Appl. Numer. Math. 228 (2026) 71–96

Contents lists available at ScienceDirect

Applied Numerical Mathematics

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

Research Paper

A multigrid solver for the Allen–Cahn equation on a virtual cubic surface

Junxiang Yang [a], Soobin Kwak [b], Seokjun Ham [b], Youngjin Hwang [b], Hyundong Kim [c,d], Juho Ma [b], Junseok Kim [b,*]

[a] School of Computer Science and Engineering, Faculty of Innovation Engineering, Macau University of Science and Technology, Macau
[b] Department of Mathematics, Korea University, Seoul, 02841, Republic of Korea
[c] Department of Mathematics and Physics, Kangwon National University, Gangneung, 25457, Republic of Korea
[d] Smart infrastructure, Kangwon National University, Gangneung, 25457, Republic of Korea

ARTICLE INFO

Keywords:
Allen–Cahn equation
Virtual cubic surface
Multigrid method

ABSTRACT

We present an unconditionally stable hybrid finite difference scheme with a multigrid solver for the nonlinear Allen–Cahn (AC) model on a virtual cubic surface. Previous fully explicit operator splitting methods suffer from strict time step restrictions that limit computational efficiency. To overcome this, the diffusion term is discretized implicitly and solved using a V-cycle multigrid algorithm, while the nonlinear term is updated analytically in closed form. The proposed scheme incorporates boundary conditions reflecting the virtual cubic surface geometry and achieves second-order spatial accuracy and first-order temporal accuracy. Numerical tests confirm that the proposed algorithm preserves the discrete maximum principle and satisfies the energy dissipation property within a multigrid tolerance, even for large time steps beyond the explicit stability limit. Computational experiments include convergence analysis, validation of energy and maximum principle preservation, and simulations of motion by mean curvature and traveling wave propagation on the cubic surface. Computational results demonstrate that the scheme accurately captures interface dynamics, including curvature-driven motion and complex shape evolution, and maintains both stability and efficiency. The use of the multigrid method ensures fast convergence for the diffusion step, and the approach is well suited for large-scale simulations of phase-field models on curved or piecewise-planar surfaces. This work provides a practical and robust computational methodology for solving the nonlinear AC equation on the virtual surfaces of a rectangular cuboid, and it enables efficient simulation of interfacial phenomena with reduced computational cost and without restrictive time step constraints.

1. Introduction

The nonlinear Allen–Cahn (AC) model is a fundamental model to describe the anti-phase coarsening in material sciences. In interdisciplinary fields, the AC equation has been successfully used for motion by mean curvature [1], shape transformation [2,3], multiple components coupling [4,5], 3D reconstruction in fluid-structure interaction [6,7], etc. In this work, we develop an unconditionally stable hybrid scheme for the nonlinear AC model on a virtual cubic surface, see Fig. 1.

* Corresponding author.
E-mail address: cfdkim@korea.ac.kr (J. Kim).
URL: https://mathematicians.korea.ac.kr/cfdkim (J. Kim)

https://doi.org/10.1016/j.apnum.2026.05.010
Received 2 December 2025; Received in revised form 10 May 2026; Accepted 18 May 2026
Available online 20 May 2026

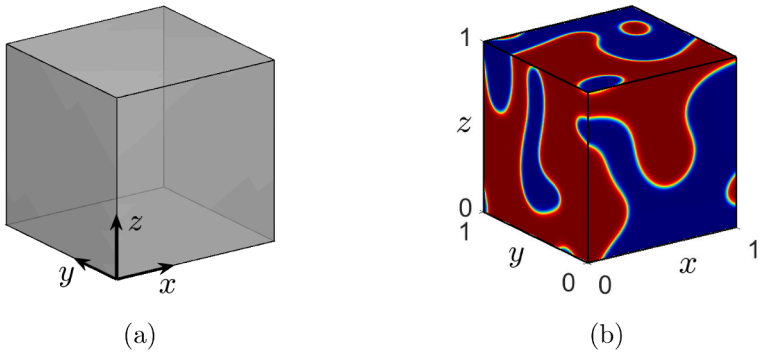

Fig. 1. (a) Virtual cubic surface domain Ω and (b) phase separation on the domain.

Significant research has been dedicated to developing and analyzing many computational techniques for solving the AC model on curved surfaces among the various phase field models. Zhang et al. [8] proposed a numerical method that combines Isogeometric Analysis (IGA) with the interior penalty discontinuous Galerkin method for solving the AC equation on three-dimensional (3D) surfaces. The proposed algorithm eliminates the need for time-consuming mesh generation and allows for easy refinements without altering the domain geometry. Kusdiantara et al. [9] numerically investigated the discrete AC equation with cubic and quintic nonlinearities on the Lieb lattice and analyzed localized solutions exhibiting linear multistability and hysteresis in the bifurcation diagram. They examined the homoclinic snaking structure and revealed the influence of the lattice geometry through numerical continuation based on the pseudo-arclength method. Xiao et al. [10] presented an unconditionally maximum-principle-preserving finite element method for the surface AC equation. The discrete maximum principle preservation of the algorithm for the surface AC equation was proved. For the surface case, Sun and Zhang [11] solved the conservative AC equation by using the proposed radial basis function approximation method. In [12], a numerical method for solving the AC equation on surfaces using generalized moving least squares and the closest point method was developed. Pan et al. [13] developed IGA with explicit-Invariant Energy Quadratization for the AC model on complex surfaces with triangular meshes.

Numerous studies have used cubed-sphere approximations in solving computational dynamics on curved spherical domains. Sjögreen [14] presented high-order numerical schemes for advection on the sphere. The advection model was solved using cubed-sphere mapping, which transforms a square into a patch on the sphere. The cubed-sphere is a technique for dividing a sphere into six faces, similar to an extended cube. Chen and Li [15] developed a general circulation model with a gnomonic equiangular cubed-sphere configuration to solve the computational fluid dynamics on the sphere. Chen [16] presented a Low Mach number Approximate Riemann Solver based finite volume method for solving the shallow-water equation on the cubed sphere grids with controllable variations of computational efficiency and diffusion properties. Zhang et al. [17] proposed a cubed-sphere grid based on a projection of the prolate-Gauss–Lobatto points. Lee and Kim [18] solved the time-fractional AC equation on geometric computational domains including surface of the cube.

In the previous study [19], the virtual cubic surface was unfolded into a planar domain and the fully explicit finite difference method (FDM) was applied. Therefore, there is a strict time step constraint for the stability of the fully explicit scheme [20]. To overcome this strict time step constraint for the stability of the fully explicit Euler scheme, we use the unconditionally stable hybrid scheme [21] which solves the diffusion term using the multigrid method [22] and solves the nonlinear term analytically. The multigrid solver is known as one of the fastest iterative techniques for solving the diffusion equation. Therefore, many studies have used multigrid methods to solve differential equations. Teunissen and Schiavello [23] solved the Poisson's equation with irregular boundaries using the multigrid method. They used the fully multigrid method but enhanced computational efficiency using adaptive mesh refinement. In [24], the authors solved the Poisson's equation using a compact scheme with sixth-order accuracy. They not only verified the convergence accuracy of the proposed method but also demonstrated its efficiency compared to existing iterative methods. Pan et al. [25] proposed a multigrid solver to solve the two-dimensional (2D) spatial fractional diffusion equation. They generalized the extrapolation cascadic multigrid method and used the Crank–Nicolson scheme for stable time integration. The proposed method demonstrated better performance than the conventional V-cycle multigrid method.

The main goal of this article is to propose a multigrid solver for an unconditionally stable hybrid FDM developed for the AC model on a virtual cubic surface.

This article is organized as follows: Section 2 introduces the AC model on a virtual cubic surface, Section 3 describes the computational solution algorithm, Section 4 presents the numerical analysis, Section 5 presents various computational experiments, and Section 6 provides the conclusions.

2. The AC equation of a virtual cubic surface

In this section, we describe the governing equation, the AC model, on a virtual cubic surface. To represent the virtual cubic surface domain in 3D, we define the six planar sub-surface domains:

$$\Omega_1 = \{(x,y,z) \in \mathbb{R}^3 | 0 < x, y < 1,\ z = 0\},\ \Omega_2 = \{(x,y,z) \in \mathbb{R}^3 | 0 < x, z < 1,\ y = 0\},$$

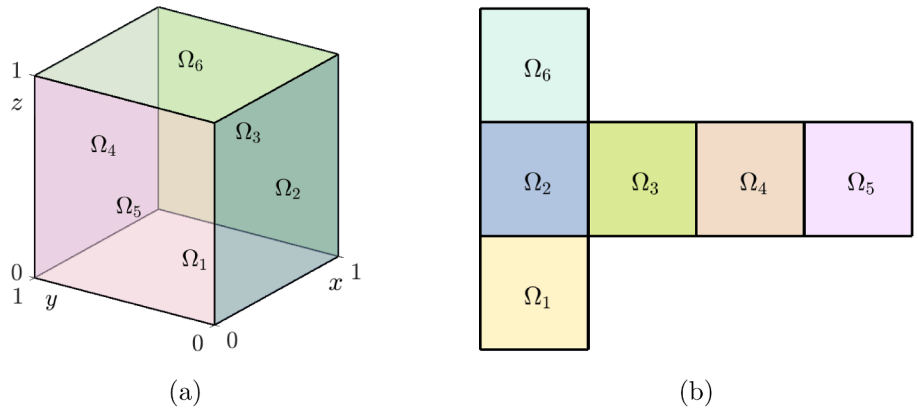

Fig. 2. Virtual cubic surface domain Ω.

$$\Omega_3 = \{(x,y,z) \in \mathbb{R}^3 | 0 < y,z < 1,\ x = 1\},\ \Omega_4 = \{(x,y,z) \in \mathbb{R}^3 | 0 < x,z < 1,\ y = 1\},$$
$$\Omega_5 = \{(x,y,z) \in \mathbb{R}^3 | 0 < y,z < 1,\ x = 0\},\ \Omega_6 = \{(x,y,z) \in \mathbb{R}^3 | 0 < x,y < 1,\ z = 1\},$$

which are shown in Fig. 2.

Then, the virtual cubic surface domain is given as

$$\Omega = \bigcup_{k=1}^{6} \Omega_k.$$

The AC equation defined on Ω at time t is expressed as

$$\frac{\partial u(x,y,z,t)}{\partial t} = -\frac{F'(u(x,y,z,t))}{\epsilon^2} + \Delta_v u(x,y,z,t), \quad (x,y,z) \in \Omega,\ t > 0, \tag{1}$$

where $u(x,y,z,t)$ is the concentration of a substance at point $(x,y,z) \in \Omega$, $F(u) = 0.25(u^2-1)^2$, and ϵ is the interfacial parameter. The initial condition is given by

$$u(x,y,z,0) = \phi(x,y,z), \qquad (x,y,z) \in \Omega, \tag{2}$$

where $\phi(x,y,z)$ is a prescribed function on Ω and satisfies $|\phi(x,y,z)| \le 1$. Since Ω is a closed virtual cubic surface, it has no physical boundary. Hence, no external boundary condition is imposed. However, our numerical discretization is performed on the unfolded two-dimensional representation of Ω, which is not closed and introduces artificial boundaries. Therefore, appropriate compatibility conditions must be imposed along the identified edges to ensure that the discrete solution remains single-valued on the original surface. The numerical boundary conditions are described in detail in Section 3, where we present the computational method.

The AC model can be formulated as an L^2-gradient flow of the Ginzburg–Landau energy, which yields the energy dissipation law $dE/dt \le 0$. Moreover, under periodic boundary conditions, the maximum bound principle $|u(x,y,z,t)| \le 1$ holds provided that $|u(x,y,z,0)| \le 1$ [26,27]. For completeness, we provide a proof of this maximum principle below [28]. We shall consider that $F(u)$ has a double-well form with minima at $u = \pm 1$. In particular,

$$F'(1) = F'(-1) = 0, \tag{3}$$

and F' satisfies the following monotonicity conditions outside $(-1,1)$:

$$F'(u) < 0,\ \forall u \in (-\infty,-1), \qquad F'(u) > 0,\ \forall u \in (1,\infty). \tag{4}$$

If the initial condition and the boundary condition satisfy

$$|u(x,y,z,0)| \le 1 \text{ in } \Omega, \qquad |u(x,y,z,t)| \le 1 \text{ on } \partial\Omega \times (0,T], \tag{5}$$

then the solution satisfies

$$\|u(x,y,z,t)\|_\infty \le 1, \qquad \forall t \in (0,T]. \tag{6}$$

Proof. We first prove the upper bound $u \le 1$.

Step 1: Upper bound. Define $w := (u-1)_+ = \max\{u-1, 0\}$. Multiply Eq. (1) by w and integrate over Ω. The chain rule yields

$$\int_\Omega u_t\, w\, dx = \int_{\{w>0\}} u_t\, w\, dx = \int_{\{w>0\}} w_t\, w\, dx = \frac{1}{2}\frac{d}{dt}\int_\Omega w^2\, dx. \tag{7}$$

Consequently,

$$\frac{1}{2}\frac{d}{dt}\int_\Omega w^2\, dx = \int_\Omega \left(\Delta u - \frac{F'(u)}{\epsilon^2}\right) w\, dx. \tag{8}$$

Diffusion term. Integration by parts together with the admissible boundary conditions yields

$$\int_\Omega \Delta u\, w\, dx = -\int_\Omega \nabla u \cdot \nabla w\, dx = -\int_\Omega |\nabla w|^2\, dx \le 0.$$

Reaction term. On $\{w > 0\}$ one has $u > 1$, and Eq. (4) implies $F'(u) > 0$. Therefore,

$$-\int_\Omega \frac{F'(u)}{\epsilon^2}\, w\, dx \le 0.$$

Substituting the two estimates into (8) yields

$$\frac{1}{2}\frac{d}{dt}\int_\Omega w^2\, dx \le 0.$$

Since (5) implies $w(x, y, z, 0) = 0$, one obtains $\int_\Omega w^2\, dx \equiv 0$ for all $t \in (0, T]$. Hence $w \equiv 0$, which yields $u \le 1$ in $\Omega \times (0, T]$.

