AIMS Mathematics, 9(12): 34447–34465.
DOI: 10.3934/math.20241641
Received: 08 October 2024
Revised: 27 November 2024
Accepted: 04 December 2024
Published: 06 December 2024

https://www.aimspress.com/journal/Math

Research article

An explicit numerical method for the conservative Allen–Cahn equation on a cubic surface

Youngjin Hwang[1], Jyoti[2], Soobin Kwak[1], Hyundong Kim[3,4] and Junseok Kim[1,*]

[1] Department of Mathematics, Korea University, Seoul, 02841, Republic of Korea

[2] The Institute of Basic Science, Korea University, Seoul, 02841, Republic of Korea

[3] Department of Mathematics and Physics, Gangneung-Wonju National University, Gangneung 25457, Republic of Korea

[4] Institute for Smart Infrastructure, Gangneung-Wonju National University, Gangneung 25457, Republic of Korea

* **Correspondence:** Email: cfdkim@korea.ac.kr.

Abstract: We introduced a fully explicit finite difference method (FDM) designed for numerically solving the conservative Allen–Cahn equation (CAC) on a cubic surface. In this context, the cubic surface refers to the combined areas of the six square faces that enclose the volume of a cube. The proposed numerical solution approach is structured into two sequential steps. First, the Allen–Cahn (AC) equation was solved by applying the fully explicit FDM, which is computationally efficient. Following this, the conservation term is resolved using the updated solution from the AC equation to ensure consistency with the underlying conservation principles. To evaluate the effectiveness of the proposed scheme, computational tests are performed to verify that the resulting numerical solution of the CAC equation successfully conserves the discrete mass. Additionally, the solution is examined for its ability to exhibit the property of constrained motion by mass conserving mean curvature, a critical characteristic of the CAC equation. These two properties are fundamental to the integrity and accuracy of the CAC equation.

Keywords: finite difference scheme; cubic domain; Lagrange multiplier; explicit scheme; conservative AC equation
Mathematics Subject Classification:80M20, 39A14, 65M06

1. Introduction

The classic Allen–Cahn (AC) equation [1] is a prototypical example of gradient flow associated with a functional:

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

where $\Omega \subset \mathbb{R}^d$, $u(\mathbf{x}, t)$ is an order parameter at spatial point $\mathbf{x}$ and time t, ϵ represents the interface width parameter, and $F(u) = 0.25(u^2 - 1)^2$. Thus, the AC equation can be derived through a variational approach as $\partial u/\partial t = -\delta\mathcal{E}/\delta u$, which can be expressed as follows.

$$\frac{\partial u(\mathbf{x}, t)}{\partial t} = -\frac{F'(u(\mathbf{x}, t))}{\epsilon^2} + \Delta u(\mathbf{x}, t). \tag{1.1}$$

The AC equation is a fundamental phase-field model in a wide range of applications such as materials science [2–4], physics [5], classification [6, 7], and image processing [8, 9]. Its versatility lies in its ability to describe complex phenomena such as the evolution of phase transitions, where distinct phases within a material change over time, and the dynamics of interfaces that separate these phases. These characteristics make the AC equation an essential tool in understanding interface motion and phase separation in various scientific and engineering contexts [10–13]. The equation captures critical aspects of interface dynamics, particularly the behavior of systems as they evolve toward a minimal energy state, driven by forces such as curvature. This makes it a key model for studying processes where the morphology of interfaces plays a central role. The conventional AC equation is not conservative but preserves the maximum principle. Recently, several studies have focused on the maximum-principle-preserving characteristic of the AC equation. Zhang et al. [14] developed and analyzed the maximum principle preserving method for solving the AC equation. Sun et al. [15] presented a class of up to temporally eighth-order maximum principle preserving method for the AC equation. Choi et al. [16] researched stability analysis and maximum principle property of the fully explicit finite difference scheme for the high-order AC equation. Kim et al. [17] presented a hybrid numerical method on nonuniform grids for solving the AC equation.

However, in the situations where mass conservation is critical, a conservative variant of the AC equation is used, which ensures that the total quantity of the evolving order parameter remains constant over time [18]. To make the AC equation conservative in nature, several different Lagrange multipliers have been proposed to satisfy the mass conservation law for the AC equation. For example, Rubinstein and Sternberg [19] introduced a nonlocal space-independent Lagrange multiplier into the AC equation. Brassel and Bretin [20] proposed a local-nonlocal space-time dependent Lagrange multiplier for mass conservation of the AC equation. The two conservative forms of the AC equation have received significant attention, and the conservative AC (CAC) equation satisfies mass conservation while effectively modeling the dynamics of interfaces. Furthermore, compared to the fourth-order Cahn–Hilliard equation, the CAC equation is a second-order partial differential equation (PDE), which makes it simpler to solve [21]. These advantages have motivated a wide range of theoretical and numerical studies on the CAC equation [22–24]. Xia et al. [25] proposed temporally second-order unconditionally stable direct schemes for the AC and CAC equations on surfaces. They found that the proposed method is unconditionally energy-stable. Sun and Zhang [26] developed a meshless radial basis function

approximation method for the CAC equation on surfaces based on an operator splitting scheme. The developed numerical method satisfies the mass conservation law of the CAC equation. The authors demonstrated the accuracy and mass conservation law of the developed method through various numerical experiments on different smooth and compact surfaces. Liu et al. [27] presented multi-physical structure-preserving method for the CAC equation. Yang and Kim [28] developed a numerical method for the conservative Allen–Cahn–Navier–Stokes model on arbitrarily curved surfaces in three-dimensional (3D) space. Previously, the authors proposed a numerical solution for the Cahn–Hilliard–Navier–Stokes system on arbitrarily curved surfaces in 3D space [29]. However, simulating mass-conserving binary flow on such surfaces can be solved more efficiently and straightforwardly using the CAC equation instead of the Cahn–Hilliard equation [30]. This is because the Cahn–Hilliard equation is a fourth-order PDE, whereas the conservative Allen–Cahn equation is second-order, which results in reduced computational cost and complexity. Figure 1 shows the solution obtained using the numerical method developed by Yang and Kim [28] for the conservative Allen–Cahn–Navier–Stokes model. The our proposed method focuses on efficiently and rapidly computing the CAC equation in two-dimensionality (2D) on the surface of a cuboid that approximates a spherical surface in 3D.

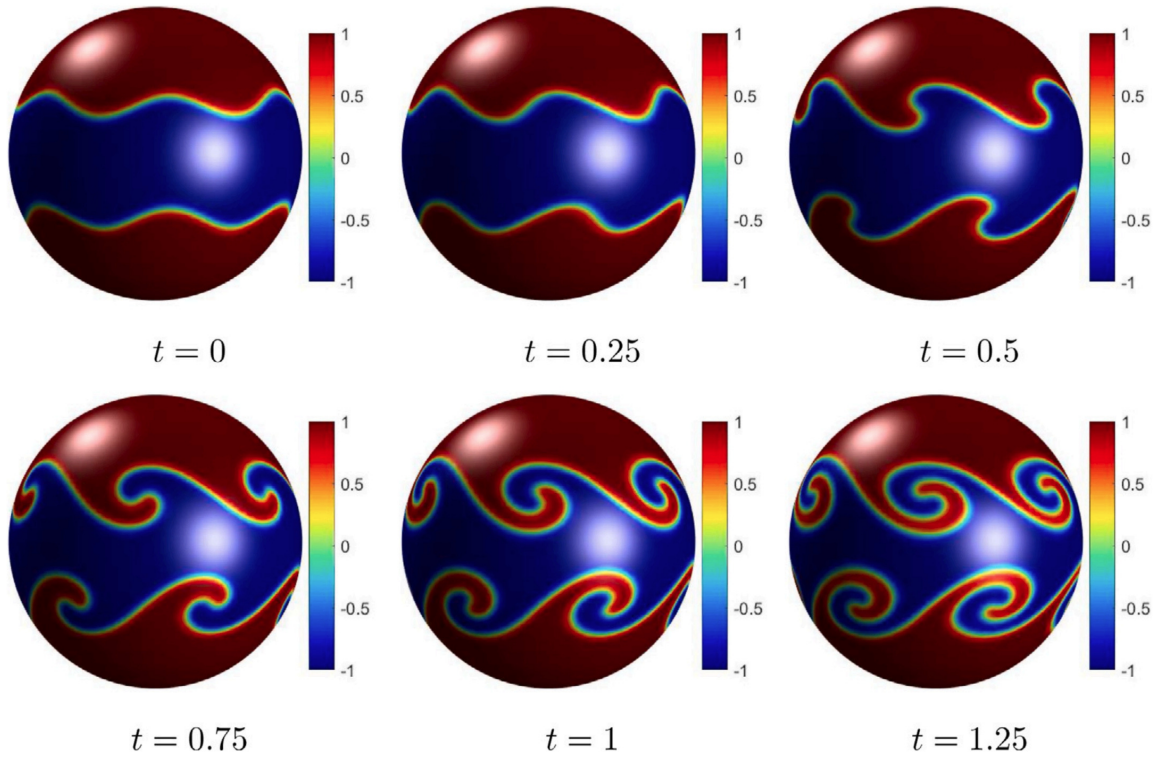


Figure 1. The snapshots of the computational solutions. Reprinted from [28] with permission from Elsevier.

The analysis of the numerical methods based on the finite difference method for the CAC equation is highly challenging. Consequently, various numerical methods have either replaced the analysis with numerical verification of properties such as the maximum principle, energy stability, and convergence or modified the CAC equation to facilitate analysis. Choi and Kim [31] presented a novel CAC equation with a new Lagrange multiplier that ensures strict preservation of the maximum principle and an unconditionally stable numerical method, which was validated through computational experiments.

