointing unit normal to Ω, form (in effect) a fourth order non-linear parabolic partial differential equation. The parameter ϵ is a measure of the interfacial thickness and Φ is a double well potential with its global minima at $c = \pm 1$, typically

$$\Phi(c) = \frac{1}{4}(1 - c^2)^2. \tag{2}$$

For derivation and analysis of the equation we refer to [12,13,10,15] and the references therein. The mobility function $B(c)$ is often taken to be constant ($B \equiv 1$). Here, we consider this choice as well as the thermodynamically reasonable choice [8]

$$B(c) = [1 - c^2]_+ := \max\{0, 1 - c^2\}. \tag{3}$$

It is well known that the Cahn–Hilliard equation is a gradient flow of the Lyapunov energy functional:

$$J(c) = \int_\Omega \frac{1}{\epsilon}\Phi(c) + \frac{\epsilon}{2}|\nabla c|^2 \,\mathrm{d}\mathbf{x}, \tag{4}$$

which is therefore non-increasing in time, we wish this to carry over to a discrete version of this functional in our numerical scheme.

Numerical simulations of the Cahn–Hilliard equation are difficult on a normal computer in a reasonable time because fourth order non-linear equations impose severe time-step restrictions on explicit methods ($\tau \sim h^4$) so implicit methods must be used. Additionally in order to fully capture the interface dynamics high spatial resolution is required, typically at least 8–10 elements (see [14], if there are an insufficient number of elements spurious numerical solutions can be introduced). In complicated problems that require uniform refinement this leads to large algebraic problems to be solved at each time-step.

The interface profile is given by (see for example [11])

$$\tanh\left(\frac{1}{\epsilon\sqrt{2}}x\right). \tag{5}$$

Thus, if we consider the interfacial region to be where the concentration c varies between -0.99 and 0.99 a simple calculation yields that the interface has its width given by width $\approx 7.5\epsilon$ (we note that this is more than twice as large as the interface for the "double-obstacle" free energy [4] which has width $\pi\epsilon$).

Traditionally iterative solution methods such as Gauss–Seidel have been used (for example [10]) but these suffer from slow convergence rates, typically O(N), where N is the number of points in the discretisation. Each iteration of a Gauss–Seidel solver is an O(N) operation and under a suitable choice of semi-implicit temporal discretisation (see below) there are no time-step restrictions. This makes the total work required for the Gauss–Seidel method O(N^2). A non-linear multigrid method for a finite difference approximation of this problem has recently been suggested by Kim et al. [17,18] which exhibits convergence rates independent of N as long as a weak time-step restriction related to the initial data is observed. The method presented here is an extension of that of Kornhuber and Krause [20] which has convergence rates independent of N with no time-step restriction. Each iteration of this multigrid solver is an O(N) operation, making the total work required O(N).

The outline of this paper is as follows: in Section 2, we describe the discretisation of the equations by a semi-implicit backward Euler method in time and continuous piecewise linear finite elements in space. Section 3 describes the multigrid algorithm. Sections 4 and 5 discuss some numerical results in both one and two space dimensions and in Sections 6 and 7, we look at how adaptive time-stepping and non-uniform meshes effect the solver.

2. Discretisation

Consider a uniform discretisation in time by a semi-implicit backward Euler method, so that $c^0(\mathbf{x}) = c(\mathbf{x},0)$, $c^n(\mathbf{x}) = c(\mathbf{x},\tau n)$, $n = 1,\ldots,N_T$:

$$\frac{c^n - c^{n-1}}{\tau} - \nabla\cdot(B(c^{n-1})\nabla w^n) = 0, \tag{6a}$$

$$\frac{1}{\epsilon}\phi(c^n) - \frac{1}{\epsilon}c^{n-1} - \epsilon\Delta c^n - w^n = 0, \tag{6b}$$

where we have used the splitting

$$\Phi'(c) = c^3 - c =: \ \phi(c) - c \tag{7}$$

so that ϕ is a monotone function. The weak form of the equations is then given by:

$$\left(c^n - c^{n-1},\chi\right)_{L^2(\Omega)} + \tau(B(c^{n-1})\nabla w^n,\nabla\chi)_{L^2(\Omega)} = 0 \quad \forall\,\chi\in H^1(\Omega), \tag{8a}$$

$$\epsilon(\nabla c^n,\nabla\chi)_{L^2(\Omega)} + \left(\frac{1}{\epsilon}\phi(c^n) - \frac{1}{\epsilon}c^{n-1} - w^n,\chi\right)_{L^2(\Omega)} = 0 \quad \forall\,\chi\in H^1(\Omega), \tag{8b}$$

where $H^1(\Omega)$ is the standard Sobolev space of $L^2(\Omega)$ functions with first derivative in $L^2(\Omega)$, and $(a,b)_{L^2(\Omega)} := \int_\Omega a\cdot b\,\mathrm{d}\mathbf{x}$ is the $L^2(\Omega)$ inner product. This method of discretising the equations is known to be unconditionally gradient stable [14] (stable for all time-step sizes), ensuring that the discrete free energy decreases monotonically. However, we must be careful when choosing time-step sizes to ensure that the dynamics of the system are accurately captured.

We discretise in space by continuous piecewise linear finite elements as follows. Given that Ω is a polygonal domain we partition Ω into a set, $\mathcal{T}^h$, of disjoint open simplices κ such that:

(1) $h_\kappa: = diam(\kappa)$,
(2) $h := \max_{\kappa\in\mathcal{T}^h}\{h_\kappa\}$,
(3) $\overline{\Omega} = \bigcup_{\kappa\in\mathcal{T}^h}\overline{\kappa}$.

Associated with the partitioning is the finite element space

$$V^h = \{\chi\in C(\overline{\Omega}):\ \chi|_\kappa \text{ is linear } \forall\,\kappa\in\mathcal{T}^h\}\subset H^1(\Omega). \tag{9}$$

In the remainder of this paper we drop the h-superscript for finite dimensional spaces and functions for ease of notation, thus $V = V^h$.

Let $\{x_i\}_{i=1}^N$ be the nodes of $\mathcal{T}^h$ and let $\{\eta_i\}_{i=1}^N$ be the standard Lagrange 'hat' basis functions (Fig. 1),

$$\eta_i\in V, \quad \eta_i(x_j) = \delta_{ij} \quad \forall\,i,j = 1,\ldots,N. \tag{10}$$

Thus, V is simply $\mathbb{R}^N$.