Step 2: Lower bound. Define $z := (-1-u)_+ = \max\{-1-u, 0\}$. Repeat the previous argument with test function z. The chain rule yields

$$\int_\Omega u_t\, z\, dx = \int_{\{z>0\}} u_t\, z\, dx = \int_{\{z>0\}} z_t\, z\, dx = \frac{1}{2}\frac{d}{dt}\int_\Omega z^2\, dx. \tag{9}$$

Consequently,

$$\frac{1}{2}\frac{d}{dt}\int_\Omega z^2\, dx = \int_\Omega \left(\Delta u - \frac{F'(u)}{\epsilon^2}\right) z\, dx. \tag{10}$$

Diffusion term. Integration by parts and the admissible boundary conditions give

$$\int_\Omega \Delta u\, z\, dx = -\int_\Omega \nabla u \cdot \nabla z\, dx = -\int_\Omega |\nabla z|^2\, dx \le 0.$$

Reaction term. On $\{z > 0\}$ one has $u < -1$, and (4) implies $F'(u) < 0$. This gives

$$-\int_\Omega \frac{F'(u)}{\epsilon^2}\, w\, dx \le 0.$$

Substituting the two estimates into (10) yields

$$\frac{1}{2}\frac{d}{dt}\int_\Omega z^2\, dx \le 0.$$

Since (5) implies $z(x, y, z, 0) = 0$, one obtains $z \equiv 0$, and thus $u \ge -1$ in $\Omega \times (0, T]$. Combining the two bounds yields (6). For further details, we refer to Sections 7.1 and 9.3 of [28]. □

Let $E(\phi)$ denote the classical Ginzburg–Landau energy considered in this work. For the fully discrete scheme induced by the Lie (operator-splitting) time discretization, a direct proof of the stepwise decay $E(\phi^{n+1}) \le E(\phi^n)$ is not available at present. We therefore adopt a modified-energy viewpoint that is standard in recent analyses of splitting methods for AC-type gradient flows. More precisely, we introduce a time-step-dependent functional $\widetilde{E}_\tau$ such that

$$\widetilde{E}_\tau(\phi^{n+1}) \le \widetilde{E}_\tau(\phi^n), \qquad \widetilde{E}_\tau(\phi) = E(\phi) + O(\tau). \tag{11}$$

For the scalar AC model, Li et al. [29] established unconditional stability and strict dissipation of an explicitly constructed $\widetilde{E}_\tau$, and they showed that $\widetilde{E}_\tau$ approximates the classical energy to $O(\tau)$. Since the present Ginzburg–Landau gradient flow has the same L^2-gradient flow structure, we use the same perspective.

In 3D, we consider a virtual Laplace operator $\Delta_v \phi(x, y, z) = \phi_{xx}(x, y, z) + \phi_{yy}(x, y, z) + \phi_{zz}(x, y, z)$ on a virtual cubic surface. Since the domain is a unit virtual cubic surface, we obtain the second partial derivatives for Ω as follows:

$$\frac{\partial^2 \phi(x,y,z)}{\partial x^2} = \begin{cases} \lim\limits_{h\to 0} \dfrac{\phi(x-h,y,0) - 2\phi(x,y,0) + \phi(x+h,y,0)}{h^2}, & (x,y,z) \in \Omega_1, \\ \lim\limits_{h\to 0} \dfrac{\phi(x-h,0,z) - 2\phi(x,0,z) + \phi(x+h,0,z)}{h^2}, & (x,y,z) \in \Omega_2, \\ 0, & (x,y,z) \in \Omega_3, \\ \lim\limits_{h\to 0} \dfrac{\phi(x-h,1,z) - 2\phi(x,1,z) + \phi(x+h,1,z)}{h^2}, & (x,y,z) \in \Omega_4, \\ 0, & (x,y,z) \in \Omega_5, \\ \lim\limits_{h\to 0} \dfrac{\phi(x-h,y,1) - 2\phi(x,y,1) + \phi(x+h,y,1)}{h^2}, & (x,y,z) \in \Omega_6, \end{cases}$$

$$\frac{\partial^2 \phi(x,y,z)}{\partial y^2} = \begin{cases} \lim\limits_{h\to 0} \dfrac{\phi(x,y-h,0) - 2\phi(x,y,0) + \phi(x,y+h,0)}{h^2}, & (x,y,z) \in \Omega_1, \\ 0, & (x,y,z) \in \Omega_2, \\ \lim\limits_{h\to 0} \dfrac{\phi(1,y-h,z) - 2\phi(1,y,z) + \phi(1,y+h,z)}{h^2}, & (x,y,z) \in \Omega_3, \\ 0, & (x,y,z) \in \Omega_4, \\ \lim\limits_{h\to 0} \dfrac{\phi(0,y-h,z) - 2\phi(0,y,z) + \phi(0,y+h,z)}{h^2}, & (x,y,z) \in \Omega_5, \\ \lim\limits_{h\to 0} \dfrac{\phi(x,y-h,1) - 2\phi(x,y,1) + \phi(x,y+h,1)}{h^2}, & (x,y,z) \in \Omega_6, \end{cases}$$

$$\frac{\partial^2\phi(x,y,z)}{\partial z^2} = \begin{cases} 0, & (x,y,z)\in\Omega_1,\\ \lim\limits_{h\to 0}\dfrac{\phi(x,0,z-h)-2\phi(x,0,z)+\phi(x,0,z+h)}{h^2}, & (x,y,z)\in\Omega_2,\\ \lim\limits_{h\to 0}\dfrac{\phi(1,y,z-h)-2\phi(1,y,z)+\phi(1,y,z+h)}{h^2}, & (x,y,z)\in\Omega_3,\\ \lim\limits_{h\to 0}\dfrac{\phi(x,1,z-h)-2\phi(x,1,z)+\phi(x,1,z+h)}{h^2}, & (x,y,z)\in\Omega_4,\\ \lim\limits_{h\to 0}\dfrac{\phi(0,y,z-h)-2\phi(0,y,z)+\phi(0,y,z+h)}{h^2}, & (x,y,z)\in\Omega_5,\\ 0, & (x,y,z)\in\Omega_6, \end{cases}$$

The conventional interpretation for the derivation of the AC equation is based on the following functional:

$$\mathcal{E}(u) = \int_\Omega \left(\frac{F(u)}{\epsilon^2} + \frac{1}{2}|\nabla u|^2\right)\mathrm{d}\mathbf{x}.$$

3. Numerical solution algorithm

This study provides a detailed description of an unconditionally stable hybrid scheme for solving the AC model on a virtual cubic surface domain. The solution approach is based on the operator splitting method (OSM), in which the AC model is decomposed into linear and nonlinear subproblems [30].

$$\frac{\partial u(x,y,t)}{\partial t} = \Delta u(x,y,t), \tag{12}$$

$$\frac{\partial u(x,y,t)}{\partial t} = -\frac{u^3(x,y,t)-u(x,y,t)}{\epsilon^2}. \tag{13}$$

We provide a detailed description of the numerical algorithm used to solve the AC model (1) defined on the unfolded unit virtual cubic surface domain. Let $\hat{\Omega}$ be the unfolded virtual cubic surface domain Ω in 2D; then $\hat{\Omega}$ is divided into the following subdomains.

$$\begin{aligned}
\hat{\Omega}_1 &= \{(x,-y) \mid (x,y,0)\in\Omega_1\} = (0,1)\times(0,1),\\
\hat{\Omega}_2 &= \{(x,z+1) \mid (x,0,z)\in\Omega_2\} = (0,1)\times(1,2),\\
\hat{\Omega}_3 &= \{(y+1,z+1) \mid (1,y,z)\in\Omega_3\} = (1,2)\times(1,2),\\
\hat{\Omega}_4 &= \{(3-x,z+1) \mid (x,1,z)\in\Omega_4\} = (2,3)\times(1,2),\\
\hat{\Omega}_5 &= \{(4-y,z+1) \mid (0,y,z)\in\Omega_5\} = (3,4)\times(1,2),\\
\hat{\Omega}_6 &= \{(x,y+2) \mid (x,y,1)\in\Omega_6\} = (0,1)\times(2,3),\\
\hat{\Omega} &= \bigcup_{k=1}^{6}\hat{\Omega}_k.
\end{aligned}$$

Let $N = 2^g$, where g is a positive integer. Then, the spatial step size is $h = 1/N$. Define index sets for discretization as

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

Then, we discretize and redefine the domain $\hat{\Omega}$ as follows:

$$\begin{aligned}
\hat{\Omega}_k &= \{(x_i,y_j) \mid (i,j)\in I_k^g\},\ \text{for } k=1,2,\dots,6,\\
\hat{\Omega} &= \bigcup_{k=1}^{6}\hat{\Omega}_k,
\end{aligned}$$

where $x_i = (i-0.5)h$ and $y_j = (j-0.5)h$. Let $u_{ij}^n = u(x_i,y_j,n\Delta t)$, where Δt is the size of the temporal step. By using the fully implicit method and the second order discrete Laplacian operator, the linear Eq. (12) is discretized as follows:

$$\begin{aligned}
\frac{u_{ij}^{n+1}-u_{ij}^n}{\Delta t} &= \Delta u_{ij}^{n+1}\\
&= \frac{u_{i+1,j}^{n+1}+u_{i-1,j}^{n+1}+u_{i,j+1}^{n+1}+u_{i,j-1}^{n+1}-4u_{ij}^{n+1}}{h^2},\ (i,j)\in I^g.
\end{aligned} \tag{14}$$

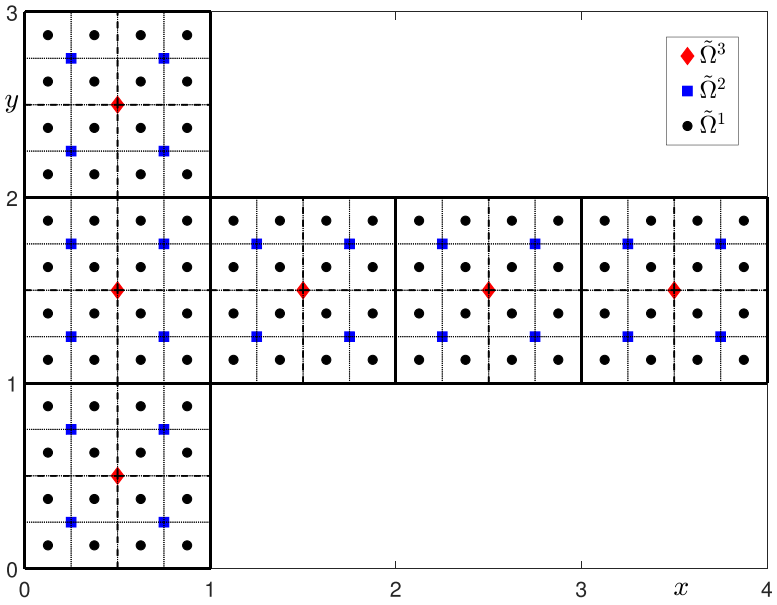

Fig. 3. Schematic for the discrete domain $\tilde{\Omega}^g$, $g = 1, 2, 3$.

For the fully implicit discretization (14) of the diffusion subproblem, a Fourier mode assumption on a uniform grid yields the amplification factor

$$\lambda(\xi,\eta) = \frac{1}{1 + \frac{4\Delta t}{h^2}\left(\sin^2\left(\frac{\xi h}{2}\right) + \sin^2\left(\frac{\eta h}{2}\right)\right)}.$$

Hence $0 < \lambda(\xi,\eta) \le 1$ for any $\Delta t > 0$, which implies that the diffusion step defined by (14) is unconditionally stable [31,32]. Boundary conditions are defined for each $\hat{\Omega}_k$ for $k = 1, 2, \dots, 6$ as:

$$\begin{aligned}
&\text{For } m = 1, 2, \dots, N, \qquad (15)\\
&\hat{\Omega}_1 \ : \ u^{n+1}_{0,m} = u^{n+1}_{3N+m,N+1},\ u^{n+1}_{m,0} = u^{n+1}_{3N+1-m,N+1},\ u^{n+1}_{N+1,m} = u^{n+1}_{2N+1-m,N+1},\\
&\hat{\Omega}_2 \ : \ u^{n+1}_{0,N+m} = u^{n+1}_{4N,N+m},\\
&\hat{\Omega}_3 \ : \ u^{n+1}_{N+m,N} = u^{n+1}_{N,N+1-m},\ u^{n+1}_{N+m,2N+1} = u^{n+1}_{N,2N+m},\\
&\hat{\Omega}_4 \ : \ u^{n+1}_{2N+m,N} = u^{n+1}_{N+1-m,1},\ u^{n+1}_{2N+m,2N+1} = u^{n+1}_{N+1-m,3N},\\
&\hat{\Omega}_5 \ : \ u^{n+1}_{3N+m,N} = u^{n+1}_{1,m},\ u^{n+1}_{3N+m,2N+1} = u^{n+1}_{1,3N+1-m},\ u^{n+1}_{4N+1,N+m} = u^{n+1}_{1,N+m},\\
&\hat{\Omega}_6 \ : \ u^{n+1}_{0,2N+m} = u^{n+1}_{4N+1-m,2N},\ u^{n+1}_{m,3N+1} = u^{n+1}_{3N+1-m,2N},\ u^{n+1}_{N+1,2N+m} = u^{n+1}_{N+m,2N}.
\end{aligned}$$

To solve the linear Eq. (12), we use the linear multigrid method [33]. To clearly explain the steps involved in a single V-cycle, we focus on the computational solution on the discrete unfolded virtual cubic surface domain with $N = 8$. We define the discrete domains for the linear multigrid method as follows:

$$\begin{aligned}
&\text{For } k = 1, 2, \dots, 6, \quad \text{and } g = 1, 2, 3,\\
&\tilde{\Omega}^g_k = \{(x_i, y_j) \mid x_i = (i - 0.5)h_g,\ y_j = (j - 0.5)h_g,\ (i,j) \in I^g_k\},\\
&\tilde{\Omega}^g = \bigcup_{k=1}^{6} \tilde{\Omega}^g_k,
\end{aligned}$$

where $h_g = 2^{3-g}h$. Each discrete domain $\tilde{\Omega}^g_k$ has a finer grid than $\tilde{\Omega}^{g-1}_k$, with a half grid step. Fig. 3 shows the grid points of discrete domains $\tilde{\Omega}^g$ for $g = 1, 2, 3$ used in the linear multigrid method.

The multigrid method uses discrete computational domains with varying grid resolutions. Within this method, a pointwise Gauss–Seidel relaxation method serves as the smoothing operator. We define the linear operator $\mathcal{L}^g$ and the source term f_{ij} as

$$\mathcal{L}^g(u^{n+1}_{ij}) = \frac{u^{n+1}_{ij}}{\Delta t} - \frac{u^{n+1}_{i+1,j} + u^{n+1}_{i-1,j} + u^{n+1}_{i,j+1} + u^{n+1}_{i,j-1} - 4u^{n+1}_{ij}}{h^2}, \qquad (16)$$

$$f_{ij} = \frac{u^{n}_{ij}}{\Delta t}, \qquad (17)$$

on the discrete domain $\tilde{\Omega}^g$. We solve a numerical solution u_{ij}^{n+1} that satisfies Eq. (12) for a given u_{ij}^n on $\hat{\Omega}$. We describe the multigrid algorithim for solving the discrete linear Eq. (14). We use the multigrid method, which is of the V-cycle type. The initial condition on $\tilde{\Omega}^g$ for the V-cycle is the numerical solution on $\hat{\Omega}$ from the previous time step. Let $g = 3$ and let $u_{ij}^{n+1,g,0} = u_{ij}^n$ on $\tilde{\Omega}^g$.