Moreover, solving this equation on complex geometries, such as curved surfaces, presents challenges due to the need to accurately capture surface geometry while ensuring numerical stability and conservation properties. Kim et al. [32] developed the unconditionally stable hybrid scheme for the CAC equation with a space-time dependent Lagrange multiplier. The authors validated the numerical methods through various numerical experiments, such as practical unconditional stability tests.

In this paper, we focus on a simple and effective finite difference method (FDM) for solving the CAC equation on cubic surfaces. In other words, the key aspect is to consider the unique characteristics of cubic surfaces. The proposed method is simple to implement and can serve as a foundation for various methods such as implicit and scalar auxiliary variable approaches. The cubic surfaces, which are characterized by their sharp edges and non-smooth features, pose additional difficulties for numerical approximation methods [33]. Traditional schemes may struggle with maintaining mass conservation and interface accuracy in such settings, making the development of specialized algorithms critical. Therefore, the development of finite difference schemes designed for the CAC equation on surfaces such as cubic geometries is an area of growing interest.

The structure of this paper is as follows. Section 2 presents the governing CAC equation. Section 3 presents the proposed computational scheme. Section 4 presents numerical tests. In Section 6, conclusions are presented.

2. Governing equation

We consider the following CAC equation [20]:

$$\frac{\partial u(\mathbf{x},t)}{\partial t} = -\frac{F'(u(\mathbf{x},t))}{\epsilon^2} + \Delta u(\mathbf{x},t) + \beta(t)\sqrt{F(u(\mathbf{x},t))}, \quad \mathbf{x} \in \Omega, \ t > 0, \tag{2.1}$$

where Ω is a bounded domain in $\mathbb{R}^d$ ($d = 1, 2, 3$), $u(\mathbf{x}, t)$ is conserved order parameter, and $\beta(t)$ is given by

$$\beta(t) = \frac{\displaystyle\int_\Omega F'(u(\mathbf{x},t))d\mathbf{x}}{\epsilon^2 \displaystyle\int_\Omega \sqrt{F(u(\mathbf{x},t))}d\mathbf{x}}.$$

We define the unit cubic surface domain $\hat{\Omega}$ in 3D space as follows to solve the problem on the unit cubic surface.

$$\hat{\Omega} = \{(x,y,z)|(x,y,0),\ (x,0,z),\ (1,y,z),\ (x,1,z),\ (0,y,z),\ (x,y,1),\ 0 \le x,y,z \le 1\}. \tag{2.2}$$

Then, the cubic surface domain $\hat{\Omega}$ is decomposed into sub-domains $\hat{\Omega}_k$, $k = 1, 2, \ldots, 6$.

$$\begin{aligned}
\hat{\Omega}_1 &= \{(x,y,z) \mid 0 < x,y < 1, z = 0\},\ \hat{\Omega}_2 = \{(x,y,z) \mid 0 < x,z < 1, y = 0\},\\
\hat{\Omega}_3 &= \{(x,y,z) \mid 0 < y,z < 1, x = 1\},\ \hat{\Omega}_4 = \{(x,y,z) \mid 0 < x,z < 1, y = 1\},\\
\hat{\Omega}_5 &= \{(x,y,z) \mid 0 < y,z < 1, x = 0\},\ \hat{\Omega}_6 = \{(x,y,z) \mid 0 < x,y < 1, z = 1\},\\
\hat{\Omega} &= \cup_{k=1}^{6}\overline{\hat{\Omega}}_k,
\end{aligned}$$

where $\overline{\hat{\Omega}}_k$ denotes the closure of the set $\hat{\Omega}_k$ for $k = 1, \ldots, 6$. To define the unfolded cubic surface domain, we define the sub-domains in 2D space as follows.

$$\begin{aligned}
\Omega_1 &= \{(x, -y) \mid (x, y, 0) \in \hat{\Omega}_1\}, \qquad \Omega_2 = \{(x, z+1) \mid (x, 0, z) \in \hat{\Omega}_2\}, \\
\Omega_3 &= \{(y+1, z+1) \mid (1, y, z) \in \hat{\Omega}_3\}, \quad \Omega_4 = \{(3-x, z+1) \mid (x, 1, z) \in \hat{\Omega}_4\}, \\
\Omega_5 &= \{(4-y, z+1) \mid (0, y, z) \in \hat{\Omega}_5\}, \quad \Omega_6 = \{(x, y+2) \mid (x, y, 1) \in \hat{\Omega}_6\}.
\end{aligned}$$

Thus, the unfolded cubic surface domain is defined by $\Omega = \cup_{k=1}^{6} \overline{\Omega}_k$. Figure 2 displays the schematic illustration of the folded and unfolded cubic surface domains. We can observe that the proposed numerical scheme can be easily extended to surfaces of arbitrarily sized cuboids. Unless stated otherwise, we consider the unit cubic surface.

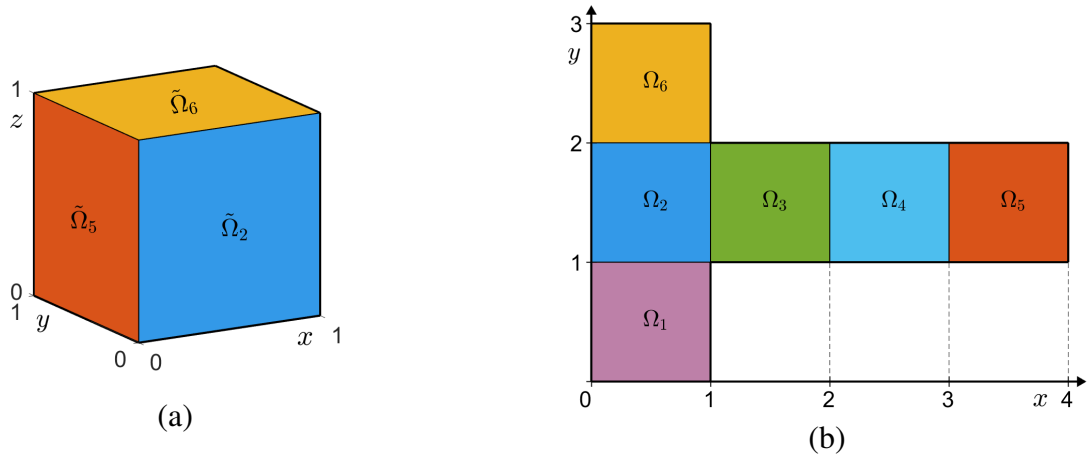


Figure 2. Schematic illustrations of (a) the folded and (b) unfolded cubic surface domains.

We consider the folded cubic surface in 3D space and apply proper boundary conditions to each sub-domain Ω_k for $k = 1, 2, \ldots, 6$. For example, we focus a sub-domain Ω_4, and the boundary conditions are shown Figure 3(a).

Similarly, we apply the boundary condition for the other sub-domain Ω_k, $k = 1, 2, 3, 5, 6$ considering the folded cubic surface domain. To prove the mass conservation law of the CAC equation on a cubic surface, we define the total mass, denoted by $\mathcal{M}(u)$, as

$$\mathcal{M}(u) = \int_{\Omega} \frac{u(\mathbf{x}, t) + 1}{2} d\mathbf{x}, \tag{2.3}$$

which can be written as

$$\int_{\Omega} u(\mathbf{x}, t) d\mathbf{x} = 2\mathcal{M}(u) - \int_{\Omega} 1 d\mathbf{x}. \tag{2.4}$$

Therefore, it is equivalent to state that $\int_{\Omega} u(\mathbf{x}, t) d\mathbf{x}$ is constant, which implies that $\mathcal{M}(u)$ is constant. Taking the time derivative of Eq (2.3), we have

$$\frac{d\mathcal{M}}{dt} = \frac{1}{2} \int_{\Omega} \frac{\partial u(\mathbf{x}, t)}{\partial t} d\mathbf{x} = \frac{1}{2} \int_{\Omega} \left[-\frac{F'(u(\mathbf{x}, t))}{\epsilon^2} + \Delta u(\mathbf{x}, t) + \beta(t) \sqrt{F(u(\mathbf{x}, t))} \right] d\mathbf{x}$$

$$= \frac{1}{2}\int_{\Omega} \Delta u(\mathbf{x}, t) d\mathbf{x}. \tag{2.5}$$

Considering that the unfolded domain Ω consists of a combination of six sub-domains Ω_k, $k = 1, 2, \ldots, 6$, we obtain the following equation:

$$\int_{\Omega} \Delta u(\mathbf{x}, t) d\mathbf{x} = \sum_{k=1}^{6} \int_{\Omega_k} \Delta u(\mathbf{x}, t) d\mathbf{x}.$$

By using the equation above and Green's theorem, we can rewrite Eq (2.5) as follows.

$$\frac{1}{2}\int_{\Omega} \Delta u(\mathbf{x}, t) d\mathbf{x} = \frac{1}{2}\sum_{k=1}^{6} \int_{\Omega_k} \Delta u(\mathbf{x}, t) d\mathbf{x} = \frac{1}{2}\sum_{k=1}^{6} \int_{\partial\Omega_k} \nabla u(\mathbf{x}, t) \cdot \mathbf{n} ds, \tag{2.6}$$