For $u,v\in C(\overline{\Omega})$, we introduce the lumped L^2 inner product given by:

$$(u,v)^h := \sum_j u(x_j)v(x_j)(\eta_i,1)_{L^2(\Omega)}. \tag{11}$$

The fully discrete equations are therefore:

$$\left(c^n - c^{n-1}, \chi\right)^h + \tau(B(c^{n-1})\nabla w^n, \nabla\chi)_{L^2(\Omega)} = 0 \qquad \forall\, \chi \in V, \tag{12a}$$

$$\epsilon(\nabla c^n, \nabla\chi)_{L^2(\Omega)} + \left(\frac{1}{\epsilon}\phi(c^n) - \frac{1}{\epsilon}c^{n-1} - w^n, \chi\right)^h = 0 \qquad \forall\, \chi \in V, \tag{12b}$$

where $c,w \in V$. Using the matrices

$$A_{ij} = (\nabla\eta_i, \nabla\eta_j)_{L^2(\Omega)},$$
$$C(\mathbf{c}^{n-1})_{ij} := C_{ij} = (B(c^{n-1}(x_i))\nabla\eta_i, \nabla\eta_j)_{L^2(\Omega)},$$
$$M_{ii} = (\eta_i, 1)_{L^2(\Omega)},$$

where M is the lumped mass matrix and the finite dimensional representations of c and w

$$\mathbf{c} = \sum_{i=1}^{N} c(x_i)\eta_i, \qquad \mathbf{w} = \sum_{i=1}^{N} w(x_i)\eta_i,$$

the resulting algebraic problem can be written as: for each time-step given a previous solution $\{\mathbf{c}^{n-1},\mathbf{w}^{n-1}\}$ solve for $\{\mathbf{c}^n,\mathbf{w}^n\}$

$$M\mathbf{c}^n - M\mathbf{c}^{n-1} + \tau C\mathbf{w}^n = 0, \tag{13a}$$

$$\epsilon A\mathbf{c}^n + \frac{1}{\epsilon}M\boldsymbol{\phi}(\mathbf{c}^n) - \frac{1}{\epsilon}M\mathbf{c}^{n-1} - M\mathbf{w}^n = 0, \tag{13b}$$

with $\mathbf{c}^0$ an interpolation of the initial data and $\boldsymbol{\phi}$ is understood to operate pointwise.

3. The multigrid solver

We decompose the finite dimensional space:

$$V = \sum_{l=1}^{L} V_l, \tag{14}$$

where the $V_l \subset V_{l+1}$ are obtained by successive (not necessarily uniform) refinements of the triangulation, see Fig. 2.

We then solve the equations on each of these subspaces using a successive subspace refinement method (see [25]), which involves solving a non-linear Gauss–Seidel iteration on each subspace. Each 'coarse' grid basis function is represented on the fine grid using standard prolongation operators (e.g., [24]); Fig. 3.

For each basis function on each level we use the following algorithm: Given a level $l \in \{1,\ldots,L\}$, a node $k \in \{1,\ldots,N_l\}$, and the basis function for this node on this level $\eta_l{}^k$, update $\mathbf{c}$ and $\mathbf{w}$ by

$$\mathbf{c}^{k+1} = \mathbf{c}^k + \alpha\boldsymbol{\eta}_l^k, \mathbf{w}^{k+1} = \mathbf{w}^k + \beta\boldsymbol{\eta}_l^k,$$

(here we have dropped the time-step superscript for notational convenience) where α, β are the solution of

$$(M(\mathbf{c}^k + \alpha\boldsymbol{\eta}_l^k), \boldsymbol{\eta}_l^k) + \tau(C(\mathbf{w}^k + \beta\boldsymbol{\eta}_l^k), \boldsymbol{\eta}_l^k) - (\mathbf{b}, \boldsymbol{\eta}_l^k) = 0, \tag{15a}$$

$$\epsilon(A(\mathbf{c}^k + \alpha\boldsymbol{\eta}_l^k), \boldsymbol{\eta}_l^k) + \left(\frac{1}{\epsilon}M\boldsymbol{\phi}(\mathbf{c}^k + \alpha\boldsymbol{\eta}_l^k) - \frac{1}{\epsilon}\mathbf{b} - M(\mathbf{w}^k + \beta\boldsymbol{\eta}_l^k), \boldsymbol{\eta}_l^k\right) = 0, \tag{15b}$$

with $\mathbf{b} = M\mathbf{c}^{n-1}$ and $(\mathbf{x},\mathbf{y}) = \mathbf{x}^{\mathrm{T}}\mathbf{y}$ the Euclidean inner product. This is equivalent to solving

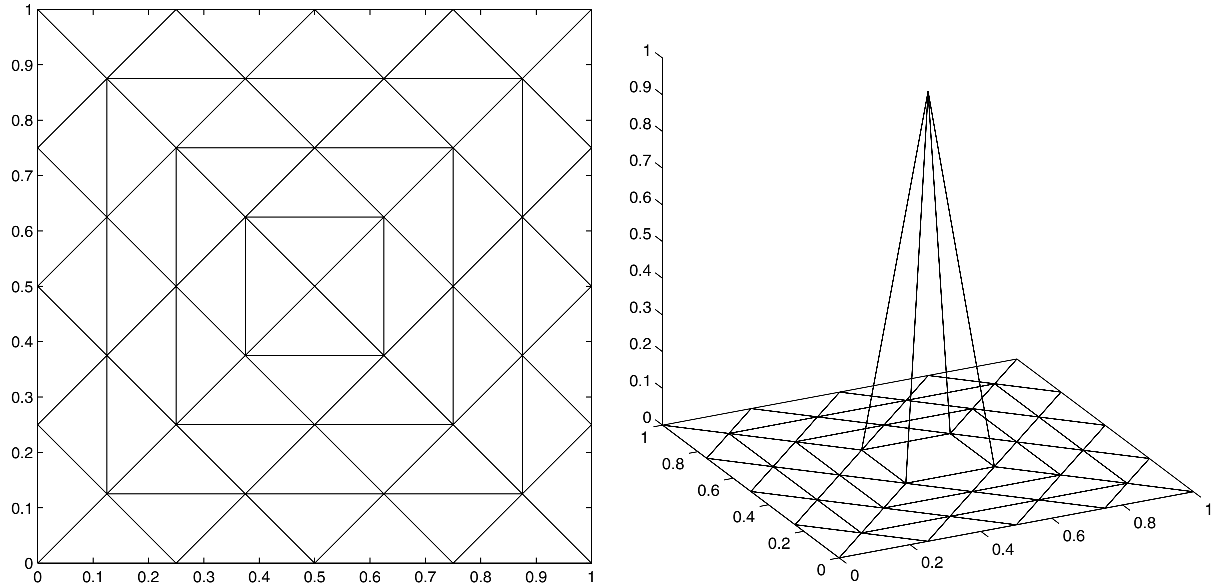


Fig. 1. A mesh and a typical basis function.