V-cycle

$$u_{ij}^{n+1,g,m+1} = \mathrm{MGC}(g, u_{ij}^{n+1,g,m}, \mathcal{L}^g, f_{ij}, \nu_1, \nu_2),$$

where m is the number of V-cycle iterations, $u_{ij}^{n+1,g,m}$ is an approximation of u_{ij}^{n+1} on $\tilde{\Omega}^g$ before an MGC, $u_{ij}^{n+1,g,m+1}$ is the approximation of u_{ij}^{n+1} on $\tilde{\Omega}^g$ after an MGC, ν_1 is the number of pre-smoothing steps, and ν_2 is the number of post-smoothing steps.

We describe the details of one cycle of MGC.

Step 1) Pre-smoothing We calculate $\bar{u}_{ij}^{n+1,g,m}$ with the number of smoothing steps ν_1 to $u_{ij}^{n+1,g,m}$:

$$\bar{u}_{ij}^{n+1,g,m} = S^{\nu_1}(u_{ij}^{n+1,g,m}, \mathcal{L}^g, f_{ij}),$$

where S^{ν_1} is relaxation operator with number of pre-smoothing steps ν_1, and f_{ij} is source term. That is, we obtain the approximation of $\bar{u}_{ij}^{n+1,g,m}$. For the relaxation operator S, we use the Gauss–Seidel iterative scheme. By Eqs. (14), (16), and (17), we obtain the equation

$$\mathcal{L}^g(u_{ij}^{n+1}) = f_{ij} \text{ on } \tilde{\Omega}^g. \tag{18}$$

We can rewrite Eq. (18) as follows

$$\bar{u}_{ij}^{n+1} = \left(f_{ij} + \frac{u_{i+1,j}^{n+1} + \bar{u}_{i-1,j}^{n+1} + u_{i,j+1}^{n+1} + \bar{u}_{i,j-1}^{n+1}}{h^2} \right) \Bigg/ \left(\frac{1}{\Delta t} + \frac{4}{h^2} \right) \text{ on } \tilde{\Omega}^g, \tag{19}$$

where $\bar{u}_{i-1,j}^{n+1}$ and $\bar{u}_{i,j-1}^{n+1}$ are values post-smoothed by Eq. (19). Thus, the relaxation operator with one smoothing step in the V-cycle of the multi-grid method consists of solving Eq. (19) for $(i, j) \in I^g$.

Step 2) Coarse grid correction

- Calculate the defect: $\bar{d}_{ij}^{g,m} = f_{ij} - \mathcal{L}^g(\bar{u}_{ij}^{n+1,g,m})$, for $(i, j) \in I^g$.
- Restrict the $\bar{d}_{ij}^{g,m}$ and $\bar{u}_{ij}^{g,m}$: $\bar{d}_{ij}^{g-1,m} = R_{g-1}^g \bar{d}_{ij}^{g,m}$, for $(i, j) \in I^{g-1}$, where

$$\begin{aligned} d_{ij}^{g-1} &= R_{g-1}^g d_{ij}^g \\ &= \frac{1}{4}\left[d_{2i-1,2j-1}^g + d_{2i-1,2j}^g + d_{2i,2j-1}^g + d_{2i,2j}^g \right], \text{ for } (i, j) \in I^{g-1}. \end{aligned}$$

- Calculate an approximate solution $\hat{u}_{ij}^{n+1,g-1,m}$ of the coarse grid equation on Ω^{g-1}, i.e.,

$$\mathcal{L}^{g-1}(u_{ij}^{n+1,g-1,m}) = \bar{d}_{ij}^{g-1,m}. \tag{20}$$

We apply a fast iteration solver for Eq. (20). When $g > 1$, Eq. (20) is solved approximately by executing g-grid cycles, starting with the zero grid function as the initial guess:

$$v_{ij}^{n+1,g-1,m} = \mathrm{MGcycle}(g-1, 0, \mathcal{L}^{g-1}, \bar{d}_{ij}^{g-1,m}, \nu_1, \nu_2).$$

Subsequently, the coarse grid values are directly assigned to the four adjacent fine grid points.

- Interpolate the correction: $v_{2i-1,2j-1}^{n+1,g,m} = v_{2i-1,2j}^{n+1,g,m} = v_{2i,2j-1}^{n+1,g,m} = v_{2i,2j}^{n+1,g,m} = v_{ij}^{n+1,g-1,m}$, for $(i, j) \in I^{g-1}$.
- Calculate the corrected approximation on $\tilde{\Omega}^g$: $\hat{u}_{ij}^{g,m} = \bar{u}_{ij}^{n+1,g,m} + v_{ij}^{n+1,g,m}$.

Step 3) Post-smoothing

$$u_{ij}^{n+1,g,m+1} = S^{\nu_2}(\hat{u}_{ij}^{g,m}, \mathcal{L}^g, f_{ij}),$$

where S^{ν_2} is relaxation operator with the number of post-smoothing steps ν_2. This completes the description of an MGC. We define the l_2-norm as

$$\|u\|_2 = \sqrt{\frac{1}{6N^2} \sum_{(i,j)\in I^g} u_{ij}^2}.$$

We repeat the V-cycle until the l_2-error $\|u^{n+1,g,m+1} - u^{n+1,g,m}\|_2$ for a V-cycle iteration is less than the given tolerance. Let M be the final number of V-cycle iterations, and define the $u_{ij}^* = u_{ij}^{n+1,g,M+1}$ for $(i, j) \in I^g$.

Next, we solve Eq. (13) analytically. That is, we solve $\partial\psi/\partial t = (\psi - \psi^3)/\epsilon^2$ with the initial condition $\psi(x_i, y_j, 0) = u_{ij}^*$ for $(i, j) \in I^g$ using the separation of variables method [34] to get $u_{ij}^{n+1} = \psi(x_i, y_j, \Delta t)$.

$$u_{ij}^{n+1} = \frac{u_{ij}^*}{\sqrt{e^{\frac{-2\Delta t}{\epsilon^2}} + (u_{ij}^*)^2 \left(1 - e^{\frac{-2\Delta t}{\epsilon^2}}\right)}}. \tag{21}$$

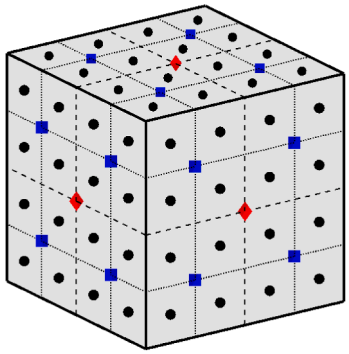

Fig. 4. Schematic of diagram for the discrete folded virtual cubic surface domain Ω^3.

Finally, to express the computational solution of the AC equation on the folded virtual cubic surface in terms of the discrete computational solution calculated on the unfolded virtual cubic surface, we define the discretized domain of the folded cube surface as a mapping from the discretized domain of the unfolded cube surface as follows

$$\begin{aligned}
&\Omega_1^g = \{(x,-y,0) \mid (x,y) \in \hat{\Omega}_1\},\ \Omega_2^g = \{(x,0,y-1) \mid (x,y) \in \hat{\Omega}_2\},\\
&\Omega_3^g = \{(1,x-1,y-1) \mid (x,y) \in \hat{\Omega}_3\},\ \Omega_4^g = \{(3-x,1,y-1) \mid (x,y) \in \hat{\Omega}_4\},\\
&\Omega_5^g = \{(0,4-x,y-1) \mid (x,y) \in \hat{\Omega}_5\},\ \Omega_6^g = \{(x,y-2,1) \mid (x,y) \in \hat{\Omega}_6\},\\
&\Omega^g = \bigcup_{k=1}^{6} \Omega_k^g.
\end{aligned}$$

The schematic diagram of the grid points of discrete folded virtual cubic surfaces is shown in Fig. 4.

4. Numerical analysis

To facilitate the following numerical analysis, the fully discrete computational scheme is summarized as follows

$$\frac{u_{ij}^* - u_{ij}^n}{\Delta t} = \Delta_h u_{ij}^* = \frac{u_{i+1,j}^* + u_{i-1,j}^* + u_{i,j+1}^* + u_{i,j-1}^* - 4u_{ij}^*}{h^2}, \tag{22}$$

$$u_{ij}^{n+1} = \frac{u_{ij}^*}{\sqrt{e^{\frac{-2\Delta t}{\epsilon^2}} + (u_{ij}^*)^2\left(1 - e^{\frac{-2\Delta t}{\epsilon^2}}\right)}}, \tag{23}$$

for each point $(x_i, y_j) \in \hat{\Omega}$. Based on the boundary conditions in Eq. (15), it is easy to observe that the phase-field value at the grid-center point connected to one of the edges of the subregion $\hat{\Omega}_k$ equals the phase-field value at the grid center point that is connected to a certain edge in another subregion $\hat{\Omega}_l$ ($l \neq k$). Therefore, the following summation by parts holds

$$(\Delta_h u, v)_h = -(\nabla_d u, \nabla_d, v)_e,$$

where $(u,v)_h = h^2 \sum_{(x_i,y_j)\in\hat{\Omega}} u_{ij} v_{ij}$ and

$$\begin{aligned}
&(\nabla_d u, \nabla_d v)_e = h^2\Big(\sum_{(x_i,y_j)\in\hat{\Omega}} D_x u_{i+\frac{1}{2},j}\, D_x v_{i+\frac{1}{2},j} + \sum_{(x_i,y_j)\in\hat{\Omega}} D_y u_{i,j+\frac{1}{2}}\, D_y v_{i,j+\frac{1}{2}} \Big),\\
&D_x u_{i+\frac{1}{2},j} = \frac{u_{i+1,j} - u_{ij}}{h}, \quad D_y u_{i,j+\frac{1}{2}} = \frac{u_{i,j+1} - u_{ij}}{h}.
\end{aligned}$$

4.1. Maximum principle

Theorem 4.1. *Assume that the numerical solution u^n satisfies the maximum principle (i.e., $\|u^n\|_\infty = \max_{(x_i,y_j)\in\hat{\Omega}} |u_{ij}^n| \le 1$), the numerical solution u^{n+1} calculated with Eqs. (22) and (23) still satisfies the maximum principle (i.e. $\|u^{n+1}\|_\infty \le 1$).*

Proof. Recasting Eq. (22), we get

$$(1+4r)u_{ij}^* - r(u_{i+1,j}^* + u_{i-1,j}^*) - r(u_{i,j+1}^* + u_{i,j-1}^*) = u_{ij}^n, \tag{24}$$

where $r = \Delta t/h^2$. Let $M = \max_{(x_i,y_j)\in\hat{\Omega}} u_{ij}^*$, we need to prove $M \le 1$. We first assume $M > 1$ and let $u_{pq}^* = M$. From Eq. (24), we get

$$(1+4r)u_{pq}^* - r(u_{p+1,q}^* + u_{p-1,q}^*) - r(u_{p,q+1}^* + u_{p,q-1}^*) = u_{pq}^n. \tag{25}$$

Since M is the maximum value, we have $u^*_{p+1,q} \le M$, $u^*_{p-1,q} \le M$, $u^*_{p,q+1} \le M$, and $u^*_{p,q-1} \le M$. The above equation can lead to the following inequality

$$(1+4r)M - r(M+M) - r(M+M) = M \le u^n_{pq}. \tag{26}$$

The conclusion $u^n_{pq} \ge M > 1$ contradicts with the condition $\|u^n\|_\infty \le 1$. Therefore, the assumption $M > 1$ does not hold, and we have $u^*_{ij} \le 1$ for each point (x_i, y_j) in $\hat{\Omega}$.

Next, we set $m = \min_{(x_i,y_j)\in\hat{\Omega}} u^*_{ij}$, assume $m < -1$ and let $u^*_{st} = m$. Eq. (24) leads to

$$(1+4r)u^*_{st} - r(u^*_{s+1,t} + u^*_{s-1,t}) - r(u^*_{s,t+1} + u^*_{s,t-1}) = u^n_{st}. \tag{27}$$

Since $u^*s+1, t \ge m$, $u^*s-1, t \ge m$, $u^*s, t+1 \ge m$, and $u^*s, t-1 \ge m$, we further get the following inequality

$$(1+4r)m - r(m+m) - r(m+m) = m \ge u^n_{st}. \tag{28}$$

The conclusion $u^n_{st} \le m < -1$ contradicts with the condition $\|u^n\|_\infty \le 1$. Therefore the assumption $m < -1$ does not hold, and we have $u^*_{ij} \ge -1$ for each point (x_i, y_j) in $\hat{\Omega}$. Based on the above mentioned discussions, we derive $\|u^*\|_\infty \le 1$.

With $\|u^*\|_\infty \le 1$, we consider the square of the denominator of Eq. (23) as follows

$$e^{\frac{-2\Delta t}{\epsilon^2}} + (u^*_{ij})^2\left(1 - e^{\frac{-2\Delta t}{\epsilon^2}}\right) = \left(1 - (u^*_{ij})^2\right)e^{\frac{-2\Delta t}{\epsilon^2}} + (u^*_{ij})^2. \tag{29}$$

Since $1 - (u^*_{ij})^2 \ge 0$ and $e^{\frac{-2\Delta t}{\epsilon^2}} > 0$, the right-hand side of the above equation leads to

$$\left(1 - (u^*_{ij})^2\right)e^{\frac{-2\Delta t}{\epsilon^2}} + (u^*_{ij})^2 \ge (u^*_{ij})^2. \tag{30}$$

Taking the square root on the above inequality, we have

$$\sqrt{e^{\frac{-2\Delta t}{\epsilon^2}} + (u^*_{ij})^2\left(1 - e^{\frac{-2\Delta t}{\epsilon^2}}\right)} \ge |u^*_{ij}|. \tag{31}$$

Thus, we derive the following inequality

$$|u^{n+1}_{ij}| = \frac{|u^*_{ij}|}{\sqrt{e^{\frac{-2\Delta t}{\epsilon^2}} + (u^*_{ij})^2\left(1 - e^{\frac{-2\Delta t}{\epsilon^2}}\right)}} \le 1. \tag{32}$$

The above result indicates $\|u^{n+1}\|_\infty \le 1$. The proof of the maximum principle is completed. □

4.2. Error estimate

In a virtual cubic domain Ω, we consider the following AC equation

$$\frac{\partial u}{\partial t} = -\frac{F'(u)}{\epsilon^2} + \Delta u. \tag{33}$$

The solution of the above equation can be evolved in time in two substeps

$$u(t_{n+1}) = S^N(\Delta t)S^L(\Delta t)u(t_n). \tag{34}$$

Here,

$$S^L : \frac{\partial u}{\partial t} = \Delta u, \quad S^N : \frac{\partial u}{\partial t} = -\frac{F'(u)}{\epsilon^2}.$$

Based on the proposed numerical scheme, the discrete versions of S^L and S^N are defined as

$$S^L_h : \frac{u^*_{ij} - u^n_{ij}}{\Delta t} = \frac{u^*_{i+1,j} + u^*_{i-1,j} + u^*_{i,j+1} + u^*_{i,j-1} - 4u^*_{ij}}{h^2}, \tag{35}$$

$$S^N_h : u^{n+1}_{ij} = \frac{u^*_{ij}}{\sqrt{e^{\frac{-2\Delta t}{\epsilon^2}} + (u^*_{ij})^2\left(1 - e^{\frac{-2\Delta t}{\epsilon^2}}\right)}}. \tag{36}$$

The grid function space on $\hat{\Omega}$ is defined as

$$W_h = \{U \mid U = \{u_{ij}\},\ (x_i, y_j) \in \hat{\Omega}\}. \tag{37}$$

Let $\tilde{u}$ denote the exact solution of Eq. (34). Define a mapping I^h: $H^2(\Omega) \to W_h$ by $I^h(u) = U$.