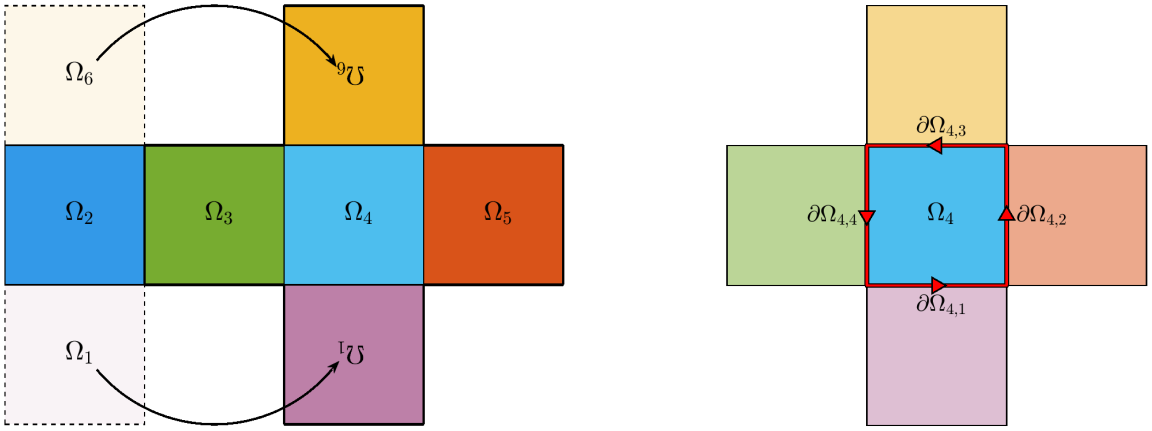
where $\partial\Omega_k$, $k = 1, 2, \ldots, 6$ are curves oriented in the positive direction. We calculate Eq (2.6) by breaking up $\partial\Omega_k$ for $k = 1, 2, \ldots, 6$ as the union of the four curves $\partial\Omega_{k,1}$, $\partial\Omega_{k,2}$, $\partial\Omega_{k,3}$, and $\partial\Omega_{k,4}$, as shown for Ω_4 in Figure 3(b).

$$\frac{1}{2}\sum_{k=1}^{6} \int_{\partial\Omega_k} \nabla u(\mathbf{x}, t) \cdot \mathbf{n} ds \quad = \quad \frac{1}{2}\sum_{k=1}^{6}\sum_{c=1}^{4} \int_{\partial\Omega_{k,c}} \nabla u(\mathbf{x}, t) \cdot \mathbf{n} ds.$$

For each $\partial\Omega_{k,c}$, $k = 1, \ldots, 6$, $c = 1, 2, \ldots, 4$, there exist line integrals with opposite directions that cancel each other out. Therefore, we have

$$\frac{d\mathcal{M}}{dt} = \frac{1}{2}\sum_{k=1}^{6}\sum_{c=1}^{4} \int_{\partial\Omega_{k,c}} \nabla u(\mathbf{x}, t) \cdot \mathbf{n} ds = 0,$$

which shows that the mass conservation law of the CAC equation on the unfolded cubic surface is satisfied.



(a) (b)

Figure 3. Schematic illustrations of boundary conditions for sub-domain Ω_4.

3. Numerical method

Now, we describe the numerical method of the CAC equation on the cubic surface. Let the number of grid N for a one sub-domain be a positive integer. Then, we define the set of indexes for the sub-domain Ω_k, $k = 1, 2, \ldots, 6$ and Ω.

$$\begin{aligned}
I_1^d &= \{(i,j) \mid i = 1,2,\ldots,N,\ j = 1,2\ldots,N\},\\
I_2^d &= \{(i,j) \mid i = 1,2\ldots,N,\ j = N+1,N+2,\ldots,2N\},\\
I_3^d &= \{(i,j) \mid i = N+1,N+2,\ldots,2N,\ j = N+1,N+2,\ldots,2N\},\\
I_4^d &= \{(i,j) \mid i = 2N+1,2N+2,\ldots,3N,\ j = N+1,N+2,\ldots,2N\},\\
I_5^d &= \{(i,j) \mid i = 3N+1,3N+2,\ldots,4N,\ j = N+1,N+2,\ldots,2N\},\\
I_6^d &= \{(i,j) \mid i = 1,2,\ldots,N,\ j = 2N+1,2N+2,\ldots,3N\},\\
I^d &= \cup_{k=1}^6 I_k^d.
\end{aligned}$$

The discretization domains for the sub-domains Ω_k, $k = 1, \cdots, 6$ in 2D space are defined by $\Omega_k^d = \{(x_i = (i-0.5)h, y_j = (j-0.5)h) \mid (i,j) \in I_k\}$, where $h = 1/N$ is a space step size. We discretize unfolded cubic surface domain Ω as $\Omega^d = \cup_{k=1}^6 \Omega_k^d$, see Figure 4(a).

Next, we consider the boundary conditions for the domain Ω^d. For clarity, we focus only on one sub-domain Ω_4^d, within the unfolded cubic surface domain Ω^d. The boundary conditions at $x = 2$ and $x = 3$ for Ω_4^d are appropriately defined by the sub-domains Ω_3^d and Ω_5^d of the discrete unfolded cubic surface domain Ω^d. However, the boundary conditions at $y = 1$ and $y = 2$ for Ω_4^d are defined by rotating Ω_1^d and Ω_6^d counterclockwise and clockwise by 180 degrees, respectively, and then placing them below and above Ω_4^d respectively. These boundary conditions are schematically shown in Figure 4(b). Thus, considering the folded cubic surface, we apply boundary conditions for each subdomain Ω_k^d, $k = 1, 2, \ldots, 6$ of the discrete unfolded cubic surface domain. For $m = 1, \ldots, N$,

$$\begin{aligned}
&\text{On} \quad \Omega_1^d, \quad u_{0,m} = u_{3N+m,N+1},\ u_{m,0} = u_{3N+1-m,N+1},\ u_{N+1,m} = u_{2N+1-m,N+1},\\
&\text{On} \quad \Omega_2^d, \quad u_{0,N+m} = u_{4N,N+m},\\
&\text{On} \quad \Omega_3^d, \quad , \ u_{N+i,N}^n = u_{N,N+1-m}^n,\ u_{N+m,2N+1}^n = u_{N,2N+m}^n,\\
&\text{On} \quad \Omega_4^d, \quad u_{2N+m,N} = u_{N+1-m,1},\ u_{2N+m,2N+1} = u_{N+1-m,3N},\\
&\text{On} \quad \Omega_5^d, \quad u_{3N+m,N} = u_{1,m},\ u_{3N+m,2N+1} = u_{1,3N+1-m},\ u_{4N+1,N+m} = u_{1,N+m},\\
&\text{On} \quad \Omega_6^d, \quad u_{0,2N+m} = u_{4N+1-m,2N},\ u_{m,3N+1} = u_{3N+1-m,2N},\ u_{N+1,2N+m} = u_{N+m,2N}.
\end{aligned}$$

The discrete total mass is defined by

$$\mathcal{M}^h(u^n) = \sum_{(i,j)\in I^d} \frac{u_{ij}^n + 1}{2} h^2.$$

The proposed algorithm consists of two steps. First, we compute

$$\begin{aligned}
\frac{u_{ij}^* - u_{ij}^n}{\Delta t} &= -\frac{F'(u_{ij}^n)}{\epsilon^2} + \frac{u_{i+1,j}^n + u_{i-1,j}^n + u_{i,j+1}^n + u_{i,j-1}^n - 4u_{ij}^n}{h^2}\\
&= \frac{u_{ij}^n - (u_{ij}^n)^3}{\epsilon^2} + \frac{u_{i+1,j}^n + u_{i-1,j}^n + u_{i,j+1}^n + u_{i,j-1}^n - 4u_{ij}^n}{h^2}.
\end{aligned}$$

Thus, we have

$$u_{ij}^{*} = u_{ij}^{n} + \Delta t\left(\frac{u_{ij}^{n} - (u_{ij}^{n})^{3}}{\epsilon^{2}} + \frac{u_{i+1,j}^{n} + u_{i-1,j}^{n} + u_{i,j+1}^{n} + u_{i,j-1}^{n} - 4u_{ij}^{n}}{h^{2}}\right). \tag{3.1}$$

Next, we compute the conservation term using the solution of Eq (3.1) to solve the CAC equation.

$$u_{ij}^{n+1} = u_{ij}^{*} + \Delta t\beta^{*}\sqrt{F(u_{ij}^{*})}, \tag{3.2}$$

where

$$\beta^{*} = \frac{\sum_{(i,j)\in I^{d}}(u_{ij}^{0} - u_{ij}^{*})}{\Delta t\sum_{(i,j)\in I^{d}}\sqrt{F(u_{ij}^{*})}}.$$

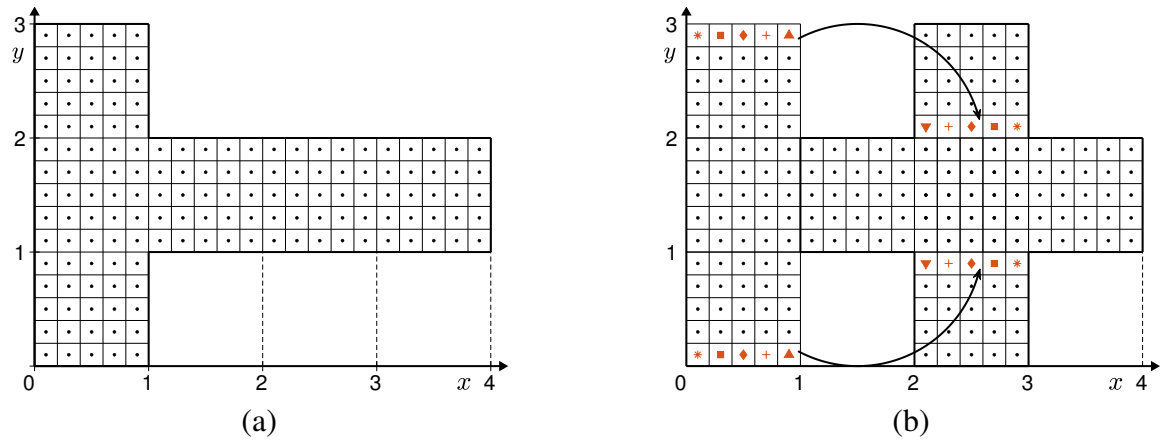


Figure 4. Schematic of numerical discrete domain Ω^{d} and boundary conditions for the Ω_{4}^{d}.