$$a_3\alpha^3 + a_2\alpha^2 + a_1\alpha + a_0 = 0, \tag{16a}$$

$$\beta = \frac{(\mathbf{b} - M\mathbf{c}^k - \tau C\mathbf{w}^k - \alpha M\boldsymbol{\eta}_l^k, \boldsymbol{\eta}_l^k)}{\tau(C\boldsymbol{\eta}_l^k, \boldsymbol{\eta}_l^k)}, \tag{16b}$$

where

$$\begin{aligned}
a_3 &= \frac{1}{\epsilon}(M\boldsymbol{\eta}_l^k, \boldsymbol{\eta}_l^k),\\
a_2 &= \frac{1}{\epsilon}(M\mathbf{c}^k(\boldsymbol{\eta}_l^k)^2, \boldsymbol{\eta}_l^k),\\
a_1 &= \left(\epsilon A\boldsymbol{\eta}_l^k + \frac{1}{\epsilon}M(\mathbf{c}^k)^2\boldsymbol{\eta}_l^k + \frac{(M\boldsymbol{\eta}_l^k, \boldsymbol{\eta}_l^k)}{\tau(C\boldsymbol{\eta}_l^k, \boldsymbol{\eta}_l^k)}M\boldsymbol{\eta}_l^k, \boldsymbol{\eta}_l^k\right),\\
a_0 &= \left(-\frac{1}{\epsilon}\mathbf{b} + \epsilon A\mathbf{c}^k - M\mathbf{w}^k - \left[\frac{(\mathbf{b} - M\mathbf{c}^k - \tau C\mathbf{w}^k, \boldsymbol{\eta}_l^k)}{\tau(C\boldsymbol{\eta}_l^k, \boldsymbol{\eta}_l^k)}\right]M\boldsymbol{\eta}_l^k, \boldsymbol{\eta}_l^k\right).
\end{aligned}$$

The non-linear equation (16a) is solved using a secant iteration, in experiments typically 2–5 iterations were required. The algorithm loops over all the basis functions on all the levels and is deemed to have converged when the relative residual of the linear equation (13a) has been reduced by a specific tolerance, tol.

In the case $V = V_L$, we recover the classical Gauss–Seidel method. Proof of convergence for this case is a simple extension of the method used in [3]. It is well known that Gauss–Seidel is an O(N) solver per iteration, that is the work required per iteration is proportional to the number of nodes of the triangulation. The number of iterations required for convergence is also O(N), making the total work required O(N^2). The multigrid scheme is also O(N) per iteration because of the nesting of the subspaces. However, if the rate of convergence is independent of the number of nodes, the total work is O(N). This means the multigrid algorithm will be a much more efficient method of solving the Cahn–Hilliard equation, especially on complicated problems when many nodes are required or when combined with adaptive mesh techniques.

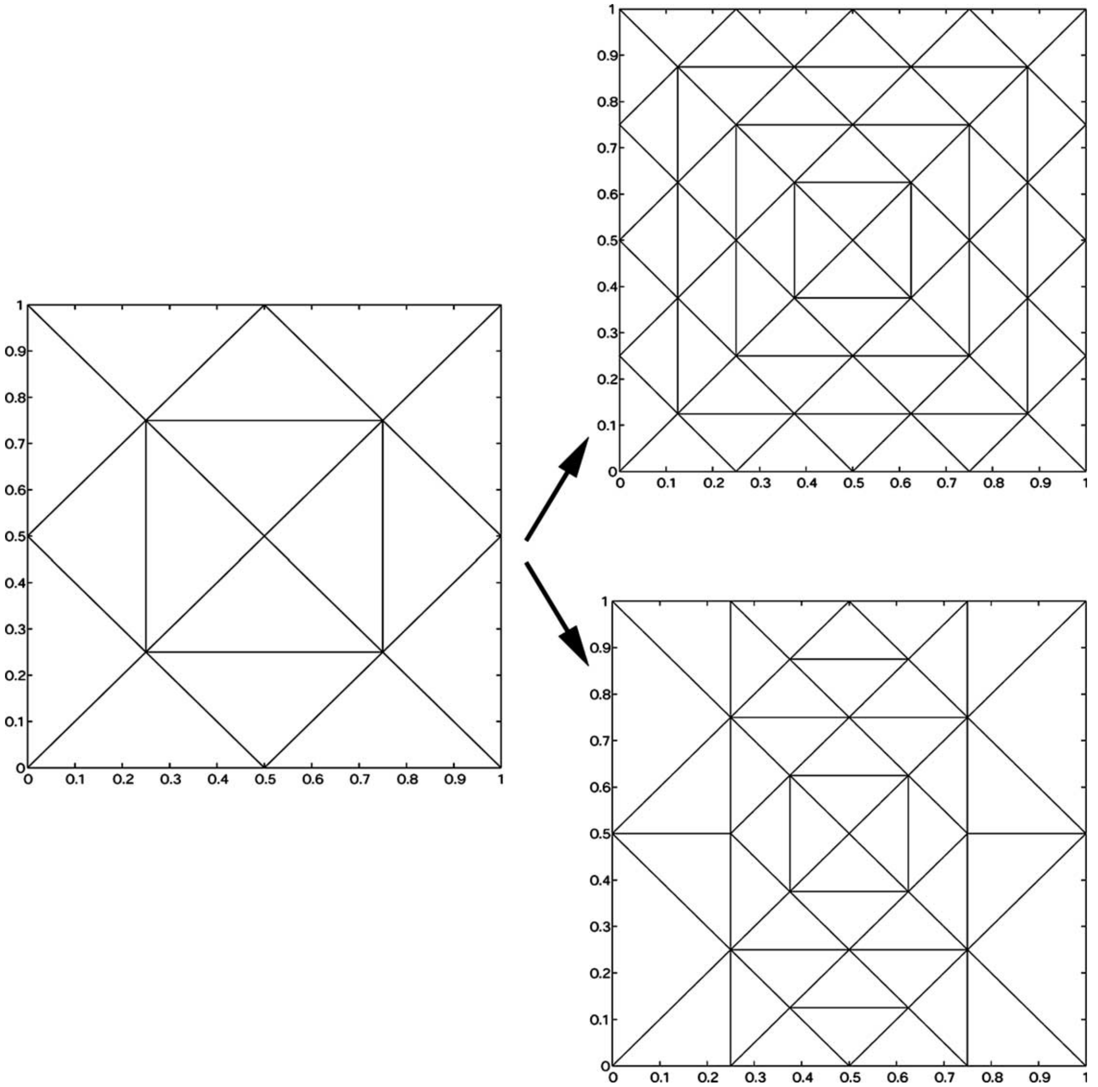


Fig. 2. Possible refinements of the mesh.