Lemma 1. *For any grid function $U \in W_h$, the following inequality holds*

$$\|S_h^L U\|_h \le \|U\|_h, \tag{38}$$

where $\|\cdot\|_h$ denotes the discrete L^2 norm, i.e., $\|\cdot\|_h = \sqrt{h^2 \sum_{(x_i,y_j)\in\hat{\Omega}} |\cdot|^2}$.

Proof. Substituting U into the form of Eq. (22), we get

$$\frac{U^* - U^n}{\Delta t} = \Delta_h U^*. \tag{39}$$

Multiplying the above equation with U^* and taking the discrete L^2 inner product, we have

$$(U^* - U^n, U^*)_h = \Delta t(\Delta_h U^*, U^*)_h = -\Delta t(\nabla_d U^*, \nabla_d, U^*)_e \le 0. \tag{40}$$

From the above inequality, we get

$$(U^*, U^*)_h - (U^n, U^*)_h \le 0, \quad i.e., \ \|U^*\|_h^2 \le (U^n, U^*)_h. \tag{41}$$

Using the Cauchy–Schwarz inequality, we have

$$(U^n, U^*)_h \le \|U^n\|_h \|U^*\|_h. \tag{42}$$

Thus, we derive $\|U^*\|_h^2 \le \|U^n\|_h \|U^*\|_h$. If $\|U^*\|_h = 0$, this inequality holds. If $\|U^*\|_h \neq 0$, this inequality can be recast as $\|U^*\|_h \le \|U^n\|_h$. Here, $U^* = S_h^L U^n$. The lemma 1 is proved by substituting U^n with U. □

Lemma 2. *For any grid function $U \in W_h$, the following inequality holds*

$$\|S_h^N U\|_h \le e^{\frac{\Delta t}{\epsilon^2}} \|U\|_h. \tag{43}$$

Proof. Taking the square on Eq. (23), we get

$$\left(u_{ij}^{n+1}\right)^2 = \frac{\left(u_{ij}^*\right)^2}{e^{\frac{-2\Delta t}{\epsilon^2}} + (u_{ij}^*)^2\left(1 - e^{\frac{-2\Delta t}{\epsilon^2}}\right)} \le \frac{\left(u_{ij}^*\right)^2}{e^{\frac{-2\Delta t}{\epsilon^2}}} = e^{\frac{2\Delta t}{\epsilon^2}} (u_{ij}^*)^2. \tag{44}$$

Since $0 \le (u_{ij}^*)^2 \le 1$ and $0 < 1 - e^{\frac{-2\Delta t}{\epsilon^2}} < 1$, $(u_{ij}^*)^2\left(1 - e^{\frac{-2\Delta t}{\epsilon^2}}\right)$ is non-negative. Taking the discrete summation on Eq. (44) and taking the square root, we have

$$\|u^{n+1}\|_h \le e^{\frac{\Delta t}{\epsilon^2}} \|u^*\|_h. \tag{45}$$

Substituting u^{n+1} and u^* with $S_h^N U$ and U, respectively. We can prove the lemma 2. □

Lemma 3. *For any function with enough regularity, the following inequality holds*

$$\|I^h S^L u - S_h^L I^h u\|_h \le \tilde{C}(\Delta t^2 + \Delta t h^2), \tag{46}$$

where $\tilde{C}$ is a constant which is independent of Δt and h.

Proof. Let $u(t_{n+1}) = S^L u(t_n)$ and $U^{n+1} = S_h^L U^n$. We need to estimate the bound of the discrete L^2 norm of

$$\mathbb{E} = I^h u(t_{n+1}) - S_h^L(I^h u(t_n)) = I^h S^L u(t_n) - S_h^L(I^h u(t_n)). \tag{47}$$

Let $e^n = I^h u(t_n) - U^n$, we have the following equality from the scheme Eq. (22)

$$U^{n+1} = (I - \Delta t \Delta_h)^{-1} U^n, \tag{48}$$

where I is the identity matrix. Using the Taylor expansion, we have

$$I^h(u_{n+1}) = (I - \Delta t \Delta_h)^{-1}(I^h u(t_n) + \Delta t \tau^n), \tag{49}$$

where

$$\tau^n = \frac{I^h u(t_{n+1}) - I^h u(t_n)}{\Delta t} - \Delta_h I^h u(t_{n+1}).$$

Subtracting Eq. (48) from Eq. (49), we get

$$e^{n+1} = (I - \Delta t \Delta_h)^{-1}(e^n + \Delta t \tau^n). \tag{50}$$

Let $U^n = I^h u(t_n)$, then $\mathbb{E} = e^{n+1} = \Delta t(I - \Delta t \Delta_h)^{-1} \tau^n$. Since the discrete L^2 operate norm of $(I - \Delta t \Delta_h)^{-1}$ is less than or equal to 1, we have

$$\|\mathbb{E}\|_h \le \Delta t \|\tau^n\|_h. \tag{51}$$

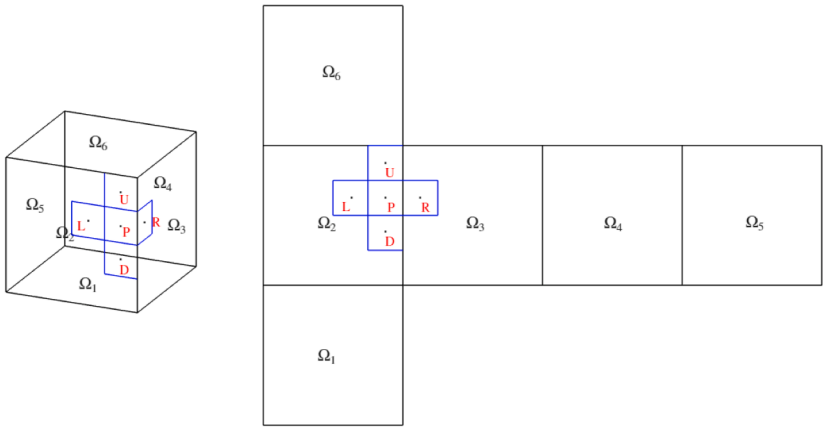

Fig. 5. Schematic illustration of a grid point P and its four adjacent grid points of type 2.

Using the Taylor expansion for $I^h u(t_{n+1})$ and using $\frac{\partial u}{\partial t} = \Delta u$ and $\frac{\partial^2 u}{\partial t^2} = \Delta^2 u$, we get

$$I^h u(t_{n+1}) = I^h u(t_n) + \Delta t I^h \Delta u(t_n) + \frac{\Delta t^2}{2} I^h \Delta^2 u(t_n) + O(\Delta t^3). \tag{52}$$

Using the Taylor expansion for $\Delta_h I^h u(t_{n+1})$, we get

$$\Delta_h I^h u(t_{n+1}) = \Delta_h I^h u(t_n) + \Delta t \Delta_h I^h \Delta u(t_n) + O(\Delta t^2). \tag{53}$$

Substituting Eqs. (52) and (53) into the definition of τ^n, we have

$$\tau^n = \underbrace{I^h \Delta u(t_n) - \Delta_h I^h u(t_n)}_{\mathrm{I}^n} + \frac{\Delta t}{2} I^h \Delta^2 u(t_n) - \Delta t \Delta_h I^h \Delta u(t_n) + O(\Delta t^2). \tag{54}$$

Here

$$\Delta_h I^h \Delta u(t_n) = I^h \Delta^2 u(t_n) - \underbrace{\left(I^h \Delta^2 u(t_n) - \Delta_h I^h \Delta u(t_n) \right)}_{\mathrm{II}^n}.$$

Thus, we get

$$\tau^n = \mathrm{I}^n + \Delta t \mathrm{II}^n - \frac{\Delta t}{2} I^h \Delta^2 u(t_n) + O(\Delta t^2). \tag{55}$$

Next, we need to establish the estimate for I^n. To simplify the expression, we define $u = u(t_n)$.

In the virtual cubic domain $\hat{\Omega}$, we split the grid point into three types. In type 1, the grid point (x_i, y_j) and its four adjacent grid points locate in a same subdomain $\hat{\Omega}_k$. In type 2, the grid points (x_i, y_j) and its three adjacent grid points locate in a same subdomain $\hat{\Omega}_{k1}$, one adjacent grid point locates in another subdomain $\hat{\Omega}_{k2}$. Fig. (5) illustrates an example of type 2. In type 3, the grid points (x_i, y_j) and its two adjacent grid points locate in a same subdomain $\hat{\Omega}_{k1}$, one adjacent grid point locate in a subdomain $\hat{\Omega}_{k2}$, another adjacent grid point locates in a subdomain $\hat{\Omega}_{k3}$. Fig. (6) illustrates an example of type 3. We define G_1, G_2, and G_3 as collections which include all grid points of type 1, of type 2, and of type 3, respectively.

For type 1, $\Delta u = u_{xx} + u_{yy}$, $u_x x$ and $u_y y$ are the second order derivatives of u with respect to x and y, respectively. Using the Taylor expansion for $u_{i+1,j}$ and $u_{i-1,j}$, we have

$$u_{i+1,j} = u_{ij} + h u_x(x_i, y_j) + \frac{h^2}{2} u_{xx}(x_i, y_j) + \frac{h^3}{6} u_{xxx}(x_i, y_j) + \frac{h^4}{24} u_{xxxx}(\xi^+, y_j), \tag{56}$$

$$u_{i-1,j} = u_{ij} - h u_x(x_i, y_j) + \frac{h^2}{2} u_{xx}(x_i, y_j) - \frac{h^3}{6} u_{xxx}(x_i, y_j) + \frac{h^4}{24} u_{xxxx}(\xi^-, y_j), \tag{57}$$

where $\xi^+ \in (x_i, x_{i+1})$ and $\xi^- \in (x_{i-1}, x_i)$. Adding the above two equations, we have

$$\frac{u_{i+1,j} + u_{i-1,j} - 2u_{ij}}{h^2} = u_{xx}(x_i, y_j) + \frac{h^2}{24} \left(u_{xxxx}(\xi^+, y_j) + u_{xxxx}(\xi^-, y_j) \right). \tag{58}$$

Using the intermediate value theorem, we can find $\tilde{\xi} \in (\xi^-, \xi^+)$ such that

$$\frac{u_{i+1,j} + u_{i-1,j} - 2u_{ij}}{h^2} = u_{xx}(x_i, y_j) + \frac{h^2}{12} u_{xxxx}(\tilde{\xi}, y_j). \tag{59}$$

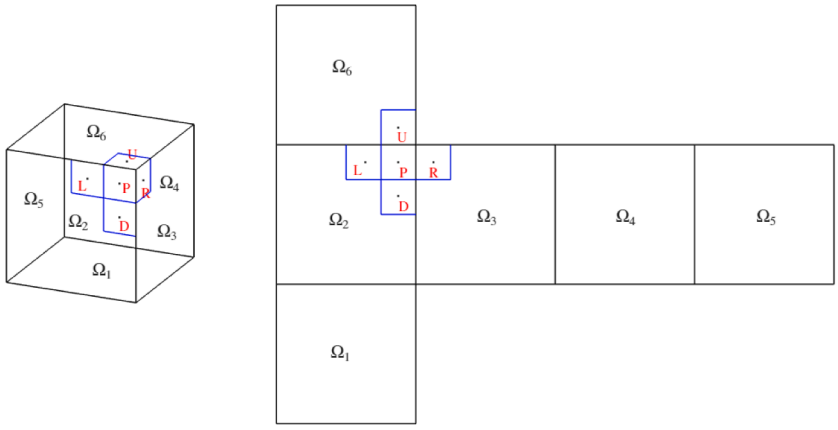

Fig. 6. Schematic illustration of a grid point P and its four adjacent grid points of type 3.