Theorem 1. *The proposed method satisfies the discrete mass conservation law of the CAC equation.*

Proof. For any nonnegative integer n, we can rewrite Eq (3.2) by the definition of β^{*} as

$$u_{ij}^{n+1} = u_{ij}^{*} + \frac{\sum_{(i,j)\in I^{d}}(u_{ij}^{0} - u_{ij}^{*})}{\sum_{(i,j)\in I^{d}}\sqrt{F(u_{ij}^{*})}}\sqrt{F(u_{ij}^{*})}.$$

Thus, we obtain the following equation:

$$\begin{aligned}\sum_{(i,j)\in I^{d}} u_{ij}^{n+1} &= \sum_{(i,j)\in I^{d}}\left(u_{ij}^{*} + \frac{\sum_{(i,j)\in I^{d}}(u_{ij}^{0} - u_{ij}^{*})}{\sum_{(i,j)\in I^{d}}\sqrt{F(u_{ij}^{*})}}\sqrt{F(u_{ij}^{*})}\right)\\ &= \sum_{(i,j)\in I^{d}} u_{ij}^{*} + \frac{\sum_{(i,j)\in I^{d}}(u_{ij}^{0} - u_{ij}^{*})}{\sum_{(i,j)\in I^{d}}\sqrt{F(u_{ij}^{*})}}\sum_{(i,j)\in I^{d}}\sqrt{F(u_{ij}^{*})} = \sum_{(i,j)\in I^{d}} u_{ij}^{0}.\end{aligned}$$

Hence, the proposed scheme satisfies the mass conservation law of the CAC equation by the above equation.

$$\begin{aligned}\mathcal{M}^h(u^{n+1}) &= \sum_{(i,j)\in I^d} \frac{u_{ij}^{n+1}+1}{2}h^2 = \frac{h^2}{2}\sum_{(i,j)\in I^d} u_{ij}^{n+1} + 3N^2h^2 \\ &= \frac{h^2}{2}\sum_{(i,j)\in I^d} u_{ij}^{0} + 3N^2h^2 = \mathcal{M}^h(u^0).\end{aligned}$$

□

4. Numerical experiments

Now, we perform the numerical simulation for the CAC equation using the proposed method. We define the three-dimensional domain $\hat{\Omega}$ as the cubic surface with a side length $L = 5$ and then Ω^d is defined as the corresponding discrete unfolded cubic surface domain. The normalized discrete total mass is defined by $\mathcal{M}^h(u^n)/L^2$. The interface layer parameter with the spatial step size is defined as

$$\epsilon_m = \frac{mh}{2\sqrt{2}\tanh^{-1}(0.9)},$$

which means there are m grid points in the interface layer. First, we consider the following initial condition on Ω^d.

$$u(x_i, y_j, 0) = -0.8 + 0.1\text{rand}(x_i, y_j),$$

where $\text{rand}(x_i, y_j)$ is random value from -1 to 1 at the point (x_i, y_j). The parameters used are $N = 128$, $\epsilon = \epsilon_8$, $\Delta t = 0.5\epsilon^2h^2/(2\epsilon^2 + h^2)$, and $T = 100.788$. We define the maximum and minimum values of the computational solutions at $n\Delta t$ as

$$Max(u^n) = \max_{(i,j)\in I^d} u_{ij}^n, \quad Min(u^n) = \min_{(i,j)\in I^d} u_{ij}^n,$$

respectively. Figure 5 displays the temporal evolution of the computational solution for the CAC equation. The computational results from the numerical simulation show a distinct process of domain coarsening, where larger domains grow progressively larger while smaller domains shrink and eventually disappear. This phenomenon reflects the tendency of the system to minimize its interfacial energy, which leads to the dominance of larger regions and the elimination of smaller ones over time.

We observed that the computational solution calculated using the proposed algorithm satisfies the discrete mass conservation. Furthermore, we maintain the properties of the CAC equation when the interface of numerical solutions of the CAC equation solved by the proposed scheme crosses the boundary between different Ω_k^d.

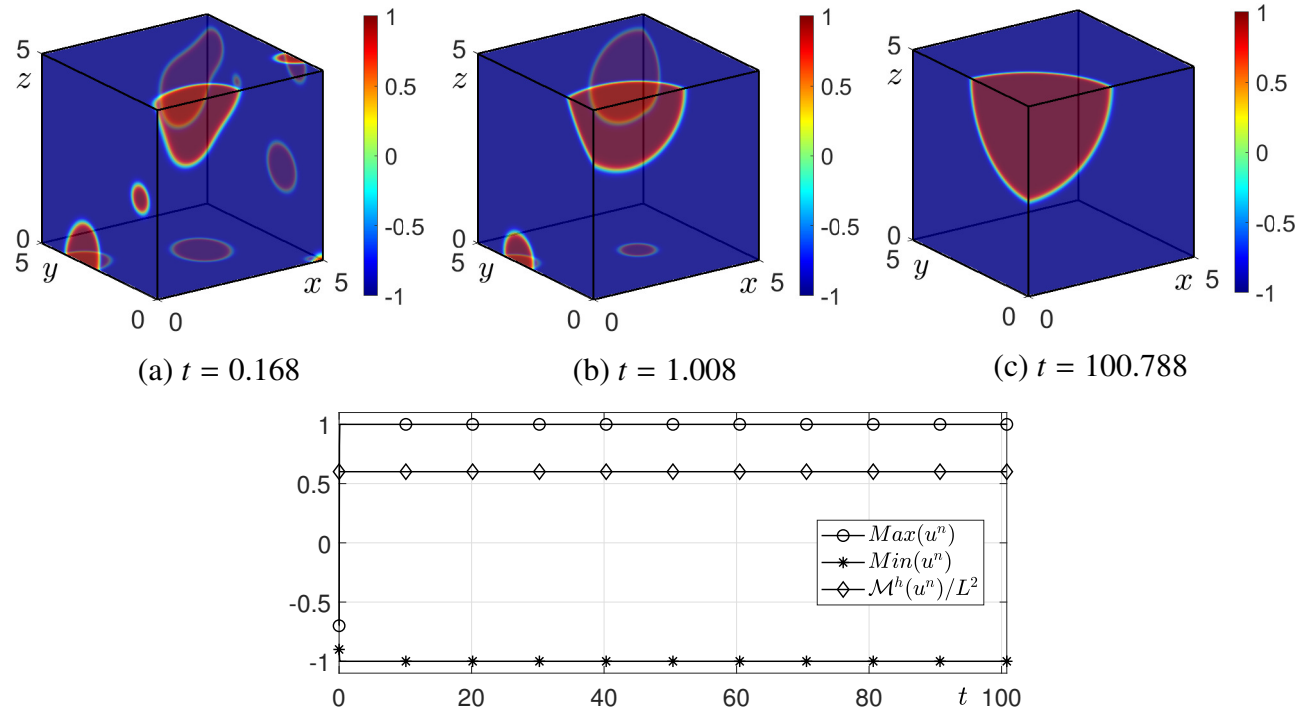


Figure 5. (a)–(c) Numerical solution for the CAC equation at times $t = 0.168$, 1.008, and 100.788. (d) Temporal evolution of the maximum and minimum values, and the normalized discrete total mass of the computational solutions.

Next, the following initial condition is considered to observe the effect of the unit folded cubic surface domain.

$$u(x_i, y_j, 0) = \begin{cases} 1 & \text{if } \sqrt{(x_i - 2.5)^2 + (y_j - 12.5)^2} < 0.3, \\ 1 & \text{if } 9.25 < y_j < 9.75, \\ -1 & \text{otherwise.} \end{cases}$$

The parameters used are $N = 128$, $\epsilon = \epsilon_8$, $\Delta t = 0.5\epsilon^2 h^2/(2\epsilon^2 + h^2)$, and $T = 7.727$. Figure 6 shows the computational solutions for the CAC equation at time $t = 0$, 1.176, 4.368, and 7.727.

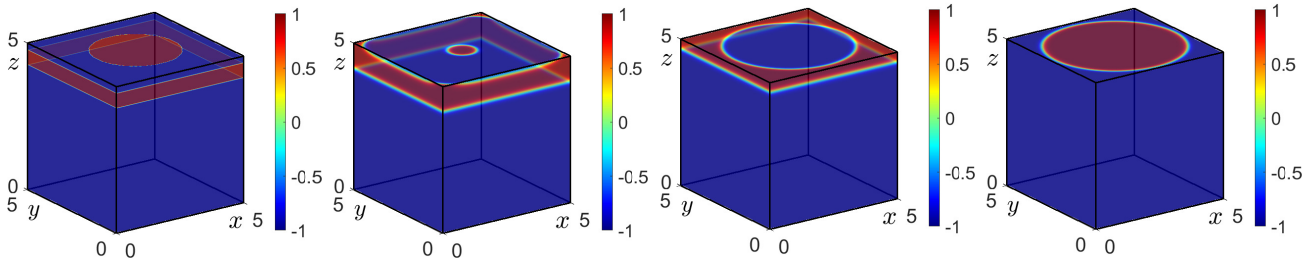
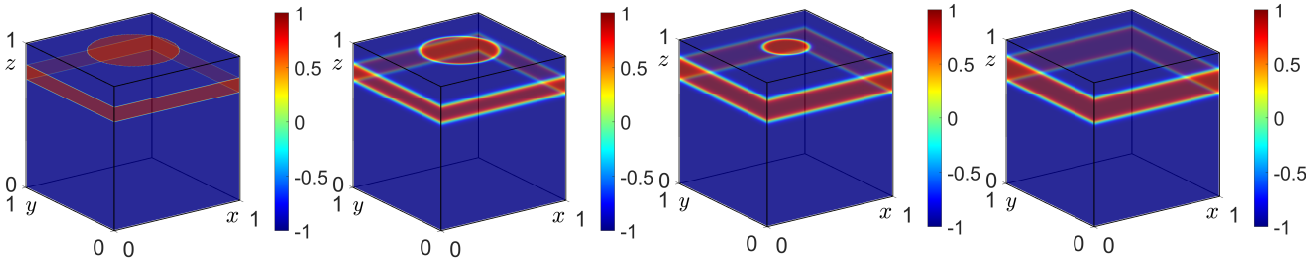


Figure 6. Snapshots of the numerical solution for the CAC equation are shown at times $t = 0$, 1.176, 4.368, and 7.727.