4. Numerical results – constant mobility

All problems are posed on a triangulation of the domain $\Omega := [0,1]^d$, $d = 1,2$. In this section, we consider the constant mobility case ($B = 1$).

4.1. Convergence rates

To illustrate the superiority of the multigrid solver, we present the comparative convergence rates of the two methods for a single time-step of spinodal decomposition, the initial data is a random perturbation of magnitude 0.1 about a mean composition of 0 with $\epsilon = 0.05$ and $\tau = 0.001$. From Fig. 4, it is clear that the

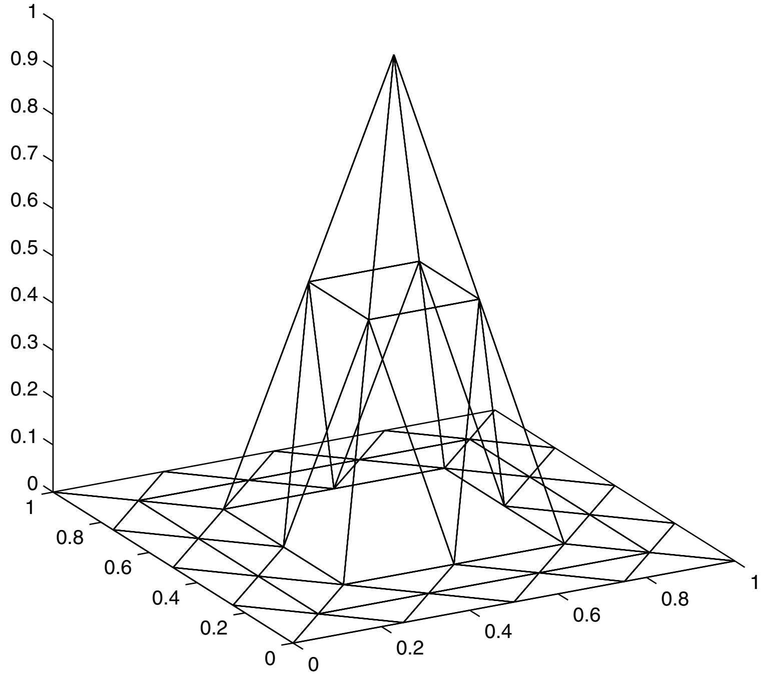


Fig. 3. A coarse grid basis function on the fine grid.

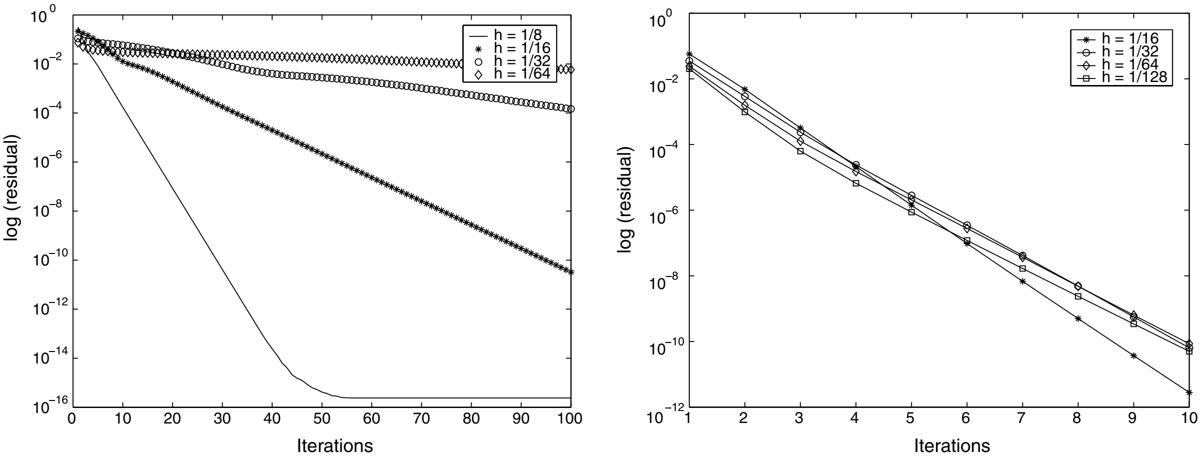


Fig. 4. Convergence rates for Gauss–Seidel (left) and multigrid (right).

multigrid algorithm converges much faster than the Gauss–Seidel algorithm and since the work required for an iteration of each is O(n) the multigrid algorithm provides a substantial decrease in the time required to solve a problem. In particular, we draw the readers attention to the scales used on the x-axes.

The remainder of the examples presented here use a solver tolerence of $\mathrm{tol} = 10^{-6}$.

4.2. One dimensional example

In many applications, we are either interested in the limit $\epsilon \to 0$ (c.f. [15,16]) or in very small ϵ. Therefore, we are interested in the behaviour of the solver in this limit. This requires small mesh size h which results in large matrix systems to be solved.

Consider the problem with $c_0 = \cos(\pi x), \tau = 0.001$; Tables 1 and 2 illustrate the maximum number of iterations required for a time-step. We see that the performance of the Gauss–Seidel solver deteriorates as ϵ decreases while the multigrid does not initially see this effect. If we further reduce ϵ then the iteration count for the multigrid solver increases but the rate of convergence is still h-independent, Table 3, i.e., all we are experiencing is a shift in the constant maximum number of iterations per time-step.

We note that it is only when ϵ is small relative to τ that the this happens. For accuracy reasons we would like $\tau \sim h$, and hence $\tau \sim \epsilon/10$ since we wish to have around 10 elements in the interface so that it is properly resolved. If we reduce τ by the same rate as ϵ or using a smaller time-step yields ϵ-independent results, Tables 4 and 5.

4.3. Two dimensional examples

4.3.1. Spinodal decomposition

Consider the case of spinodal decomposition in two spatial dimensions: the initial data is a random perturbation of magnitude 0.1 about a mean composition of 0. Tables 6 and 7 illustrate the maximum iteration count per time-step for the early stages.