Similarly, we can get

$$\frac{u_{i,j+1}+u_{i,j-1}-2u_{ij}}{h^2} = u_{yy}(x_i,y_j)+\frac{h^2}{12}u_{yyyy}(x_i,\hat{\xi}). \tag{60}$$

Adding Eqs. (59) and (60), we get

$$(\Delta_h I^h u)_{ij} = I^h(\Delta u)(x_i,y_j)+\frac{h^2}{12}u_{xxxx}(\tilde{\xi},y_j)+\frac{h^2}{12}u_{yyyy}(x_i,\hat{\xi}). \tag{61}$$

Therefore, $\mathrm{I}^n_{ij} = -\frac{h^2}{12}\left(u_{xxxx}(\tilde{\xi},y_j)+\frac{h^2}{12}u_{yyyy}(x_i,\hat{\xi})\right)$. Thus, we derive

$$|\mathrm{I}^n_{ij}|_{(x_i,y_j)\in G_1} \le \frac{h^2}{12}\left(\|u_{xxxx}\|_\infty+\|u_{yyyy}\|_\infty\right). \tag{62}$$

For type 2, Fig. 5 shows that although the grid point P and its adjacent grid point R do not locate in a same subdomain, these grid points can still be regarded as locating in a bigger domain because Ω_2 and Ω_3 are connected. Then, we can use the Taylor expansion to implement similar analysis as we have established for type 1. When a grid point P and its one adjacent grid point cannot be regarded as locating in a same large domain, i.e., the subdomain $\hat{\Omega}_{k1}$ including P and the subdomain $\hat{\Omega}_{k2}$ including that adjacent grid point are not connected, the analysis based on Taylor expansion can still be established because the ghost point which is adjacent to P is defined by using the proposed boundary condition. For a specific case shown in Fig. 5, the Taylor expansion leads to

$$u(R)+u(L)-2u(P) = h^2u_{xx}(P)+\frac{h^4}{12}u_{xxxx}(\tilde{P}_1), \tag{63}$$

$$u(U)+u(D)-2u(P) = h^2u_{yy}(P)+\frac{h^4}{12}u_{yyyy}(\tilde{P}_2). \tag{64}$$

Combining the above two equations, we get

$$\Delta_h I^h u(P) = I^h\Delta u(P)+\frac{h^2}{12}(u_{xxxx}(\tilde{P}_1)+u_{yyyy}(\tilde{P}_2)). \tag{65}$$

The above equation leads to $\mathrm{I}^n_p = -\frac{h^2}{12}(u_{xxxx}(\tilde{P}_1)+u_{yyyy}(\tilde{P}_2))$. Thus, we derive

$$|\mathrm{I}^n_p| \le \frac{h^2}{12}(\|u_{xxxx}\|_\infty+\|u_{yyyy}\|_\infty). \tag{66}$$

More generally, for each grid point $(x_i,y_j)\in G_2$, we can derive

$$|\mathrm{I}^n_{ij}|_{(x_i,y_j)\in G_2} \le \frac{h^2}{12}(\|u_{xxxx}\|_\infty+\|u_{yyyy}\|_\infty). \tag{67}$$

For type 3, the above mentioned analysis can still be used. Therefore, we derive that all each grid point $(x_i,y_j)\in G_3$, the following inequality holds

$$|\mathrm{I}^n_{ij}|_{(x_i,y_j)\in G_3} \le \frac{h^2}{12}(\|u_{xxxx}\|_\infty+\|u_{yyyy}\|_\infty). \tag{68}$$

Next, we implement the discrete L^2 estimate for all grid points in $\hat{\Omega}=G_1\cup G_2\cup G_3$. The square of discrete L^2 norm of I^n can be defined as

$$\|\mathrm{I}^n\|_h^2 = h^2\sum_{(x_i,y_j)\in G_1}\left(\mathrm{I}^n_{ij}\right)^2+h^2\sum_{(x_i,y_j)\in G_2}\left(\mathrm{I}^n_{ij}\right)^2h^2\sum_{(x_i,y_j)\in G_3}\left(\mathrm{I}^n_{ij}\right)^2. \tag{69}$$

From the above mentioned results for type 1, type 2, and type 3, we can find a constant C_1 which is independent with h such that

$$|\mathrm{I}_{ij}^n| \le C_1 h^2 (|u_{xxxx}| + |u_{yyyy}|)_{ij}, \quad \text{for } (x_i, y_j) \in \hat{\Omega}. \tag{70}$$

The above inequality leads to

$$h^2 \sum_{(x_i,y_j)\in\hat{\Omega}} \left(\mathrm{I}_{ij}^n\right)^2 \le C_1^2 h^6 \sum_{(x_i,y_j)\in\hat{\Omega}} \left(|u_{xxxx}| + |u_{yyyy}|\right)_{ij}^2. \tag{71}$$

Since

$$C_1^2 h^6 \sum_{(x_i,y_j)\in\hat{\Omega}} \left(|u_{xxxx}| + |u_{yyyy}|\right)_{ij}^2 \le C_2 \int_{\Omega} \left(|u_{xxxx}| + |u_{yyyy}|\right)^2 d\mathbf{x}, \tag{72}$$

where C_2 is a constant which is independent with h, Ω is the continuous virtual cubic domain, and $\mathbf{x}$ is spatial coordinate. We further get

$$\begin{aligned} h^2 \sum_{(x_i,y_j)\in\hat{\Omega}} \left(\mathrm{I}_{ij}^n\right)^2 &\le C_1^2 C_2 h^4 \int_{\Omega} \left(u_{xxxx}^2 + u_{yyyy}^2 + 2u_{xxxx}u_{yyyy}\right) d\mathbf{x} \\ &\le 2C_1^2 C_2 h^4 \int_{\Omega} \left(u_{xxxx}^2 + u_{yyyy}^2\right) d\mathbf{x} \\ &= 2C_1^2 C_2 h^4 \left(\|u_{xxxx}\|_{L^2}^2 + \|u_{yyyy}\|_{L^2}^2\right). \end{aligned} \tag{73}$$

Here, the inequality $2ab \le a^2 + b^2$ is used. $\|\cdot\|_{L^2}$ denotes the continuous L^2 norm. With enough regularity of exact solution u, we can find a constant C_3 such that $\|u_{xxxx}\|_{L^2}^2 + \|u_{yyyy}\|_{L^2}^2 \le C_3$. Thus, we derive

$$\|\mathrm{I}^n\|_h \le h^2 C_4. \tag{74}$$

Here, C_4 is a constant. Similarly, the enough regularity of exact solution u also leads to

$$\|\mathrm{II}^n\|_h \le h^2 C_5. \tag{75}$$

Thus, we get

$$\|\tau^n\|_h \le \tilde{C}_1 h^2 + \tilde{C}_2 \Delta t h^2 + \tilde{C}_3 \Delta t + O(\Delta t^2). \tag{76}$$

Here, $\tilde{C}_1 = C_4$, $\tilde{C}_2 = C_5$, and $\tilde{C}_3$ is a constant which is related to the upper bound of $\Delta^2 u(t_n)$. Finally, we prove that

$$\begin{aligned} \|\mathbb{E}\|_h &\le \Delta t \|\tau^n\|_h \\ &\le \tilde{C}_1 \Delta t h^2 + \tilde{C}_2 \Delta t^2 h^2 + \tilde{C}_3 \Delta t^2 + O(\Delta t^3) \\ &= \tilde{C}\left(\Delta t^2 + \Delta t h^2\right). \end{aligned} \tag{77}$$

The proof of lemma 3 is completed. □

Theorem 4.2. *Suppose $\hat{u}^{n+1}$ with enough regularity is the exact solution of the AC equation Eq. (33), $\tilde{u}^{n+1}$ is the exact solution of Eq. (34), U^{n+1} is the numerical solution of scheme Eqs. (22) and (23). Then, there exists a positive constant C which is independent with Δt and h, such that*

$$\|I^h \hat{u}^{n+1} - U^{n+1}\|_h \le C(\Delta t + h^2). \tag{78}$$

Proof. For $n \ge 0$, we have

$$\|I^h \hat{u}^{n+1} - U^{n+1}\|_h \le \|I^h \hat{u}^{n+1} - I^h \tilde{u}^{n+1}\|_h + \|I^h \tilde{u}^{n+1} - U^{n+1}\|_h. \tag{79}$$

The first term on the left-hand side of the above inequality is bounded by

$$\|I^h \tilde{u}^{n+1} - I^h \hat{u}^{n+1}\|_h \le \bar{C}_1 \Delta t^2. \tag{80}$$

For the second term on the left-hand side, we have

$$\begin{aligned} \|I^h \tilde{u}^{n+1} - U^{n+1}\|_h &= \|I^h S^N S^L \tilde{u}^n - S_h^N S_h^L U^n\|_h \\ &\le \|I^h S^N S^L \tilde{u}^n - S_h^N I^h S^L \tilde{u}^n\|_h + \|S_h^N I^h S_L \tilde{u}^n - S_h^N S_h^L U^n\|_h \\ &\le \|I^h S^N S^L \tilde{u}^n - S_h^N I^h S^L \tilde{u}^n\|_h + e^{\frac{\Delta t}{\epsilon^2}} \|I^h S^L \tilde{u}^n - S_h^L U^n\|_h \\ &\le e^{\frac{\Delta t}{\epsilon^2}} \|I^h S^L \tilde{u}^n - S_h^L I^h \tilde{u}^n\|_h + e^{\frac{\Delta t}{\epsilon^2}} \|S_h^L I^h \tilde{u}^n - S_h^L U^n\|_h \\ &\le e^{\frac{\Delta t}{\epsilon^2}} \tilde{C}(\Delta t h^2 + \Delta t^2) + e^{\frac{\Delta t}{\epsilon^2}} \|I^h \tilde{u}^n - U^n\|_h \\ &\le e^{\frac{T}{\epsilon^2}} \left[\tilde{C}(\Delta t h^2 + \Delta t^2) + \|I^h \tilde{u}^n - U^n\|_h\right]. \end{aligned} \tag{81}$$

Table 1
The spatial error and its convergence rate for various ϵ.

		$h = 1/32$	$h = 1/64$	$h = 1/128$	$h = 1/256$
$\epsilon = 0.12$	error	6.41647×10^{-2}	1.91865×10^{-2}	4.92118×10^{-3}	1.25388×10^{-3}
	rate	1.74	1.96	1.97	
$\epsilon = 0.06$	error	6.41653×10^{-2}	1.91867×10^{-2}	4.92122×10^{-3}	1.25389×10^{-3}
	rate	1.74	1.96	1.97	
$\epsilon = 0.03$	error	6.41675×10^{-2}	1.91874×10^{-2}	4.92139×10^{-3}	1.25393×10^{-3}
	rate	1.74	1.96	1.97	
$\epsilon = 0.015$	error	6.41764×10^{-2}	1.91900×10^{-2}	4.92205×10^{-3}	1.25410×10^{-3}
	rate	1.74	1.96	1.97	

Here, T is the computational time after $(n+1)$th time iteration, the lemmas 1–3 are used. Combing the above results, we get

$$
\begin{aligned}
&\|I^h\tilde{u}^{n+1} - U^{n+1}\|_h \le \bar{C}_1\Delta t^2 + e^{\frac{T}{\epsilon^2}}\left[\tilde{C}(\Delta t h^2 + \Delta t^2) + \|I^h\tilde{u}^n - U^n\|_h\right] \\
&\le (n+1)\left[\bar{C}_1\Delta t^2 + e^{\frac{T}{\epsilon^2}}\tilde{C}(\Delta t h^2 + \Delta t^2)\right] + e^{\frac{2T}{\epsilon^2}}\left[\tilde{C}(\Delta t h^2 + \Delta t^2) + \|I^h\tilde{u}^{n-1} - U^{n-1}\|_h\right] \\
&\le (n+1)\left[\bar{C}_1\Delta t^2 + e^{\frac{T}{\epsilon^2}}\tilde{C}(\Delta t h^2 + \Delta t^2)\right] + e^{\frac{2T}{\epsilon^2}}\left[\tilde{C}(\Delta t h^2 + \Delta t^2)\right. \\
&\qquad\qquad \left. + e^{\frac{T}{\epsilon^2}}\left[\tilde{C}(\Delta t h^2 + \Delta t^2) + \|I^h\tilde{u}^{n-2} - U^{n-2}\|_h\right]\|_h\right] \\
&\cdots \\
&\le \frac{T}{\Delta t}\left[\bar{C}_1\Delta t^2 + e^{\frac{T}{\epsilon^2}}\tilde{C}(\Delta t h^2 + \Delta t^2)\right] + \frac{T'}{\Delta t}C'(\Delta t h^2 + \Delta t^2) + C''\|I^h\tilde{u}^0 - U^0\|_h \\
&= T\left[\bar{C}_1\Delta t + e^{\frac{T}{\epsilon^2}}\tilde{C}(h^2 + \Delta t)\right] + T'C'(\Delta t + h^2) \\
&= C(\Delta t + h^2).
\end{aligned} \tag{82}
$$

Here, C', C'', and C are constants which are independent with Δt and h. T' is the computational time after nth time iteration. At initial state, $I^h\tilde{u}^0 = U^0$ is used. The theorem is proved. □

Remark 1. In this work, we aim to develop an effective numerical method for solving the AC equation on virtual cubic domain. Based on the temporally first-order operator splitting strategy and standard five-point finite difference stencil, the fully discrete scheme is proposed. We analytically prove the discrete maximum principle-preserving property and establish the error estimate. In a recent work [29], authors have theoretically proved that the operator splitting scheme satisfied the unconditionally energy stability with respect to a modified discrete energy. However, the unconditionally energy stability of operator splitting method with respect to an original discrete energy is still an open problem. In our further work, the estimate of original energy stability will be further investigated.

5. Computational tests

This section presents several computational experiments that are conducted to assess the accuracy, efficiency, and stability of the proposed computational scheme for solving the AC equation on the virtual cubic surface.

5.1. Convergence test

First, we study the convergence of the proposed algorithm for the AC equation on the virtual cubic surface. We consider the following initial condition:

$$u(x, y, 0) = 0.5\cos(2\pi x)\cos(2\pi y)$$

on the virtual cubic surface domain Ω. For the spatial accuracy, we use the parameters $\Delta t = 10^{-8}$, and set the final time at $t = 5 \times 10^{-8}$. We conduct numerical experiments for multiple values of ϵ in order to investigate the dependence of both the spatial error and the observed convergence rate on ϵ. We evaluate the relative error in the discrete l_2-norm by

$$\|e^h\|_2 = \sqrt{\frac{1}{6N^2}\sum_{(i,j)\in I_g}\left(e^h_{ij}\right)^2}, \tag{83}$$

where u^{ref} denotes a reference solution computed on a sufficiently fine mesh size $h_{\text{ref}} = 1/1024$, and

$$e^h_{ij} = u^h_{ij} - \left(\mathcal{R}_h u^{\text{ref}}\right)_{ij}, \qquad (i,j) \in I_g. \tag{84}$$

Table 2
The temporal error and its convergence rate for various ϵ.