We can observe that the interface layer of computational solutions at Ω_k for $k = 2, 3, 4, 5$ tends to approach the boundary with Ω_6 by the CAC equation, while the characteristics of folded cubic surfaces result in the interface layer crossing the boundary and forming a circle according to the geometric properties of the CAC equation. We use the following initial condition, similar to the above, except with some different phase positions.

$$u(x_i, y_j, 0) = \begin{cases} 1 & \text{if } \sqrt{(x_i - 2.5)^2 + (y_j - 12.5)^2} < 0.3, \\ 1 & \text{if } 8.75 < y_j < 9.25, \\ -1 & \text{otherwise.} \end{cases}$$

The parameters used are $N = 128$, $\epsilon = \epsilon_8$, $\Delta t = 0.5\epsilon^2 h^2/(2\epsilon^2 + h^2)$, and $T = 1.680$. Figure 7 shows the numerical solutions for the CAC equation on the folded cubic surface using the proposed scheme.



(a) Initial (b) $t = 0.336$ (c) $t = 1.008$ (d) $t = 1.680$

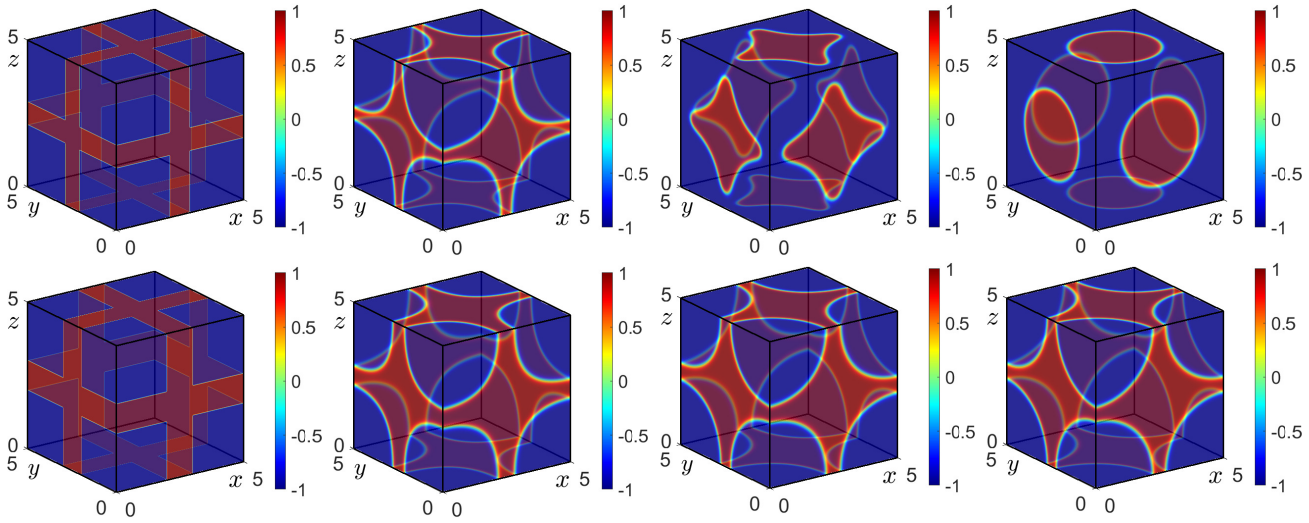
Figure 7. Snapshots of the numerical solution for the CAC equation are shown at times $t = 0,\ 0.336,\ 1.008$, and 1.680.

We observe that the computational solution quickly becomes equilibrium in Ω_k for $k = 2, 3, 4, 5$ since the interface layer does not cross the boundary of Ω_6, unlike the results in Figure 6.

Next, the two initial conditions on Ω^d are given by

$$u(x_i, y_j, 0) = \begin{cases} 1 & \text{if } 2.5 - w < x_i < 2.5 + w,\ 2.5 - w < y_j < 2.5 + w,\ \ (x_i, y_j) \in \Omega_1^d, \\ 1 & \text{if } 2.5 - w < x_i < 2.5 + w,\ 7.5 - w < y_j < 7.5 + w,\ \ (x_i, y_j) \in \Omega_2^d, \\ 1 & \text{if } 7.5 - w < x_i < 7.5 + w,\ 7.5 - w < y_j < 7.5 + w,\ \ (x_i, y_j) \in \Omega_3^d, \\ 1 & \text{if } 12.5 - w < x_i < 12.5 + w,\ 7.5 - w < y_j < 7.5 + w,\ \ (x_i, y_j) \in \Omega_4^d, \\ 1 & \text{if } 17.5 - w < x_i < 17.5 + w,\ 7.5 - w < y_j < 7.5 + w,\ \ (x_i, y_j) \in \Omega_5^d, \\ 1 & \text{if } 2.5 - w < x_i < 2.5 + w,\ 12.5 - w < y_j < 12.5 + w,\ \ (x_i, y_j) \in \Omega_6^d, \\ -1 & \text{otherwise,} \end{cases}$$

where $w = 0.4$ or $w = 0.5$. We used parameters $N = 128$, $\epsilon = \epsilon_8$, $\Delta t = 0.5\epsilon^2 h^2/(2\epsilon^2 + h^2)$, $T = 1.680$. The top and bottom rows in Figure 8 show the temporal evolution of the numerical solutions of the CAC equation with $w = 0.4$ and $w = 0.5$, respectively, at times $t = 0$, 0.504, 0.840, and 1.680.



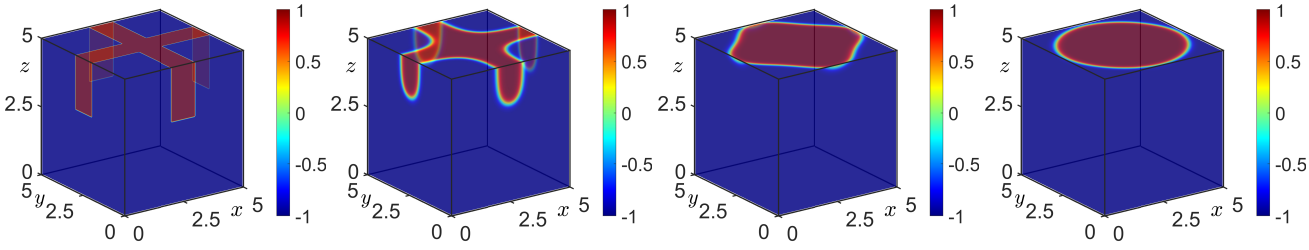
(a) Initial (b) $t = 0.504$ (c) $t = 0.840$ (d) $t = 1.680$

Figure 8. Snapshot of the numerical solutions of the CAC equation with $w = 0.4,\ 0.5$ from top to bottom.

We investigate the property of motion by mass conserving mean curvature flow on a cubic surface. The initial condition is defined as follows:

$$u(x, y, 0) = \begin{cases} 1, & \text{if } 12 < y < 13, \\ 1, & \text{if } 2 < (x \bmod 5) < 3,\ y > 8, \\ -1, & \text{otherwise,} \end{cases}$$

where x mod 5 represents the remainder when x is divided by 5. The parameters used are $N = 100$, $\epsilon = \epsilon_8$, $\Delta t = 0.5\epsilon^2 h^2/(2\epsilon^2 + h^2)$, and $T = 1.927$. Figure 9 shows the temporal evolution of u. In Fig. 9(a), the phase of u presents like a cross-shaped structure with sharp interfaces. As time progresses, the interface gradually smooths out due to mass conserving mean curvature flow. As shown in Figure 9(d), the interface eventually evolves into a stable, circular shape near the top of the domain.



(a) $t = 0$ (b) $t = 0.275$ (c) $t = 1.101$ (d) $t = 1.927$

Figure 9. Numerical solutions of u for motion by mean curvature flow at (a) $t = 0$, (b) $t = 0.275$, (c) $t = 1.101$, and (d) $t = 1.927$.

5. Discussion

In this section, we discuss the application of an implicit scheme for the CAC equation on cubic surfaces, along with its potential benefits and limitations of the implicit scheme, and compare

the implicit scheme with the proposed explicit numerical method through numerical experiments. Generally, the implicit scheme provides higher stability than the explicit scheme. This means that, compared to the time step size limitation of the explicit scheme, the implicit scheme can use relatively larger time step sizes without causing the numerical solution to blow up, thus maintaining stability [34, 35]. However, since the implicit scheme requires solving a nonlinear system at each time step, the computations can become more complex and time-consuming depending on the given discretization grid [36]. To compare the proposed method with the implicit scheme, a nonlinear convex splitting method is applied to the CAC equation [37].

$$\frac{u_{ij}^{*} - u_{ij}^{n}}{\Delta t} = \frac{u_{ij}^{n} - (u_{ij}^{*})^3}{\epsilon^2} + \frac{u_{i+1,j}^{*} + u_{i-1,j}^{*} + u_{i,j+1}^{*} + u_{i,j-1}^{*} - 4u_{ij}^{*}}{h^2}, \tag{5.1}$$

$$u_{ij}^{n+1} = u_{ij}^{*} + \Delta t \beta^{*} \sqrt{F(u_{ij}^{*})}. \tag{5.2}$$