Table 1
Gauss–Seidel – $\tau = 0.001$ fixed

h	ϵ		
	0.08	0.04	0.02
1/32	188	318	830
1/64	714	1217	1720
1/128	2790	4774	6753

Table 2
Multigrid – $\tau = 0.001$ fixed

h	ϵ			
	0.08	0.04	0.02	0.01
1/32	8	8	8	8
1/64	8	8	8	8
1/128	8	8	8	8
1/256	7	8	8	8

Table 3
Multigrid – $\tau = 0.001$ fixed

h	ϵ			
	0.02	0.01	0.005	0.0025
1/256	8	8	15	29
1/512	8	8	15	30
1/1024	8	8	16	29
1/2048	8	8	15	29

Table 4
Multigrid – $\tau = \epsilon/10$

h	ϵ			
	0.02	0.01	0.005	0.0025
1/256	8	8	8	8
1/512	8	8	8	8
1/1024	8	8	9	9
1/2048	8	8	8	9

Table 5
Multigrid – $\tau = 0.00025$ fixed

h	ϵ			
	0.02	0.01	0.005	0.0025
1/256	8	8	8	9
1/512	8	8	7	9
1/1024	8	8	7	9
1/2048	8	8	8	9

Table 6
Spinodal decomposition – $\tau = 0.005$

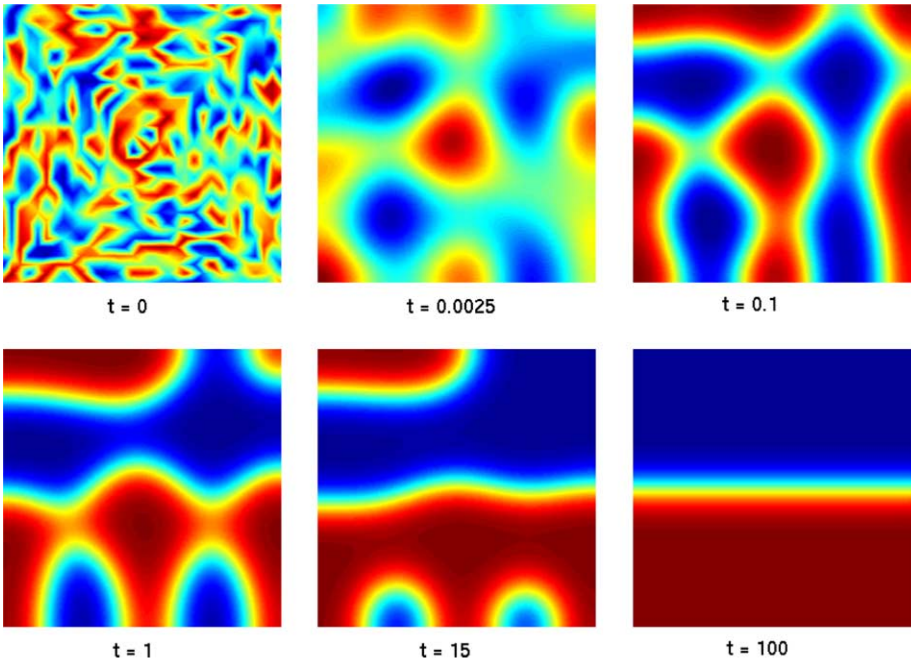
h	ϵ		
	0.1	0.07	0.05
1/16	6	6	5
1/32	7	7	6
1/64	7	7	7
1/128	7	7	7

Table 7
Spinodal decomposition – $\tau = 0.001$

h	ϵ		
	0.1	0.07	0.05
1/16	6	5	5
1/32	6	6	6
1/64	7	7	7
1/128	7	7	7

Again we see that the convergence is independent of the mesh size. Fig. 5 shows the evolution of the solution at six different times. The solution rapidly separates into regions of $c = \pm 1$ and then over much longer time-scales the regions evolve until we are left with an interface with minimal measure (c.f. Section 4.3.2). This is an example of the classical behaviour of the Cahn–Hilliard equation.

As discussed above, we are interested in retaining the Lyapunov qualities of the energy functional. The finite element discretisation and gradient stability of this algorithm ensures that this is the case. Most importantly the discrete total energy of the system decreases monotonically, Fig. 6.



t = 0

t = 0.0025

t = 0.1

t = 1

t = 15

t = 100

Fig. 5. Spinodal decomposition, $\epsilon = 0.05$, $\tau = 0.001$, $h = 1/128$.

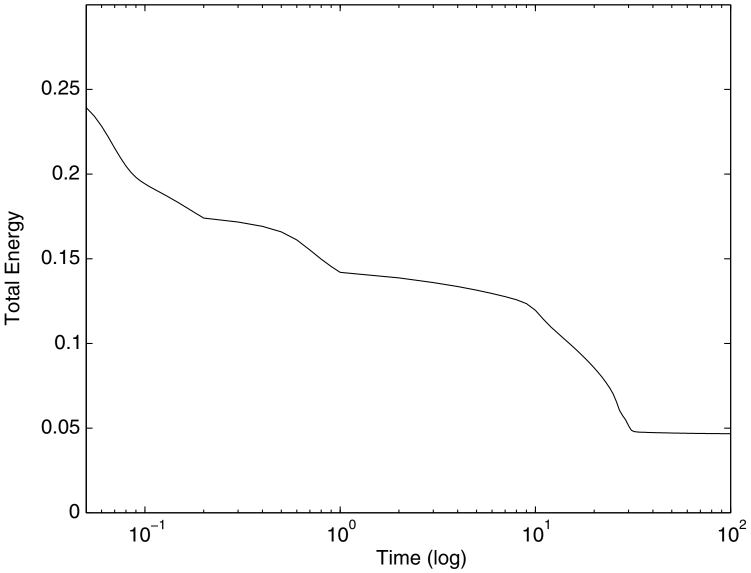
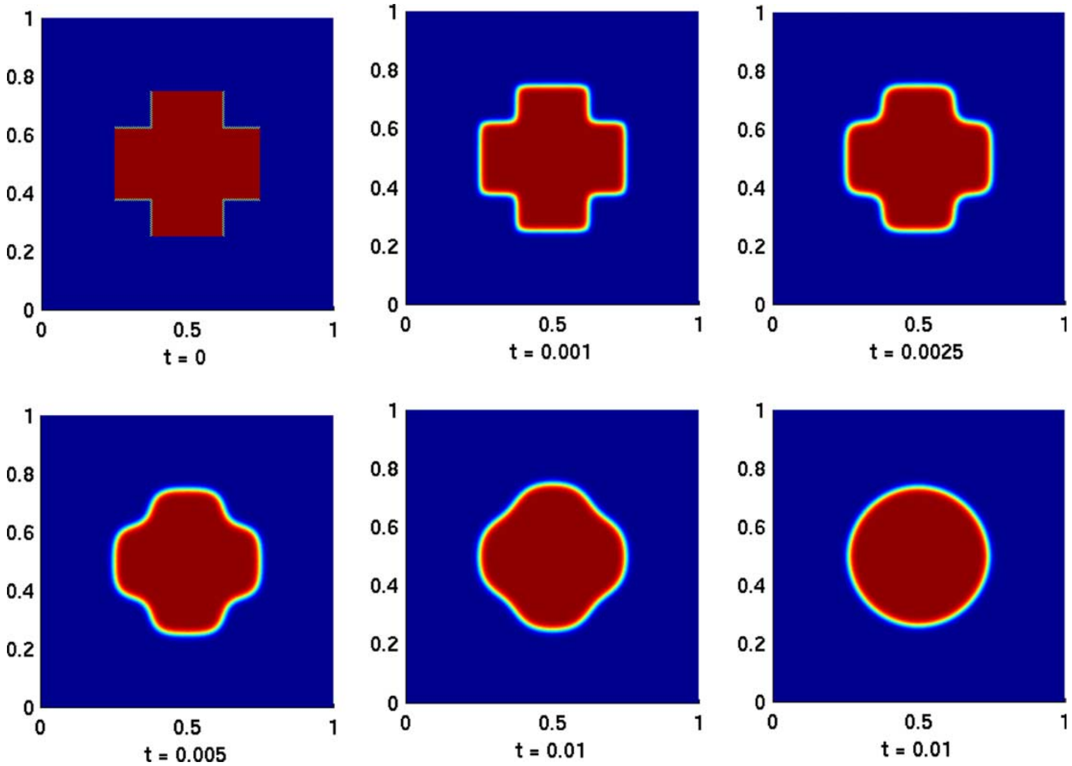