		$\Delta t = 1/2^{10}$	$\Delta t = 1/2^{11}$	$\Delta t = 1/2^{12}$	$\Delta t = 1/2^{13}$
$\epsilon = 0.12$	error	1.70229×10^{-2}	8.67476×10^{-3}	4.35386×10^{-3}	2.15502×10^{-3}
	rate	0.97	0.99	1.01	
$\epsilon = 0.06$	error	3.03248×10^{-2}	1.54636×10^{-2}	7.76426×10^{-3}	3.84398×10^{-3}
	rate	0.97	0.99	1.01	
$\epsilon = 0.03$	error	1.44236×10^{-1}	7.59658×10^{-2}	3.86003×10^{-2}	1.91936×10^{-2}
	rate	0.92	0.97	1.00	
$\epsilon = 0.015$	error	7.30919×10^{-1}	6.51450×10^{-1}	4.58679×10^{-1}	2.26132×10^{-1}
	rate	0.16	0.50	1.02	

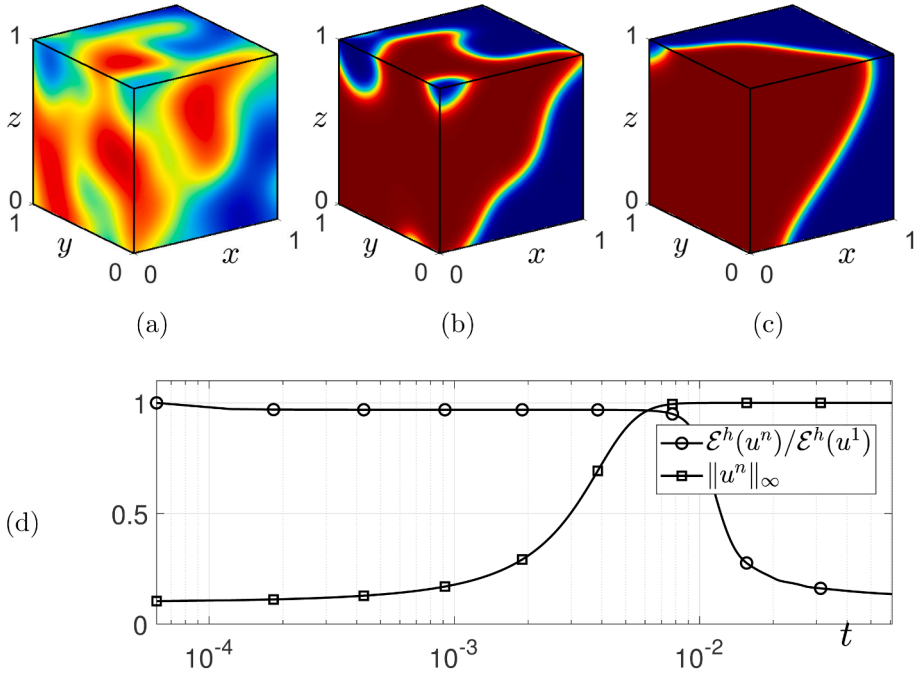

Fig. 7. (a)–(c) are snapshots of the temporal evolution of the numerical solution at times $t = 180, 300\Delta t$, and $1000\Delta t$, respectively. (d) is the normalized discrete total energy and the maximum norm for the AC equation over temporal evolution.

Here $\mathcal{R}_h$ denotes the restriction operator from the reference grid to the mesh size h. For cell-centered data, we use the block-averaging restriction; that is, if $r = h/h_{\text{ref}} \in \mathbb{N}$, then

$$\left(\mathcal{R}_h u^{\text{ref}}\right)_{ij} = \frac{1}{r^2} \sum_{a=1}^{r} \sum_{b=1}^{r} u^{\text{ref}}_{r(i-1)+a,\, r(j-1)+b}. \tag{85}$$

Table 1 lists spatial errors and their convergence rates for $h = 1/32, 1/64, 1/128$, and $1/256$.

For the temporal accuracy test, we fix the spatial resolution at $N = 128$ and evaluate the relative error in the discrete l_2-norm by

$$\|e^{\Delta t}\|_2 = \sqrt{\frac{1}{6N^2} \sum_{(i,j)\in I_g} \left(e^{\Delta t}_{ij}\right)^2}, \tag{86}$$

where u^{ref} denotes a reference solution computed with a sufficiently small time step $\Delta t_{\text{ref}} = 1/2^{18}$, and

$$e^{\Delta t}_{ij} = u^{\Delta t}_{ij} - u^{\text{ref}}_{ij}, \qquad (i,j) \in I_g. \tag{87}$$

Table 2 lists temporal errors and their convergence rates for $\Delta t = 1/2^{10}$, $\Delta t = 1/2^{11}$, $\Delta t = 1/2^{12}$, and $\Delta t = 1/2^{13}$. As a result, it is evident that the proposed scheme for solving the AC model on the virtual cubic surface exhibits first-order temporal accuracy and second-order spatial accuracy. Moreover, Tables 1 and 2 indicate larger errors for smaller values of ϵ. Table 2 also reports reduced temporal convergence rates when ϵ becomes small.

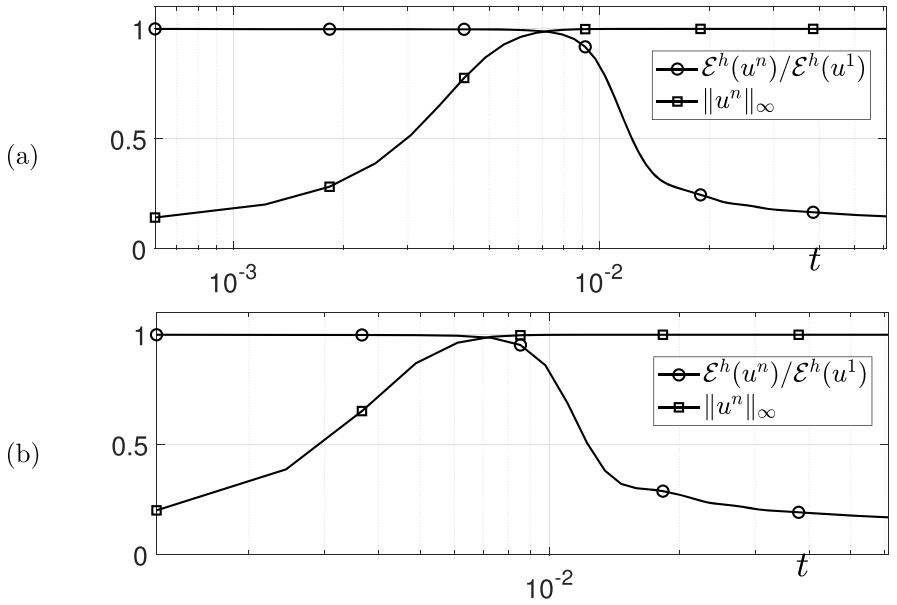

Fig. 8. (a) and (b) are the normalized discrete total energy and the maximum norm for the AC equation over temporal evolution for $\Delta t = 10h^2$ and $20h^2$, respectively.

5.2. Energy dissipation and maximum principle

We conduct computational tests to validate that the proposed computational algorithm satisfies the maximum principle and the energy dissipation properties for the AC equation. The discrete total energy is divided into two parts. The term $\mathcal{E}^h_{1,2,6}(u^n)$ corresponds to the domain $\hat{\Omega}_1 \cup \hat{\Omega}_2 \cup \hat{\Omega}_6$, and the term $\mathcal{E}^h_{3,4,5}(u^n)$ corresponds to the domain $\hat{\Omega}_3 \cup \hat{\Omega}_4 \cup \hat{\Omega}_5$, and they are defined as follows:

$$\begin{aligned}
\mathcal{E}^h_{1,2,6}(u^n) = \frac{1}{4}\Bigg[& \sum_{j=1}^{3N}\left(\left(u^n_{1,j}-u^n_{0,j}\right)^2 + 2\sum_{i=1}^{N-1}\left(u^n_{i+1,j}-u^n_{ij}\right)^2 + \left(u^n_{N+1,j}-u^n_{N,j}\right)^2\right) \\
& + \sum_{i=1}^{N}\left(\left(u^n_{i,1}-u^n_{i,0}\right)^2 + 2\sum_{j=1}^{3N-1}\left(u^n_{i,j+1}-u^n_{ij}\right)^2 + \left(u^n_{i,3N+1}-u^n_{i,3N}\right)^2\right) \\
& + \frac{h^2}{\epsilon^2}\sum_{i=1}^{N}\sum_{j=1}^{3N}\left(\left(u^n_{ij}\right)^2-1\right)^2\Bigg], \\
\mathcal{E}^h_{3,4,5}(u^n) = \frac{1}{4}\Bigg[& \sum_{j=N+1}^{2N}\left(\left(u^n_{N+1,j}-u^n_{N,j}\right)^2 + 2\sum_{i=N+1}^{4N-1}\left(u^n_{i+1,j}-u^n_{ij}\right)^2 + \left(u^n_{4N+1,j}-u^n_{4N,j}\right)^2\right) \\
& + \sum_{i=N+1}^{4N}\left(\left(u^n_{i,N+1}-u^n_{i,N}\right)^2 + 2\sum_{j=N+1}^{2N-1}\left(u^n_{i,j+1}-u^n_{ij}\right)^2 + \left(u^n_{i,2N+1}-u^n_{i,2N}\right)^2\right) \\
& + \frac{h^2}{\epsilon^2}\sum_{i=N+1}^{4N}\sum_{j=N+1}^{2N}\left(\left(u^n_{ij}\right)^2-1\right)^2\Bigg].
\end{aligned}$$

Here, to calculate the discrete total energy of the AC equation on each domain $\hat{\Omega}_k$ for $k = 1, 2, \dots, 6$, we use the boundary condition (15) for each domain. Thus, the discrete total energy $\mathcal{E}^h(u^n)$ at $t = n\Delta t$ on $\hat{\Omega}$ is defined as

$$\mathcal{E}^h(u^n) = \mathcal{E}^h_{1,2,6}(u^n) + \mathcal{E}^h_{3,4,5}(u^n).$$

Next, we define the maximum norm as

$$\|u^n\|_\infty = \max_{(i,j)\in I^g} |u^n_{ij}|.$$

For computational test, the initial condition on $\hat{\Omega}$ is given by

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

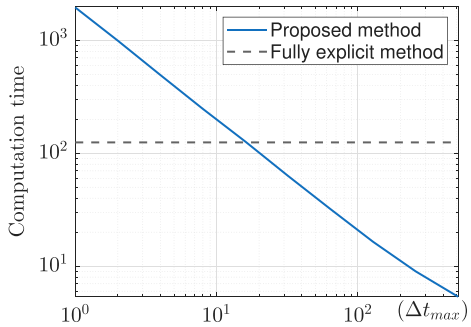

Fig. 9. Computational time and relative speedup of the proposed method as the time step size increases.

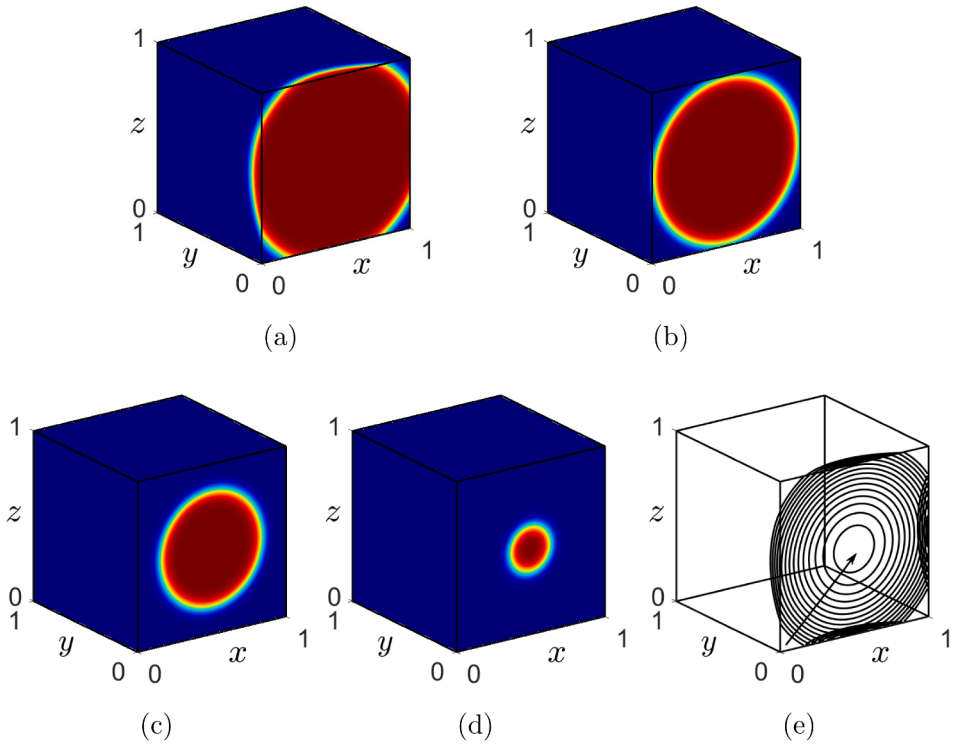

Fig. 10. (a)–(d) are snapshots of the temporal evolutions of the numerical solution at times $t = 0$, $10000\Delta t$, $20000\Delta t$, and $28000\Delta t$, respectively. (e) is the contour of the numerical solution at the $\phi = 0$ level.

where rand(x_i, y_j) is a random value between -1 and 1 at (x_i, y_j). The parameters used are $N = 128$, final time $T = 1000\Delta t$, $\nu_1 = \nu_2 = 2$, $tolerance = 1.0e - 7$, and $\epsilon = \epsilon_{22}$. Here, $\epsilon_m = mh/[2\sqrt{2}\tanh^{-1}(0.9)]$ denotes the value of ϵ chosen so that the width of the transition layer is resolved by approximately m grid points [35]. In previous work, we obtained that the time step size restriction of the fully explicit OSM for the AC equation is $\Delta t \leq 0.25h^2$ [19]. In present work, we use a relatively large time step size $\Delta t = h^2$, $10h^2$, and $20h^2$ relative to the maximum value $0.25h^2$ of the time step restriction of the fully explicit OSM for the AC equation. We define the normalized discrete total energy as $\mathcal{E}^h(u^n)/\mathcal{E}^h(u^1)$. Fig. 7 shows snapshots of the numerical solution at times $t = 20\Delta t$, $200\Delta t$, and $1000\Delta t$, along with the normalized discrete total energy and the maximum norm over the temporal evolution. Fig. 8 shows the normalized discrete total energy and the maximum norm as functions of time for $\Delta t = 10h^2$ and $20h^2$. Although the original energy stability is hard to prove, we observe that the numerical solution of the AC equation obtained with a large time step size numerically satisfies both the discrete energy dissipation property and the maximum principle.

Table 3
Computational time versus time step size for the proposed method, compared with the fully explicit method.

Δt	$\Delta t_{\max}$	$4\Delta t_{\max}$	$16\Delta t_{\max}$	$64\Delta t_{\max}$	$128\Delta t_{\max}$	$256\Delta t_{\max}$	$512\Delta t_{\max}$
time	1960.66	494.37	126.86	32.02	16.45	9.00	5.36
Ratio	0.06	0.25	0.98	3.90	7.59	13.87	23.31

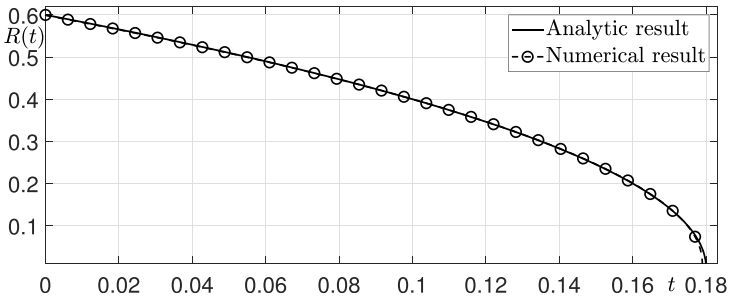

Fig. 11. Temporal evolution of the radius of the computational solution for the AC equation solved using the proposed algorithm.