Let $u_{ij}^{*,s}$ and $u_{ij}^{*,s+1}$ be the approximations of u_{ij}^{*} before and after a Gauss–Seidel iteration, respectively. We linearize the nonlinear term $(u_{ij}^{*})^3$ in the Gauss–Seidel iteration method as

$$(u_{ij}^{*,s+1})^3 = 3(u_{ij}^{*,s})^2 u_{ij}^{*,s+1} - 2(u_{ij}^{*,s})^3.$$

Thus, Eq. (5.1) can be rewritten in Gauss–Seidel form as follows.

$$u_{ij}^{*,s+1} = \left[\left(\frac{1}{\Delta t} + \frac{1}{\epsilon^2} \right) u_{ij}^{n} + \frac{2}{\epsilon^2} (u_{ij}^{*,s})^3 + \frac{u_{i+1,j}^{*,s} + u_{i-1,j}^{*,s+1} + u_{i,j+1}^{*,s} + u_{i,j-1}^{*,s+1}}{h^2} \right] \Big/ D, \tag{5.3}$$

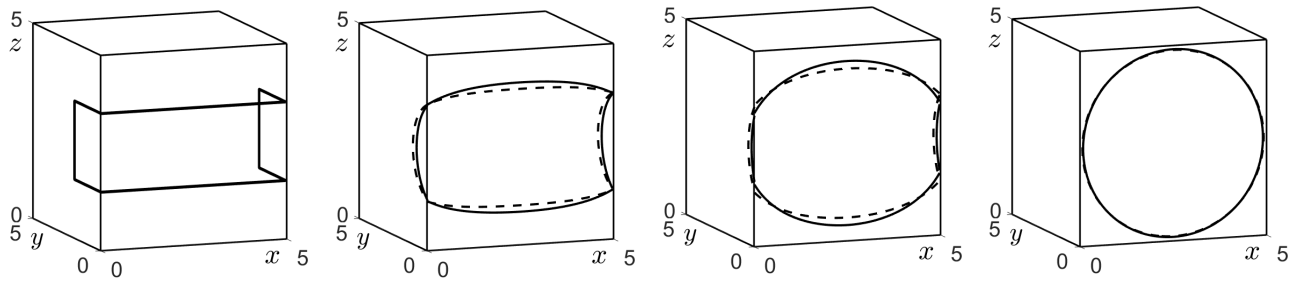
where

$$D = \frac{1}{\Delta t} + \frac{3(u_{ij}^{*,s})^2}{\epsilon^2} + \frac{4}{h^2}.$$

The discrete l_2-norm is defined as $\|u^n\|_2 = \sqrt{\sum_{(i,j)\in I^d} (u_{ij}^{n})^2/(6N^2)}$. We calculate Eq. (5.3) repeatedly until the l_2-norm of the consecutive error $\|u_{ij}^{*,s+1} - u_{ij}^{*,s}\|_2$ is less than a given tolerance $tol = 10^{-6}$. For numerical simulation, the initial condition on Ω^d is given by

$$u(x_i, y_j, 0) = \begin{cases} 1, & \text{if } 6.5 < y_j < 8.5,\ x_i < 7, \\ 1, & \text{if } 6.5 < y_j < 8.5,\ x_i > 18, \\ -1, & \text{otherwise.} \end{cases}$$

The parameters used are $N = 128$, $\epsilon = \epsilon_8$, and the final time $T = 12.5$. The implicit convex splitting scheme allows for the use of larger time step sizes. Therefore, we use $\Delta t = 0.00025$ for the proposed method, while the implicit method uses a time step size that is 8 times larger, $\Delta t = 0.002$. Figure 10 shows the zero-contour of the computational solutions of the CAC equation using the proposed and implicit convex splitting methods. The solid line shows the computational solution from the proposed scheme, while the dashed line displays that from the implicit scheme.



(a) Initial (b) $t = 2$ (c) $t = 4$ (d) $t = 12.5$

Figure 10. The zero-level contour of the numerical solution using the proposed and implicit schemes at times $t = 0$, 2, 4, and 12.5. The solid line represents the numerical solution using the proposed method and the dashed line represents the numerical solution using the implicit method.

We observed that the numerical solution using the implicit scheme is less affected by the motion by mass conserving mean curvature compared to the numerical solution using the proposed method. Table 1 lists the central processing unit (CPU) time for obtaining a numerical solution for time $t = 12.5$ of the CAC equation using the explicit and implicit convex splitting methods.

Table 1. CPU times for different numerical scheme.

Method	Proposed	Implicit
CPU time(s)	177.6461	384.0172

We observed that although the time step size used in the implicit method was 8 times larger than that used in the proposed method, the CPU time for the implicit method was more than 2 times as long.

6. Conclusions

In this work, we introduced a fully numerical method for solving the CAC equation on a cubic surface. The method is structured in two steps: first, solving the AC equation using an efficient explicit finite difference approach, followed by incorporating a conservation term to ensure consistency with the conservation principles of the CAC equation. Numerical experiments confirm that the proposed scheme effectively conserves discrete mass, a critical requirement for the accurate solution of the CAC equation. Furthermore, the solution showed constrained motion by mass conserving mean curvature, another essential property of the CAC equation, which validates the robustness and accuracy of the proposed method. These findings demonstrate that the scheme is both reliable and computationally efficient for solving the CAC equation on complex geometries such as cubic surfaces. In future work, we will consider unconditionally stable and high-order accurate numerical methods for the CAC equation on cubic surfaces and perform an analysis of the stability, convergence, and consistency, etc., of the numerical methods.

Author contributions

Youngjin Hwang: Conceptualization, Methodology, Software, Visualization, Formal analysis, Validation, Investigation, Writing-original draft; Jyoti: Validation, Investigation, Funding acquisition, Writing-original draft; Soobin Kwak: Visualization, Validation, Investigation, Writing-original draft; Hyundong Kim: Validation, Investigation, Funding acquisition, Writing-original draft; Junseok Kim: Conceptualization, Methodology, Project administration, Supervision, Funding acquisition, Writing-original draft;

Acknowledgments

The corresponding author (J.S. Kim) was supported by the National Research Foundation of Korea(NRF) grant funded by the Korea government(MSIT) (No. 2022R1A2C1003844). Jyoti was supported by Brain Pool program funded by the Ministry of Science and ICT through the National Research Foundation of Korea (2022H1D3A2A02081237). Hyundong Kim was supported by Basic Science Research Program through the National Research Foundation of Korea(NRF) funded by the Ministry of Education (2021R1A6A1A03044326). The authors extend their thanks to the reviewers for the valuable and constructive input they provided during the revision of the article.

Conflict of interest

Professor Hyundong Kim is the Guest Editor for AIMS Mathematics and was not involved in the editorial review or the decision to publish this article. All authors declare that there are no competing interests.

Appendix

The following code is the main program with a random initial condition, which is also available from the corresponding author's webpage:

`https://mathematicians.korea.ac.kr/cfdkim/open-source-codes/`

```
clear; fs = 23; axisfs = 19; lw = 1.5; ms = 8;
N = 128; NN = 100; cN = linspace(-1,1,NN+1); falpha = 0.6; Nx = 4*N; Ny = 3*N; rand('seed',0830)
xL = 0; xR = 20; yL = 0; yR = Ny/Nx*xR; h = (xR-xL)/Nx; h2 = h^2; NNN = 100;
x = linspace(xL-0.5*h,xR+0.5*h,Nx+2); y = linspace(yL-0.5*h,yR+0.5*h,Ny+2);
m = 8; eps = m*h/(2*sqrt(2)*atanh(0.9)); dt = 0.5*eps^2*h^2/(2*eps^2+h^2); Nt = 300000; T = dt*Nt
phi = -0.8+0.1*(1-2*rand(Nx+2,Ny+2)); phi0 = phi;

figure(1); clf; hold on; box on; set(gcf,'position',[100 500 500 350])
xxx = linspace(xL/4+0.5*h,xR/4-0.5*h,N); yyy = linspace(xL/4+0.5*h,xR/4-0.5*h,N);
[xx, yy] = meshgrid(xxx,yyy); t = hgtransform;
[C,hh] = contourf(xx,yy,phi(2:N+1,N+1:-1:2)',cN,'facealpha',falpha,'LineStyle','none');
hh.ZLocation = xL/4+0.5*h;
[C,hh] = contourf(xx,yy,phi(2:N+1,2*N+2:3*N+1)',cN,'facealpha',falpha,'LineStyle','none');
hh.ZLocation = xR/4-0.5*h;
[C,hh] = contourf(xx,yy,phi(2:N+1,N+2:2*N+1)',cN,'facealpha',falpha,'Parent',t,'LineStyle','none');
hh.ZLocation = -0.5*h; ry_angle = 1/2*pi;
Ry = makehgtform('xrotate',ry_angle); t.Matrix = Ry; t2 = hgtransform;
[C,hh] = contourf(xx,yy,phi(N+2:2*N+1,N+2:2*N+1),cN,'facealpha',falpha,'Parent',t2,'LineStyle','none');
```

```
hh.ZLocation = -0.5*h; ry_angle = -1/2*pi; Ry = makehgtform('yrotate',ry_angle);
t2.Matrix = Ry; hh.ZLocation = -xR/4+0.5*h; t3 = hgtransform;
[C,hh] = contourf(xx,yy,phi(3*N+1:-1:2*N+2,N+2:2*N+1)',cN,'facealpha',falpha,'Parent',t3,'LineStyle','none');
hh.ZLocation = -0.5*h; ry_angle = 1/2*pi; Ry = makehgtform('xrotate',ry_angle);
t3.Matrix = Ry; hh.ZLocation = -xR/4+0.5*h; t4 = hgtransform;
[C,hh] = contourf(xx,yy,phi(4*N+1:-1:3*N+2,N+2:2*N+1),cN,'facealpha',falpha,'Parent',t4,'LineStyle','none');
hh.ZLocation = -0.5*h; ry_angle = -1/2*pi; Ry = makehgtform('yrotate',ry_angle);
t4.Matrix = Ry; hh.ZLocation = -0.5*h;
set(gca,'CLim',[-1 1],'fontsize',axisfs); axis([xL/4 xR/4 xL/4 xR/4 xL/4 xR/4])
plot3([xL/4 xL/4],[xL/4 xL/4],[xL/4 xR/4],'k-',[xL/4 xL/4],[xL/4 xR/4],[xL/4 xL/4],'k-' ...
    ,[xL/4 xR/4],[xL/4 xL/4],[xL/4 xL/4],'k-','linewidth',lw)
plot3([xR/4 xR/4],[xR/4 xR/4],[xL/4 xR/4],'k-',[xR/4 xR/4],[xL/4 xR/4],[xR/4 xR/4],'k-' ...
    ,[xL/4 xR/4],[xR/4 xR/4],[xR/4 xR/4],'k-','linewidth',lw)
plot3([xL/4 xL/4],[xR/4 xR/4],[xL/4 xR/4],'k-',[xL/4 xL/4],[xL/4 xR/4],[xR/4 xR/4],'k-' ...
    ,[xL/4 xR/4],[xL/4 xL/4],[xR/4 xR/4],'k-','linewidth',lw)
plot3([xR/4 xR/4],[xL/4 xL/4],[xL/4 xR/4],'k-',[xR/4 xR/4],[xL/4 xR/4],[xL/4 xL/4],'k-' ...
    ,[xL/4 xR/4],[xR/4 xR/4],[xL/4 xL/4],'k-','linewidth',lw)
xticks([xL/4 xR/4]); yticks([xL/4 xR/4]); zticks([xL/4 xR/4]); view([-35 20]); axis image;
axis([xL/4 xR/4 xL/4 xR/4 xL/4 xR/4]); colormap jet; a = colorbar; a.Position = [0.88 0.12 0.0275 0.8];
text('interpreter','latex','string','$x$','FontSize',fs+2,'Position',[3.16 -1.77 0.26])
text('interpreter','latex','string','$y$','FontSize',fs,'Position',[-1.96 2.19 0.54])
text('interpreter','latex','string','$z$','FontSize',fs+2,'Position',[-1.58 3.89 4.96]); drawnow;

for it = 1:Nt
phi(1,2:N+1) = phi(3*N+2:4*N+1,N+2); phi(2:N+1,1) = phi(3*N+1:-1:2*N+2,N+2);
phi(N+2,2:N+1) = phi(2*N+1:-1:N+2,N+2); phi(1,N+2:2*N+1) = phi(4*N+1,N+2:2*N+1);
phi(1,2*N+2:3*N+1) = phi(4*N+1:-1:3*N+2,2*N+1); phi(2:N+1,3*N+2) = phi(3*N+1:-1:2*N+2,2*N+1);
phi(N+2,2*N+2:3*N+1) = phi(N+2:2*N+1,2*N+1); ophi1 = phi;
phi(N+2:2*N+1,N+1) = phi(N+1,N+1:-1:2); phi(N+2:2*N+1,2*N+2) = phi(N+1,2*N+2:3*N+1);
phi(2*N+2:3*N+1,N+1) = phi(N+1:-1:2,2); phi(2*N+2:3*N+1,2*N+2) = phi(N+1:-1:2,3*N+1);
phi(3*N+2:4*N+1,N+1) = phi(2,2:N+1); phi(3*N+2:4*N+1,2*N+2) = phi(2,3*N+1:-1:2*N+2);
phi(4*N+2,N+2:2*N+1) = phi(2,N+2:2*N+1); ophi = phi;
phi(2:N+1,2:Ny+1) = ophi1(2:N+1,2:Ny+1) ...
    +dt*((ophi1(2:N+1,2:Ny+1)-ophi1(2:N+1,2:Ny+1).^3)/eps^2 ...
    +(ophi1(1:N,2:Ny+1)+ophi1(3:N+2,2:Ny+1)+ophi1(2:N+1,1:Ny)+ophi1(2:N+1,3:Ny+2) ...
    -4*ophi1(2:N+1,2:Ny+1))/h^2);
phi(N+2:Nx+1,N+2:2*N+1) = ophi(N+2:Nx+1,N+2:2*N+1) ...
    +dt*((ophi(N+2:Nx+1,N+2:2*N+1)-ophi(N+2:Nx+1,N+2:2*N+1).^3)/eps^2 ...
    +(ophi(N+1:Nx,N+2:2*N+1)+ophi(N+3:Nx+2,N+2:2*N+1) ...
    +ophi(N+2:Nx+1,N+1:2*N)+ophi(N+2:Nx+1,N+3:2*N+2)-4*ophi(N+2:Nx+1,N+2:2*N+1))/h^2);
beta = (sum(phi0(2:N+1,2:Ny+1)-phi(2:N+1,2:Ny+1),'all')+sum(phi0(N+2:Nx+1,N+2:2*N+1) ...
    -phi(N+2:Nx+1,N+2:2*N+1),'all'))/(sum(0.5*abs(phi(2:N+1,2:Ny+1).^2-1),'all') ...
    +sum(0.5*abs(phi(N+2:Nx+1,N+2:2*N+1).^2-1),'all'));
phi(2:N+1,2:Ny+1) = phi(2:N+1,2:Ny+1)+beta*0.5*abs(phi(2:N+1,2:Ny+1).^2-1);
phi(N+2:Nx+1,N+2:2*N+1) = phi(N+2:Nx+1,N+2:2*N+1)+beta*0.5*abs(phi(N+2:Nx+1,N+2:2*N+1).^2-1);

if mod(it,NNN) == 0
figure(2); clf; hold on; box on; set(gcf,'position',[100 500 500 350])
xxx = linspace(xL/4+0.5*h,xR/4-0.5*h,N); yyy = linspace(xL/4+0.5*h,xR/4-0.5*h,N);
[xx, yy] = meshgrid(xxx,yyy); t = hgtransform;
[C,hh] = contourf(xx,yy,phi(2:N+1,N+1:-1:2)',cN,'facealpha',falpha,'LineStyle','none');
hh.ZLocation = xL/4+0.5*h;
[C,hh] = contourf(xx,yy,phi(2:N+1,2*N+2:3*N+1)',cN,'facealpha',falpha,'LineStyle','none');
hh.ZLocation = xR/4-0.5*h;
[C,hh] = contourf(xx,yy,phi(2:N+1,N+2:2*N+1)',cN,'facealpha',falpha,'Parent',t,'LineStyle','none');
hh.ZLocation = -0.5*h; ry_angle = 1/2*pi;
Ry = makehgtform('xrotate',ry_angle); t.Matrix = Ry; t2 = hgtransform;
[C,hh] = contourf(xx,yy,phi(N+2:2*N+1,N+2:2*N+1),cN,'facealpha',falpha,'Parent',t2,'LineStyle','none');
hh.ZLocation = -0.5*h; ry_angle = -1/2*pi; Ry = makehgtform('yrotate',ry_angle);
t2.Matrix = Ry; hh.ZLocation = -xR/4+0.5*h; t3 = hgtransform;
[C,hh] = contourf(xx,yy,phi(3*N+1:-1:2*N+2,N+2:2*N+1)',cN,'facealpha',falpha,'Parent',t3,'LineStyle','none');
hh.ZLocation = -0.5*h; ry_angle = 1/2*pi; Ry = makehgtform('xrotate',ry_angle);
t3.Matrix = Ry; hh.ZLocation = -xR/4+0.5*h; t4 = hgtransform;
[C,hh] = contourf(xx,yy,phi(4*N+1:-1:3*N+2,N+2:2*N+1),cN,'facealpha',falpha,'Parent',t4,'LineStyle','none');
hh.ZLocation = -0.5*h; ry_angle = -1/2*pi; Ry = makehgtform('yrotate',ry_angle);
t4.Matrix = Ry; hh.ZLocation = -0.5*h;
set(gca,'CLim',[-1 1],'fontsize',axisfs); axis([xL/4 xR/4 xL/4 xR/4 xL/4 xR/4])
plot3([xL/4 xL/4],[xL/4 xL/4],[xL/4 xR/4],'k-',[xL/4 xL/4],[xL/4 xR/4],[xL/4 xL/4],'k-' ...
    ,[xL/4 xR/4],[xL/4 xL/4],[xL/4 xL/4],'k-','linewidth',lw)
plot3([xR/4 xR/4],[xR/4 xR/4],[xL/4 xR/4],'k-',[xR/4 xR/4],[xL/4 xR/4],[xR/4 xR/4],'k-' ...
    ,[xL/4 xR/4],[xR/4 xR/4],[xR/4 xR/4],'k-','linewidth',lw)
```

```
plot3([xL/4 xL/4],[xR/4 xR/4],[xL/4 xR/4],'k-',[xL/4 xL/4],[xL/4 xR/4],[xR/4 xR/4],'k-' ...
    ,[xL/4 xR/4],[xL/4 xL/4],[xR/4 xR/4],'k-','linewidth',lw)
plot3([xR/4 xR/4],[xL/4 xL/4],[xL/4 xR/4],'k-',[xR/4 xR/4],[xL/4 xR/4],[xL/4 xL/4],'k-' ...
    ,[xL/4 xR/4],[xR/4 xR/4],[xL/4 xL/4],'k-','linewidth',lw)
xticks([xL/4 xR/4]); yticks([xL/4 xR/4]); zticks([xL/4 xR/4]); view([-35 20]); axis image;
axis([xL/4 xR/4 xL/4 xR/4 xL/4 xR/4]); colormap jet; a = colorbar; a.Position = [0.88 0.12 0.0275 0.8];
text('interpreter','latex','string','$x$','FontSize',fs+2,'Position',[3.16 -1.77 0.26])
text('interpreter','latex','string','$y$','FontSize',fs,'Position',[-1.96 2.19 0.54])
text('interpreter','latex','string','$z$','FontSize',fs+2,'Position',[-1.58 3.89 4.96]); drawnow
end
end
```