Fig. 6. Total energy of the system: $J(c)$.

4.3.2. Minimisation of the measure of the interface

It is known that the Cahn–Hilliard equation minimises the measure of the interface [21]. Thus, we expect steady-state solutions to have interfaces that are either straight lines which divide the domain (as in the



t = 0

t = 0.001

t = 0.0025

t = 0.005

t = 0.01

t = 0.01

Fig. 7. Evolution of a cross – $\epsilon = 0.01$, $h = 1/128$, $\tau = 10^{-4}$.

example above) or arcs of circles. For example a cross degenerates into a circle, see Fig. 7. This took a maximum of 7 iterations per time-step.

5. Numerical results – degenerate mobility

In this section, we consider the equation with the degenerate mobility function $B(c) = [1 - c^2]_+$ and compare the results with the constant mobility case, showing that the degeneracy does not affect the solver.

5.1. One dimensional example

Take initial data [2]:

$$c_0(\mathbf{x}) = \begin{cases} 1 & \text{if } 0 \leqslant x \leqslant \frac{1}{3} - \frac{1}{20}, \\ 20(\frac{1}{3} - x) & \text{if } |x - \frac{1}{3}| \leqslant \frac{1}{20}, \\ -20|x - \frac{41}{50}| & \text{if } |x - \frac{41}{50}| \leqslant \frac{1}{20}, \\ 0 & \text{otherwise} \end{cases} \tag{17}$$

with $\epsilon = 0.03$, $h = 1/256$, $\tau = 5 \times 10^{-4}$. If we compare both types of mobility for this problem we see that they have the same final solution but are significantly different during their evolution, Figs. 8 and 9.

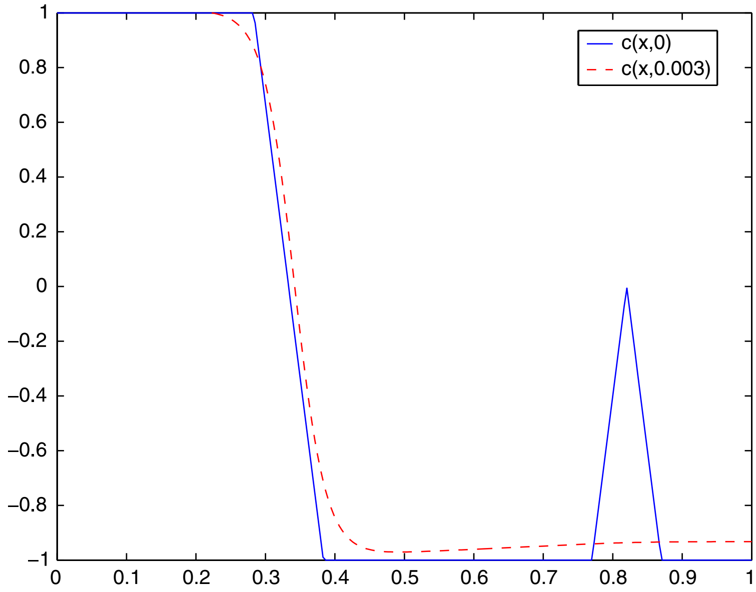


Fig. 8. Constant mobility.

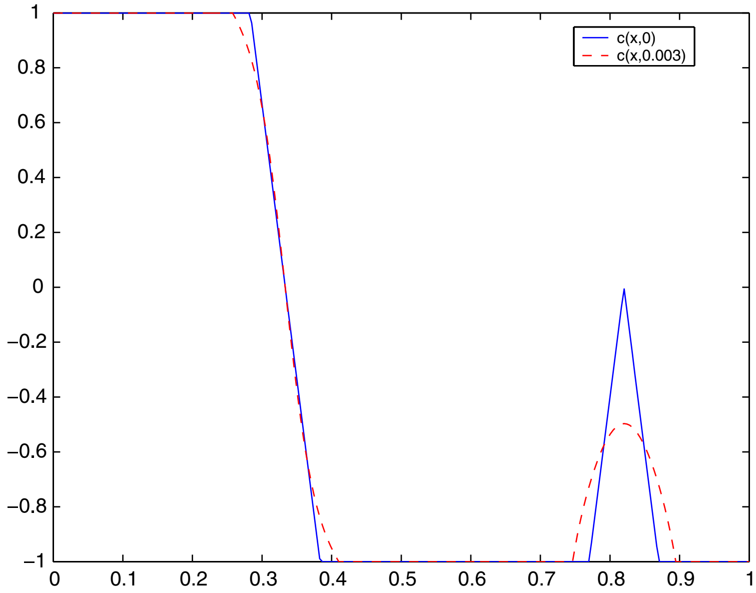


Fig. 9. Degenerate mobility.

For constant mobility the solution diffuses throughout the whole of the domain causing the spreading of the solution around the spike. The degenerate mobility limits the diffusion in the bulk regions so most of the diffusion is through the interface, so the spike only diffuses through its interfaces, not through the whole domain. See Table 8 for the maximum iteration count of the degenerate mobility case.