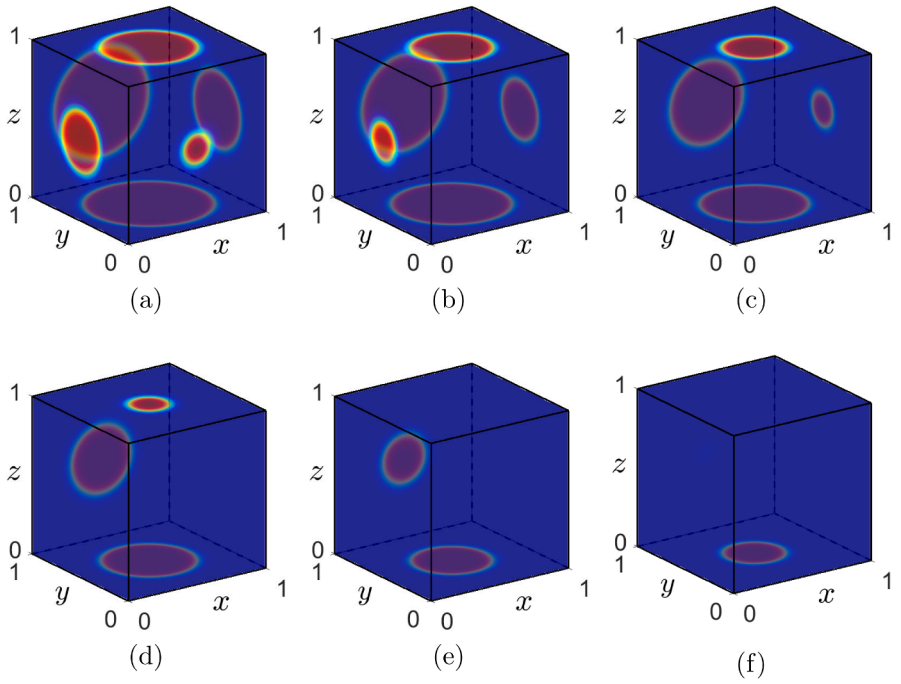

Fig. 12. (a)–(f) are snapshots of the temporal evolutions of the numerical solution at times $t = 0$, $2000\Delta t$, $4000\Delta t$, $6000\Delta t$, $8000\Delta t$, and $10000\Delta t$, respectively.

5.3. Computational efficiency test

We compare the proposed method with fully explicit schemes to verify computational efficiency. The parameters used are $N = 128$, final time $T = 10240\Delta t_{\max}$, $\nu_1 = \nu_2 = 2$, $tolerance = 1.0e-7$, and $\epsilon = \epsilon_{16}$. Here, $\Delta t_{\max}$ is defined as 0.99 times the maximum admissible time step given by the stability condition of the fully explicit method in [36]. For computational test, the initial condition on $\hat{\Omega}$ is given by

$$u(x_i, y_j, 0) = \text{rand}(x_i, y_j),$$

where $\text{rand}(x_i, y_j)$ is a random value between -1 and 1 at (x_i, y_j). Fig. 9 and Table 3 demonstrate the computational efficiency of the proposed method as the time step size increases. As Δt grows from $\Delta t_{\max}$ to $512\Delta t_{\max}$, the total computational time of the proposed method decreases significantly, while the fully explicit method remains constrained by the stability-limited time step. This behavior results in a substantial relative speedup, which increases monotonically with Δt, as reported in Table 3.

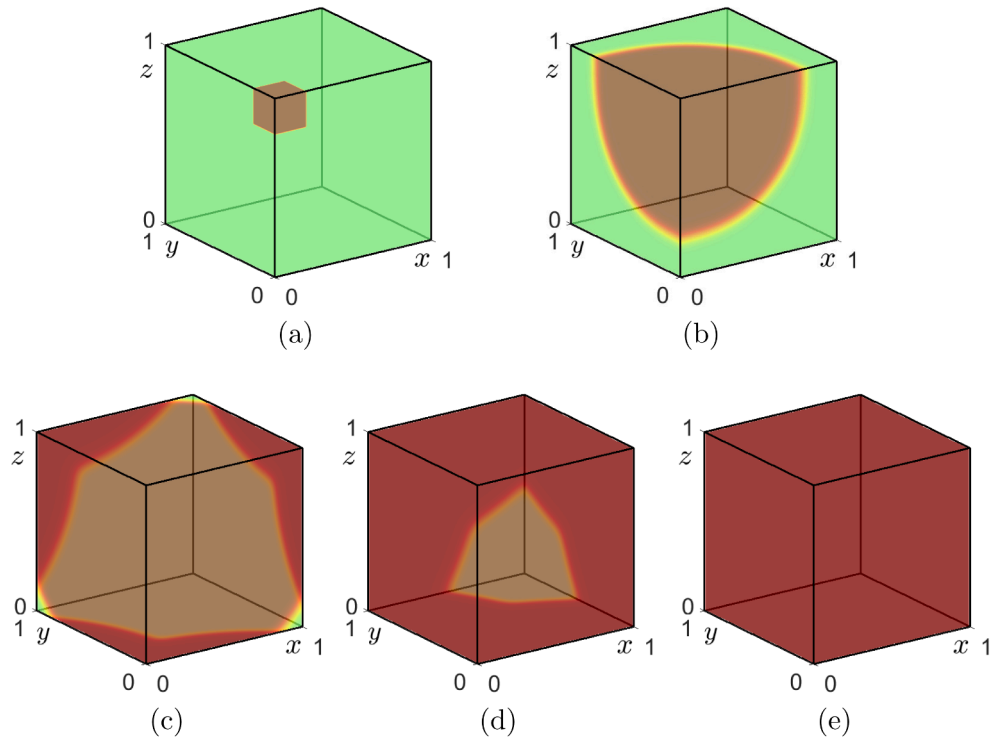

Fig. 13. Temporal evolution of the traveling wave solution ϕ on the virtual cubic surface at time (a) $t = 0$, (b) $t = 800\Delta t$, (c) $t = 1400\Delta t$, (d) $t = 2000\Delta t$, and $t = 2600\Delta t$.

5.4. Motion by mean curvature

Now, we conduct a computational test to observe whether the computational solution of the AC equation approximates the motion by the mean curvature. For computational test, the initial condition on $\hat{\Omega}$ is given by

$$u(x_i, y_j, 0) = \begin{cases} \tanh\left(\dfrac{0.6 - \sqrt{(x_i - 4.5)^2 + (y_j - 1.5)^2}}{\sqrt{2}\epsilon}\right), & (x_i, y_j) \in \hat{\Omega}_5, \\ \tanh\left(\dfrac{0.6 - \sqrt{(x_i - 0.5)^2 + (y_j - 1.5)^2}}{\sqrt{2}\epsilon}\right), & \text{otherwise.} \end{cases}$$

We use the parameters $N = 128$, $\Delta t = 0.1h^2$, final time $T = 30000\Delta t$, $\nu_1 = \nu_2 = 2$, $tolerance = 1.0e - 7$, and $\epsilon = \epsilon_{16}$. Figs. 10(a)–(d) display the temporal evolution of the computational solution of the AC model on the virtual cubic surface at $t = 0$, $10000\Delta t$, $20000\Delta t$, and $28000\Delta t$, respectively. In addition, Fig. 10(e) shows the contours of the numerical solutions at the $\phi = 0$ level. The temporal evolutions of the numerical and analytic radii are shown in Fig. 11. We observe that the numerical solution solved using the multigrid algorithm approximates the motion by mean curvature.

We conduct a computational test to examine whether the solution behavior remains consistent with motion by mean curvature under more complex initial configurations. For computational test, the initial condition on $\hat{\Omega}$ is given by

$$u(x_i, y_j, 0) = \tanh\left(\frac{r_k - \sqrt{(x_i - a_k)^2 + (y_j - b_k)^2}}{\sqrt{2}\epsilon}\right), \quad (x_i, y_j) \in \hat{\Omega}_k,$$

where the parameters are specified as $r_k = \{0.4, 0.1, 0.25, 0.35, 0.2, 0.3\}$, $a_k = \{0.5, 0.5, 1.5, 2.5, 3.5, 0.5\}$, $b_k = \{0.5, 1.5, 1.5, 1.5, 1.5, 2.5\}$ for $k = 1, 2, \ldots, 6$. The numerical parameters are the same as in the previous experiment except that the final time is set to final time $T = 10000\Delta t$. Figs. 12(a)–(f) show the temporal evolution of the numerical solution of the AC equation on the virtual cubic surface at times $t = 0$, $2000\Delta t$, $4000\Delta t$, $6000\Delta t$, $8000\Delta t$, and $10000\Delta t$, respectively. The numerical results indicate that the complex initial shapes gradually shrink over time, with each curved interface eventually disappearing.

5.5. Traveling wave solution

To study the propagation characteristics of a traveling wave solution, the following initial condition is used:

$$u(x_i, y_j, 0) = \begin{cases} 1, & \text{if } |x| < 0.2 \text{ and } |y-2| < 0.2 \\ & \quad \text{or } |x-4| < 0.2 \text{ and } |y-2| < 0.2, \\ 0, & \text{otherwise,} \end{cases}$$

as shown in Fig. 13(a). Fig. 13 shows the temporal evolution of the traveling wave solution ϕ starting from a localized initial condition (88). The simulation is performed with parameters $N = 128$, $\Delta t = 0.1h^2$, final time $T = 2600\Delta t$, $\nu_1 = \nu_2 = 2$, and $\epsilon = \epsilon_8$. The rectangular region is initially activated near the corner of the virtual cubic surface, while the rest of the domain is initialized with zero. As time progresses, the traveling wave propagates outward symmetrically.

6. Conclusions

We proposed the unconditionally stable hybrid numerical method for the nonlinear AC equation on the virtual cubic surface using the multigrid method. The fully explicit OSM for the AC equation has a strict time step size restriction. To overcome this restriction for the time step size, we proposed the unconditionally stable hybrid scheme that solves the diffusion term of the AC equation using the multigrid method and then solves the nonlinear term analytically. The proposed method overcomes the time step restriction by using the multigrid method that includes boundary conditions based on the properties of virtual cubic surfaces, solving the AC equation on virtual cubic surfaces. Numerical experiments were performed to conduct various tests, such as the convergence test, to demonstrate the accuracy and efficiency of the proposed method. The numerical results show that the proposed method overcomes the time step size restriction of the fully explicit OSM for the AC equation. The numerical solution is unconditionally stable even when using a relatively large time step size.

CRediT authorship contribution statement

Junxiang Yang: Writing – review & editing, Writing – original draft, Validation, Software, Methodology, Investigation, Formal analysis, Conceptualization; **Soobin Kwak:** Writing – review & editing, Writing – original draft, Visualization, Validation, Resources, Investigation; **Seokjun Ham:** Writing – original draft, Visualization, Validation, Software, Methodology, Investigation, Formal analysis; **Youngjin Hwang:** Writing – original draft, Visualization, Validation, Software, Methodology, Investigation, Formal analysis; **Hyundong Kim:** Writing – review & editing, Writing – original draft, Visualization, Validation, Software, Methodology, Formal analysis, Data curation, Conceptualization; **Juho Ma:** Writing – review & editing, Validation, Methodology, Investigation, Data curation; **Junseok Kim:** Writing – review & editing, Writing – original draft, Validation, Supervision, Software, Project administration, Methodology, Investigation, Funding acquisition, Formal analysis, Conceptualization.

Data availability

Data will be made available on request.

Use of AI tools declaration

This article was not created using Artificial Intelligence (AI) tools.

Conflicts of interest

None declared.

Acknowledgement

Junxiang Yang is supported by the Macau Science and Technology Development Fund (FDCT) (No. FDCT-24-080-SCSE, No. FDCT-25-076-SCSE) and Guangdong Basic and Applied Basic Research Foundation, General Program (No. 2026A1515012029). Hyundong Kim was supported by Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education (RS-2026-25469327, RS-2025-25415913). 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). The authors express their sincere gratitude to the reviewers for their valuable and constructive comments, which greatly improved the quality of the revised manuscript.

Appendix A.

The following Listing A.1 is a MATLAB code for the AC equation on a virtual cubic surface.

```
clear all;
N = 64; Nx = 4*N; Ny = 3*N; h = 1/N;
x = linspace(0.5*h,4-0.5*h,Nx); y = linspace(0.5*h,3-0.5*h,Ny);
m = 8; ep = m*h/(2*sqrt(2)*atanh(0.9));
dt = 0.1*h^2; Nt = 7000; Ns = round(Nt/3);
nu_1 = 2; nu_2 = nu_1; tol = 1.0e-7;

% for Multigrid
TL= log2(N); MGu=cell(1,TL); MGf=cell(1,TL);
for g = 1:TL
MGu{g} = zeros(4*2^g,3*2^g); MGf{g} = zeros(4*2^g,3*2^g);
end

% Initial condition
[X,Y] = ndgrid(x,y); r = 0.6;
R1 = sqrt((X-0.5).^2+(Y-1.5).^2); R2 = sqrt((X-4.5).^2 + (Y-1.5).^2);
u = MGu{TL};
u(1:3*N,:) = tanh((r-R1(1:3*N,:))/(sqrt(2)*ep));
u(3*N+1:Nx,N+1:2*N) = tanh((r-R2(3*N+1:Nx,N+1:2*N))/(sqrt(2)*ep));
u(N+1:Nx,[1:N,2*N+1:Ny]) = NaN;
MGu{TL} = u;

x1 = linspace(0.5*h,1-0.5*h,N); y1 = linspace(0.5*h,1-0.5*h,N);
[xx, yy] = meshgrid(x1,y1); cN = linspace(-1,1,501);
figure(1); clf; hold on; colormap jet; clim([-1,1]);
set(gca,'Box','on','BoxStyle','full','LineWidth',1.5,'View',[-35,20])
% Omega_1
[~,hh] = contourf(xx,yy,MGu{TL}(1:N,N:-1:1)',cN,'LineStyle','none');
hh.ZLocation = 0.5*h;
% Omega_2
t = hgtransform;
[~,hh] = contourf(xx,yy,MGu{TL}(1:N,N+1:2*N)',cN,'Parent',t,'LineStyle','none');
t.Matrix = makehgtform('xrotate',1/2*pi); hh.ZLocation = -0.5*h;
% Omega_3
t = hgtransform;
[~,hh] = contourf(xx,yy,MGu{TL}(N+1:2*N,N+1:2*N),cN,'Parent',t,'LineStyle','none');
t.Matrix = makehgtform('yrotate',-1/2*pi); hh.ZLocation = -1+0.5*h;
% Omega_4
t = hgtransform;
[~,hh] = contourf(xx,yy,MGu{TL}(3*N:-1:2*N+1,N+1:2*N)',cN,'Parent',t, ...
'LineStyle','none'); t.Matrix = makehgtform('xrotate',1/2*pi);
hh.ZLocation = -1+0.5*h;
% Omega_5
t = hgtransform;
[~,hh] = contourf(xx,yy,MGu{TL}(Nx:-1:3*N+1,N+1:2*N),cN,'Parent',t, ...
'LineStyle','none'); t.Matrix = makehgtform('yrotate',-1/2*pi);
hh.ZLocation = -0.5*h;
% Omega_6
[~,hh] = contourf(xx,yy,MGu{TL}(1:N,2*N+1:Ny)',cN,'LineStyle','none');
hh.ZLocation = 1-0.5*h;
axis image; axis([0 1 0 1 0 1])
tcks=[0,0.5,1]; xticks(tcks); yticks(tcks); zticks(tcks)
text(0.8,-0.07,-0.07,'$x$','interpreter','latex')
text(-0.07,0.8,-0.07,'$y$','interpreter','latex')
text(-0.07,1.07,0.8,'$z$','interpreter','latex')

%%% Main loop
for it = 1:Nt
MGf{TL} = MGu{TL}/dt; err = 2*tol;
while err > tol
oldu = MGu{TL}; MGu{TL} = relax(nu_1,MGu{TL},MGf{TL},dt);
d = defect(MGu{TL},MGf{TL},dt);
MGf{TL-1}=0.25*(d(2:2:4*2^(TL),2:2:3*2^(TL))+d(1:2:4*2^(TL)-1,2:2:3*2^(TL)) ...
+d(2:2:4*2^(TL),1:2:3*2^(TL)-1)+d(1:2:4*2^(TL)-1,1:2:3*2^(TL)-1));
for g = TL-1:-1:2
```