References

1. S. M. Allen, J. W. Cahn, A microscopic theory for antiphase boundary motion and its application to antiphase domain coarsening, *Acta Metall.*, **27** (1979), 1085–1095. https://doi.org/10.1016/0001-6160(79)90196-2
2. S. Zhai, Z. Weng, Y. Mo, X. Feng, Energy dissipation and maximum bound principle preserving scheme for solving a nonlocal ternary Allen–Cahn model, *Comput. Math. Appl.,* **155** (2024), 150–164. https://doi.org/10.1016/j.camwa.2023.06.018
3. M. Emamjomeh, M. Nabati, A. Dinmohammadi, Numerical study of two operator splitting localized radial basis function method for Allen–Cahn problem, *Eng. Anal. Bound. Elem.,* **163** (2024), 126–137. https://doi.org/10.1016/j.enganabound.2023.07.014
4. L. Q. Chen, Phase-field models for microstructure evolution, *Ann. Rev. Mater. Res.,* **32** (2002), 113–140. https://doi.org/10.1146/annurev.matsci.32.112001.132041
5. M. Fatima, R. P. Agarwal, M. Abbas, P. O. Mohammed, M. Shafiq, N. Chorfi, Extension of Cubic B-Spline for Solving the Time-Fractional Allen–Cahn Equation in the Context of Mathematical Physics, *Computation,* **12** (2024), 51. https://doi.org/10.3390/computation12030051
6. Z. Lu, J. Wang, A novel and efficient multi-scale feature extraction method for EEG classification, *AIMS Math.,* **9** (2024), 16605–16622. https://doi.org/10.3934/math.2024848
7. S. Kim, J. Kim, Automatic Binary Data Classification Using a Modified Allen–Cahn Equation, *Int. J. Pattern Recognit. Artif. Intell.,* **35** (2021), 2150013. https://doi.org/10.1142/S021800142150013X
8. D. Lee, S. Lee, Image segmentation based on modified fractional Allen–Cahn equation, *Math. Probl. Eng.,* **2019** (2019), 3980181. https://doi.org/10.1155/2019/8059716
9. M. Beneš, V. Chalupeck'y, K. Mikula, Geometrical image segmentation by the Allen–Cahn equation, *Appl. Numer. Math.,* **51** (2004), 187–205. https://doi.org/10.1016/j.apnum.2004.04.006
10. D. Lee, Gradient-descent-like scheme for the Allen–Cahn equation, *AIP Adv.,* **13** (2023), 085010. https://doi.org/10.1063/5.0154657
11. D. Lee, Computing the area-minimizing surface by the Allen–Cahn equation with the fixed boundary, *AIMS Math.,* **8** (2023), 23352–23371. https://doi.org/10.3934/math.20231184
12. J. Yang, Y. Li, C. Lee, Y. Choi, J. Kim, Fast evolution numerical method for the Allen–Cahn equation, *J. King Saud Univ. Sci.,* **35** (2023), 102430. https://doi.org/10.1016/j.jksus.2022.102430

13. B. Xia, R. Yu, X. Song, X. Zhang, J. Kim, An efficient data assimilation algorithm using the Allen–Cahn equation, *Eng. Anal. Bound. Elem.,* **155** (2023), 511–517. https://doi.org/10.1016/j.enganabound.2023.07.005

14. H. Zhang, X. Qian, S. Song, Third-order accurate, large time-stepping and maximum-principle-preserving schemes for the Allen–Cahn equation, *Numer. Algorithms,* **95** (2024), 1213–1250. https://doi.org/10.1007/s11075-023-01482-w

15. J. Sun, H. Zhang, X. Qian, S. Song, Up to eighth-order maximum-principle-preserving methods for the Allen–Cahn equation, *Numer. Algorithms,* **92** (2023), 1041–1062. https://doi.org/10.1007/s11075-022-01324-w

16. J. Choi, S. Ham, S. Kwak, Y. Hwang, J. Kim, Stability analysis of an explicit numerical scheme for the Allen–Cahn equation with high-order polynomial potentials, *AIMS Math.,* **9** (2024), 19332–19344. https://doi.org/10.3934/math.2024803

17. H. Kim, G. Lee, S. Kang, S. Ham, Y. Hwang, J. Kim, Hybrid numerical method for the Allen–Cahn equation on nonuniform grids, *Comput. Math. Appl.,* **158** (2024), 167–178. https://doi.org/10.1016/j.camwa.2023.07.010

18. X. Chen, X. Qian, S. Song, Fourth-order structure-preserving method for the conservative Allen–Cahn equation, *Adv. Appl. Math. Mech.,* **15** (2023), 159–181. https://doi.org/10.4208/aamm.OA-2021-0303

19. J. Rubinstein, P. Sternberg, Nonlocal reaction-diffusion equations and nucleation, *IMA J. Appl. Math.,* **48** (1992), 249–264. https://doi.org/10.1093/imamat/48.3.249

20. M. Brassel, E. Bretin, A modified phase field approximation for mean curvature flow with conservation of the volume, *Math. Meth. Appl. Sci.,* **10** (2011), 1157–1180. https://doi.org/10.1002/mma.1348

21. J. Yang, J. Kim, Efficient and structure-preserving time-dependent auxiliary variable method for a conservative Allen–Cahn type surfactant system, *Eng. Comput.,* **38** (2022), 5231–5250. https://doi.org/10.1007/s00366-021-01583-5

22. L. Bronsard, B. Stoth, Volume-preserving mean curvature flow as a limit of a nonlocal Ginzburg–Landau equation, *SIAM J. Math. Anal.,* **28** (1997), 769–807. https://doi.org/10.1137/S0036141096298974

23. X. Yang, J. J. Feng, C. Liu, J. Shen, Numerical simulations of jet pinching-off and drop formation using an energetic variational phase-field method, *J. Comput. Phys.,* **218** (2006), 417–428. https://doi.org/10.1016/j.jcp.2006.02.010

24. Z. Zhang, H. Tang, An adaptive phase field method for the mixture of two incompressible fluids, *Comput. Fluids,* **36** (2007), 1307–1318. https://doi.org/10.1016/j.compfluid.2006.10.001

25. B. Xia, Y. Li, Z. Li, Second-order unconditionally stable direct methods for Allen–Cahn and conservative Allen–Cahn equations on surfaces, *Mathematics,* **8** (2020), 1486. https://doi.org/10.3390/math8091486

26. Z. Sun, S. Zhang, A radial basis function approximation method for conservative Allen–Cahn equations on surfaces, *Appl. Math. Lett.,* **143** (2023), 108634. https://doi.org/10.1016/j.aml.2023.108634

27. X. Liu, Q. Hong, H. L. Liao, Y. Gong, A multi-physical structure-preserving method and its analysis for the conservative Allen–Cahn equation with nonlocal constraint, *Numer. Algorithms,* **97** (2024), 1–21. https://doi.org/10.1007/s11075-023-01502-9

28. J. Yang, J. Kim, Numerical study of incompressible binary fluids on 3D curved surfaces based on the conservative Allen–Cahn–Navier–Stokes model, *Comput. Fluids,* **228** (2021), 105094. https://doi.org/10.1016/j.compfluid.2021.105094

29. J. Yang, J. Kim, A phase-field model and its efficient numerical method for two-phase flows on arbitrarily curved surfaces in 3D space, *Comput. Meth. Appl. Mech. Eng.,* **372** (2020), 113382. https://doi.org/10.1016/j.cma.2020.113382

30. C. Lee, S. Kim, S. Kwak, Y. Hwang, S. Ham, S. Kang, J. Kim, Semi-automatic fingerprint image restoration algorithm using a partial differential equation, *AIMS Math.,* **8** (2023), 27528–27541. https://doi: 10.3934/math.20231408

31. Y. Choi, J. Kim, Maximum principle preserving and unconditionally stable scheme for a conservative Allen–Cahn equation, *Eng. Anal. Bound. Elem.,* **150** (2023), 111–119. https://doi.org/10.1016/j.enganabound.2023.05.005

32. J. Kim, S. Lee, Y. Choi, A conservative Allen–Cahn equation with a space–time dependent Lagrange multiplier, *Int. J. Eng. Sci.,* **84** (2014), 11–17. https://doi.org/10.1016/j.ijengsci.2014.06.004

33. Y. Hwang, J. Yang, G. Lee, S. Ham, S. Kang, S. Kwak, et al., Fast and efficient numerical method for solving the Allen–Cahn equation on the cubic surface, *Math. Comput. Simul.,* **215** (2024), 338–356. https://doi.org/10.1016/j.matcom.2023.07.024

34. Y. Wang, X. Xiao, X. Feng, Numerical simulation for the conserved Allen–Cahn phase field model of two-phase incompressible flows by an efficient dimension splitting method, *Commun. Nonlinear Sci. Numer. Simul.,* **131** (2024), 107874. https://doi.org/10.1016/j.cnsns.2024.107874

35. X. Yang, Efficient, second-order in time, and energy stable scheme for a new hydrodynamically coupled three components volume-conserved Allen–Cahn phase-field model, *Math. Models Meth. Appl. Sci.,* **31** (2021), 753–787. https://doi.org/10.1142/S0218202521500184

36. W. Cai, J. Ren, X. Gu, Y. Wang, Parallel and energy conservative/dissipative schemes for sine–Gordon and Allen–Cahn equations, *Comput. Meth. Appl. Mech. Eng.,* **425** (2024), 116938. https://doi.org/10.1016/j.cma.2024.116938

37. J. Kim, D. Jeong, S. D. Yang, Y. Choi, A finite difference method for a conservative Allen–Cahn equation on non-flat surfaces, *J. Comput. Phys.,* **334** (2017), 170–181. https://doi.org/10.1016/j.jcp.2016.12.060

AIMS Press