Table 8
Degenerate mobility

h	ϵ			
	0.06	0.03	0.015	0.0025
1/32	7	6	6	6
1/64	8	8	7	9
1/128	9	8	8	8
1/256	9	8	8	8

Table 9
Spinodal decomposition, degenerate mobility – $\tau = 0.001$

h	ϵ		
	0.01	0.07	0.05
1/16	6	6	6
1/32	7	7	7
1/64	7	7	7
1/128	7	7	7

5.2. Two dimensional example

Consider the spinodal decomposition example from Section 4.3.1 but now with degenerate mobility; the following maximum iteration counts were achieved, Table 9. From these examples it is clear that the solver is also optimal (mesh independent convergence) in the degenerate mobility case.

6. Adaptive time-stepping

One of the difficulties in performing numerical simulations of the Cahn–Hilliard equation is the differing time scales over which the solution evolves. The initial stage where the two species separate occurs very quickly while the evolution of the free boundaries to a steady state takes many magnitudes of time longer. An adaptive time-stepping algorithm allows accurate solutions with much less work required.

The sharp interface limit (i.e., $\epsilon \to 0$) of the Cahn–Hilliard equation is motion of the interface via a Mullins–Sekerka law for constant mobility and motion by surface diffusion (Laplacian of the mean curvature) for the degenerate mobility, [1,9,22]. Mullins–Sekerka flow is scale invariant under a scaling $x \to \lambda x$, $t \to \lambda^3 t$ and surface diffusion under $x \to \lambda x$, $t \to \lambda^4 t$, so the natural length scale of the solution should evolve like $\ell(t) \sim t^{1/3}$ or as $\ell(t) \sim t^{1/4}$, respectively as $t \to \infty$. Kohn and Otto [19] proved that in a time averaged sense this was in fact the maximum rate that evolution could occur for the Cahn–Hilliard equation. This provides us with a 'worse case' scenario of how much we expect the time-step to grow in an adaptive regime. Following the scheme described in [23] we find that the mesh independent convergence rate for our solver is retained. For the spinodal decomposition example above with this time-stepping algorithm we obtain the following maximum iteration counts, Tables 10 and 11.

The robustness of the solver is also reproduced when a simple first order predictor–corrector scheme is used. From this, we conclude that our algorithm can be used in conjunction with an adaptive time-stepping regime.

Table 10
Multigrid – $\tau = 0.01t^{1/3}$, constant mobility

h	ϵ		
	0.1	0.07	0.05
1/16	6	6	6
1/32	7	7	7
1/64	7	7	7
1/128	7	7	7

Table 11
Multigrid – $\tau = 0.01t^{1/4}$, degenerate mobility

h	ϵ		
	0.1	0.07	0.05
1/16	6	6	6
1/32	7	7	7
1/64	7	7	7
1/128	7	7	7

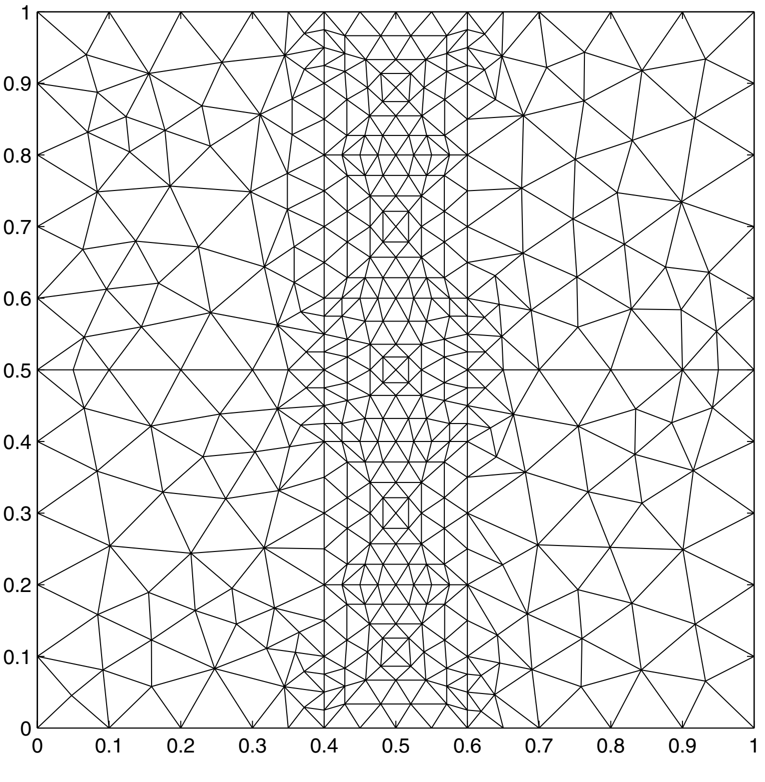


Fig. 10. Non-uniform mesh.

7. Non-uniform meshes

The need for high spacial resolution in the interface means that with uniform mesh refinement the matrix problem grows very large. However we only require large numbers of elements in the interface, not in the bulk region where the solution is constant. Thus, in order to further improve the efficiency of the solver we would like to couple it to an adaptive mesh strategy. Here, we present results that indicate such a coupling would retain the optimality of the multigrid solver.

We consider the case of $c_0 = \cos(\pi x)$ so that the interface will be contained in a central strip of the domain. An example of the triangulation that we use for this problem is given in Fig. 10. Table 12 shows the maximum iteration counts for this problem with $\tau = 10^{-4}$.

Table 12
Non-uniform mesh

$h_{\text{interface}}$	ϵ		
	0.08	0.04	0.02
0.03	8	6	7
0.015	8	5	7
0.0075	7	5	6
0.0037	6	5	5

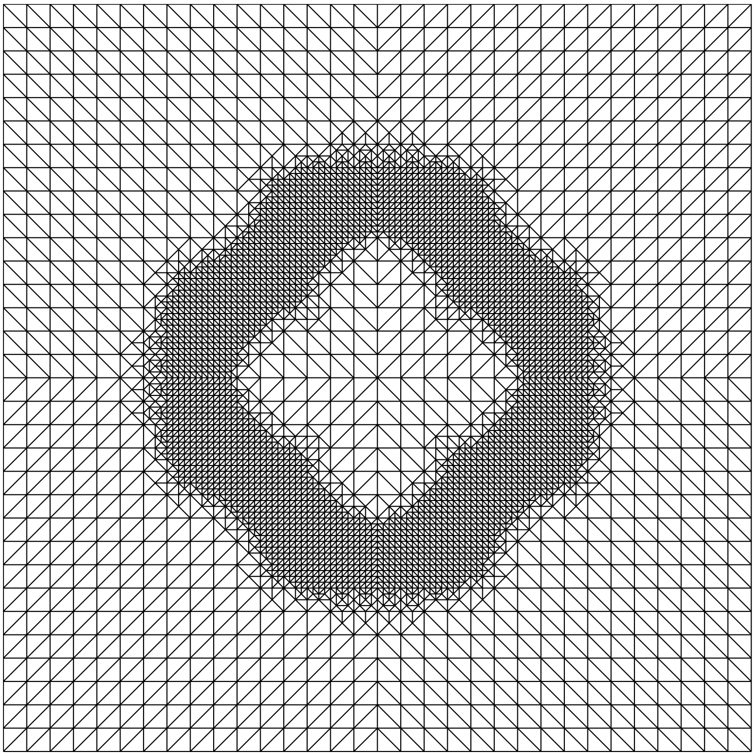


Fig. 11. Non-uniform mesh.

Table 13
Non-uniform mesh – cross

Level	6	7	8
Count	12	12	12

Now consider the example of a cross given earlier. At a given time-step an adaptive mesh based on a strategy of refinement in the interfacial region would look like Fig. 11.

The following iteration counts were obtained with $\epsilon = 0.01$, $\tau = 0.001$, Table 13.

From these examples, we can see that the multigrid method is not affected by the non-uniformity of the grid. Additional experiments have shown that this is true as long as the triangles do not become too anisotropic. Thus, we conclude that the multigrid algorithm used in conjunction with an adaptive mesh strategy would still maintain mesh independent convergence rates.

8. Conclusions

We have presented a multigrid finite element solver for the Cahn–Hilliard equation that it efficient and robust with respect to both mesh refinement and time-step size. The limit $\epsilon \to 0$ has been investigated numerically. Numerical experiments indicate that the solver can be coupled to both variable time-stepping and adaptive mesh regimes to improve performance even further. We remark that this algorithm can also be applied to similar degenerate fourth order p.d.e. problems, for example the thin film equation.

Acknowledgement

The authors thank Charlie Elliott for his useful comments and discussions.

References

[1] N.D. Alikakos, P.W. Bates, Xinfu Chen, Convergence of the Cahn–Hilliard equation to the Hele–Shaw model, Arch. Rat. Mech. Anal. 128 (2) (1994) 165–205.

[2] J.W. Barrett, J.F. Blowey, H. Garcke, Finite element approximation of the Cahn–Hilliard equation with degenerate mobility, SIAM J. Numer. Anal. 37 (1998) 286–318.

[3] J.W. Barrett, R. Nürnberg, V. Styles, Finite element approximation of a void electromigration model, SIAM J. Numer. Anal. 42 (2004) 738–772.

[4] J.F. Blowey, C.M. Elliott, The Cahn–Hilliard gradient theory for phase separation with non-smooth free energy part II: numerical analysis, Eur. J. Appl. Math. 3 (1992) 147–179.

[5] F. Boyer, A theoretical and numerical model for the study of incompressible mixture flows, Comput. Fluids 31 (1) (2002) 41–68.

[6] J.W. Cahn, On spinodal decomposition, Acta Metall. Mater. 9 (1961) 795–801.

[7] J.W. Cahn, J.E. Hilliard, Free energy of a nonuniform system, I, interfacial free energy, J. Chem. Phys. 28 (2) (1958) 258–267.

[8] J.W. Cahn, J. Taylor, Linking anisotropic sharp and diffuse surface motion laws via gradient flows, J. Stat. Phys. 77 (1994) 183–197.

[9] Xinfu Chen, Global asymptotic limit of solutions of the Cahn–Hilliard equation, J. Differ. Geom. 44 (2) (1996) 262–311.

[10] C.M. Elliott, The Cahn–Hilliard Model for the Kinetics of Phase SeparationNumber 88 in International Series of Numerical Mathematics, Birkhäuser Verlag, Basel, Germany, 1989 (pp. 35–73).

[11] C.M. Elliott, D.A. French, Numerical studies of the Cahn–Hilliard equation for phase separation, IMA J. Appl. Math. 39 (1987) 97–128.

[12] C.M. Elliott, D.A. French, F.A. Milner, A second order splitting method for the Cahn–Hilliard equation, Numer. Math. 54 (1989) 575–590.

[13] C.M. Elliott, S. Larsson, Error estimates with smooth and nonsmooth data for a finite element method for the Cahn–Hilliard equation, Math. Comput. 58 (198) (1992) 603–630.

[14] C.M. Elliott, A.M. Stuart, The global dynamics of discrete semilinear parabolic equations, SIAM J. Numer. Anal. 30 (6) (1993) 1622–1663.

[15] Xiaobing Feng, Andreas Prohl, Numerical analysis of the Cahn–Hilliard equation and approximation for the Hele–Shaw problem, part I: error analysis under minimum regularities, IMA Preprint Series, vol. 1798, 2001.

[16] Xiaobing Feng, Andreas Prohl, Numerical analysis of the Cahn–Hilliard equation and approximation for the Hele–Shaw problem, part II: error analysis and convergence of the interface, IMA Preprint Series, vol. 1799, 2001.

[17] Junseok Kim, Modelling and Simulation of Multi-Component, Multi-Phase Fluid Flows. PhD thesis, University of Minnesota, 2002.

[18] Junseok Kim, Kyungkeun Kang, John Lowengrub, Conservative multigrid methods for Cahn–Hilliard fluids, J. Comp. Phys. 193 (2) (2004) 357–379.

[19] R.V. Kohn, F. Otto, Upper bounds on coarsening rates, Commun. Math. Phys. 229 (2002) 375–395.

[20] R. Kornhuber, R. Krause, On multigrid methods for vector-valued Allen–Cahn equations with obstacle potential, in: I. Herrera, D.E. Keyes, O.B. Widlund, R. Yates (Eds.), Fourteenth International Conference on Domain Decomposition Methods, 2002.

[21] L. Modica, The gradient theory of phase transitions and the minimal interface criterion, Arch. Rat. Mech. Anal. 98 (1987) 123–142.

[22] R.L. Pego, Front migration in the nonlinear Cahn–Hilliard equation, Proc. R. Soc. Lond. A 422 (1989) 261–278.

[23] B.P. Vollmayr-Lee, A.D. Rutenberg, Fast and accurate coarsening simulation with an unconditionally stable time step, Phys. Rev. E 68 (2003).

[24] P. Wesseling, An Introduction to Multigrid Methods, Wiley, Chichester, England, 1992.

[25] Jinchao Xu, Iterative methods by space decomposition and subspace correction, SIAM Rev. 34 (4) (1992) 581–613.