Listing A.1. MATLAB program.

```
MGu{g} = relax(nu_1,zeros(4*2^g,3*2^g),MGf{g},dt);
d = defect(MGu{g},MGf{g},dt);

MGf{g-1} = 0.25*(d(2:2:4*2^(g),2:2:3*2^(g))+d(1:2:4*2^(g)-1,2:2:3*2^(g)) ...
+d(2:2:4*2^(g),1:2:3*2^(g)-1)+d(1:2:4*2^(g)-1,1:2:3*2^(g)-1));
end
MGu{1} = relax(nu_1,zeros(4*2,3*2),MGf{1},dt);
for g = 2:TL
for i = 1:4*2^(g-1)
for j = 1:3*2^(g-1)
MGu{g}(2*i-1:2*i,2*j-1:2*j) = MGu{g}(2*i-1:2*i,2*j-1:2*j)+MGu{g-1}(i,j);
end
end
MGu{g} = relax(nu_2,MGu{g},MGf{g},dt);
end
err = sqrt(sum((MGu{TL}(1:N,1:Ny)-oldu(1:N,1:Ny)).^2,'all') ...
+sum((MGu{TL}(1:Nx,N+1:2*N)-oldu(1:Nx,N+1:2*N)).^2,'all')/(6*N^2));
end
MGu{TL} = MGu{TL}./sqrt(exp(-2*dt/ep^2)+MGu{TL}.^2*(1-exp(-2*dt/ep^2)));

if mod(it,Ns) == 0
figure(1); clf; hold on; colormap jet; clim([-1,1]);
set(gca,'Box','on','BoxStyle','full','LineWidth',1.5,'View',[-35,20])
% Omega_1
[~,hh] = contourf(xx,yy,MGu{TL}(1:N,N:-1:1)',cN,'LineStyle','none');
hh.ZLocation = 0.5*h;
% Omega_2
t = hgtransform;
[~,hh] = contourf(xx,yy,MGu{TL}(1:N,N+1:2*N)',cN,'Parent',t,'LineStyle','none');
hh.ZLocation = -0.5*h; t.Matrix = makehgtform('xrotate',1/2*pi);
% Omega_3
t = hgtransform;
[~,hh] = contourf(xx,yy,MGu{TL}(N+1:2*N,N+1:2*N),cN,'Parent',t, ...
'LineStyle','none'); t.Matrix = makehgtform('yrotate',-1/2*pi);
hh.ZLocation = -1+0.5*h;
% Omega_4
t = hgtransform;
[~,hh] = contourf(xx,yy,MGu{TL}(3*N:-1:2*N+1,N+1:2*N)',cN,'Parent',t, ...
'LineStyle','none'); t.Matrix = makehgtform('xrotate',1/2*pi);
hh.ZLocation = -1+0.5*h;
% Omega_5
t = hgtransform;
[~,hh] = contourf(xx,yy,MGu{TL}(Nx:-1:3*N+1,N+1:2*N),cN,'Parent',t, ...
'LineStyle','none'); t.Matrix = makehgtform('yrotate',-1/2*pi);
hh.ZLocation = -0.5*h;
% Omega_6
[~,hh] = contourf(xx,yy,MGu{TL}(1:N,2*N+1:Ny)',cN,'LineStyle','none');
hh.ZLocation = 1-0.5*h;
axis image; axis([0 1 0 1 0 1])
text(0.8,-0.07,-0.07,'$x$','interpreter','latex')
text(-0.07,0.8,-0.07,'$y$','interpreter','latex')
text(-0.07,1.07,0.8,'$z$','interpreter','latex')
end
end


function u = relax(nu_1,u,f,dt)
[Nx,Ny] = size(u); N = Nx/4; h=1/N; coef = 1/dt+4/h^2;
for it = 1:nu_1
% j = 1
u(1,1) = (f(1,1)+(u(2,1)+u(3*N+1,N+1)+u(1,2)+u(3*N,N+1))/h^2)/coef;
```

```
for i = 2:N-1
u(i,1) = (f(i,1)+(u(i+1,1)+u(i-1,1)+u(i,2)+u(3*N+1-i,N+1))/h^2)/coef;
end
u(N,1) = (f(N,1)+(u(2*N,N+1)+u(N-1,1)+u(N,2)+u(2*N+1,N+1))/h^2)/coef;
% j = 2,...,N
for j = 2:N
u(1,j) = (f(1,j)+(u(2,j)+u(3*N+j,N+1)+u(1,j+1)+u(1,j-1))/h^2)/coef;
for i = 2:N-1
u(i,j) = (f(i,j)+(u(i+1,j)+u(i-1,j)+u(i,j+1)+u(i,j-1))/h^2)/coef;
end
u(N,j) = (f(N,j)+(u(2*N+1-j,N+1)+u(N-1,j)+u(N,j+1)+u(N,j-1))/h^2)/coef;
end
% j = N+1
u(1,N+1) = (f(1,N+1)+(u(2,N+1)+u(Nx,N+1)+u(1,N+2)+u(1,N))/h^2)/coef;
for i = 2:N
u(i,N+1) = (f(i,N+1)+(u(i+1,N+1)+u(i-1,N+1)+u(i,N+2)+u(i,N))/h^2)/coef;
end
for i = N+1:2*N
u(i,N+1) = (f(i,N+1)+(u(i+1,N+1)+u(i-1,N+1)+u(i,N+2)+u(N,N+N+1-i))/h^2)/coef;
end
for i = 2*N+1:3*N
u(i,N+1) = (f(i,N+1)+(u(i+1,N+1)+u(i-1,N+1)+u(i,N+2)+u(N+2*N+1-i,1))/h^2)/coef;
end
for i = 3*N+1:Nx-1
u(i,N+1) = (f(i,N+1)+(u(i+1,N+1)+u(i-1,N+1)+u(i,N+2)+u(1,i-3*N))/h^2)/coef;
end
u(Nx,N+1) = (f(Nx,N+1)+(u(1,N+1)+u(Nx-1,N+1)+u(Nx,N+2)+u(1,N))/h^2)/coef;
% j = N+2,...,2N-1
for j = N+2:2*N-1
u(1,j) = (f(1,j)+(u(2,j)+u(Nx,j)+u(1,j+1)+u(1,j-1))/h^2)/coef;
for i = 2:Nx-1
u(i,j) = (f(i,j)+(u(i+1,j)+u(i-1,j)+u(i,j+1)+u(i,j-1))/h^2)/coef;
end
u(Nx,j) = (f(Nx,j)+(u(1,j)+u(Nx-1,j)+u(Nx,j+1)+u(Nx,j-1))/h^2)/coef;
end
% j = 2N
u(1,2*N) = (f(1,2*N)+(u(2,2*N)+u(Nx,2*N)+u(1,2*N+1)+u(1,2*N-1))/h^2)/coef;
for i = 2:N
u(i,2*N) = (f(i,2*N)+(u(i+1,2*N)+u(i-1,2*N)+u(i,2*N+1)+u(i,2*N-1))/h^2)/coef;
end
for i = N+1:2*N
u(i,2*N) = (f(i,2*N)+(u(i+1,2*N)+u(i-1,2*N)+u(N,N+i)+u(i,2*N-1))/h^2)/coef;
end
for i = 2*N+1:3*N
u(i,2*N) = (f(i,2*N)+(u(i+1,2*N)+u(i-1,2*N)+u(3*N+1-i,Ny)+u(i,2*N-1))/h^2)/coef;
end
for i = 3*N+1:Nx-1
u(i,2*N) = (f(i,2*N)+(u(i+1,2*N)+u(i-1,2*N)+u(1,6*N+1-i)+u(i,2*N-1))/h^2)/coef;
end
u(Nx,2*N) = (f(Nx,2*N)+(u(1,2*N)+u(Nx-1,2*N)+u(1,2*N+1)+u(Nx,2*N-1))/h^2)/coef;
% j = 2N+1,...,Ny-1
for j = 2*N+1:Ny-1
u(1,j) = (f(1,j)+(u(2,j)+u(6*N+1-j,2*N)+u(1,j+1)+u(1,j-1))/h^2)/coef;
for i = 2:N-1
u(i,j) = (f(i,j)+(u(i+1,j)+u(i-1,j)+u(i,j+1)+u(i,j-1))/h^2)/coef;
end
u(N,j) = (f(N,j)+(u(j-N,2*N)+u(N-1,j)+u(N,j+1)+u(N,j-1))/h^2)/coef;
end
% j = Ny
u(1,Ny) = (f(1,Ny)+(u(2,Ny)+u(3*N+1,2*N)+u(3*N,2*N)+u(1,Ny-1))/h^2)/coef;
for i = 2:N-1
```

```
u(i,Ny) = (f(i,Ny)+(u(i+1,Ny)+u(i-1,Ny)+u(3*N+1-i,2*N)+u(i,Ny-1))/h^2)/coef;
end
u(N,Ny) = (f(N,Ny)+(u(2*N,2*N)+u(N-1,Ny)+u(2*N+1,2*N)+u(N,Ny-1))/h^2)/coef;
end
end

function d = defect(u,f,dt)
[Nx,Ny] = size(u); d = zeros(Nx,Ny); N = Nx/4; h=1/N;
% j = 1
d(1,1) = f(1,1)-u(1,1)/dt+(-4*u(1,1)+u(2,1)+u(3*N+1,N+1)+u(1,2)+u(3*N,N+1))/h^2;
for i = 2:N-1
d(i,1) = f(i,1)-u(i,1)/dt+(-4*u(i,1)+u(i+1,1)+u(i-1,1)+u(i,2)+u(3*N+1-i,N+1))/h^2;
end
d(N,1) = f(N,1)-u(N,1)/dt+(-4*u(N,1)+u(2*N,N+1)+u(N-1,1)+u(N,2)+u(2*N+1,N+1))/h^2;
% j = 2,...,N
for j = 2:N
d(1,j) = f(1,j)-u(1,j)/dt+(-4*u(1,j)+u(2,j)+u(3*N+j,N+1)+u(1,j+1)+u(1,j-1))/h^2;
for i = 2:N-1
d(i,j) = f(i,j)-u(i,j)/dt+(-4*u(i,j)+u(i+1,j)+u(i-1,j)+u(i,j+1)+u(i,j-1))/h^2;
end
d(N,j)=f(N,j)-u(N,j)/dt+(-4*u(N,j)+u(2*N+1-j,N+1)+u(N-1,j)+u(N,j+1)+u(N,j-1))/h^2;
end
% j = N+1
d(1,N+1) = f(1,N+1)-u(1,N+1)/dt+(-4*u(1,N+1)+u(2,N+1)+u(Nx,N+1)+u(1,N+2)+u(1,N))/h
    ^2;
for i = 2:N
d(i,N+1) = f(i,N+1)-u(i,N+1)/dt+...
(-4*u(i,N+1)+u(i+1,N+1)+u(i-1,N+1)+u(i,N+2)+u(i,N))/h^2;
end
for i = N+1:2*N
d(i,N+1) = f(i,N+1)-u(i,N+1)/dt+...
(-4*u(i,N+1)+u(i+1,N+1)+u(i-1,N+1)+u(i,N+2)+u(N,N+N+1-i))/h^2;
end
for i = 2*N+1:3*N
d(i,N+1) = f(i,N+1)-u(i,N+1)/dt+...
(-4*u(i,N+1)+u(i+1,N+1)+u(i-1,N+1)+u(i,N+2)+u(N+2*N+1-i,1))/h^2;
end
for i = 3*N+1:Nx-1
d(i,N+1) = f(i,N+1)-u(i,N+1)/dt+...
(-4*u(i,N+1)+u(i+1,N+1)+u(i-1,N+1)+u(i,N+2)+u(1,i-3*N))/h^2;
end
d(Nx,N+1) = f(Nx,N+1)-u(Nx,N+1)/dt+...
(-4*u(Nx,N+1)+u(1,N+1)+u(Nx-1,N+1)+u(Nx,N+2)+u(1,N))/h^2;
% j = N+2,...,2N-1
for j = N+2:2*N-1
d(1,j) = f(1,j)-u(1,j)/dt+(-4*u(1,j)+u(2,j)+u(Nx,j)+u(1,j+1)+u(1,j-1))/h^2;
for i = 2:Nx-1
d(i,j) = f(i,j)-u(i,j)/dt+(-4*u(i,j)+u(i+1,j)+u(i-1,j)+u(i,j+1)+u(i,j-1))/h^2;
end
d(Nx,j) = f(Nx,j)-u(Nx,j)/dt+(-4*u(Nx,j)+u(1,j)+u(Nx-1,j)+u(Nx,j+1)+u(Nx,j-1))/h^2;
end
% j = 2N
d(1,2*N) = f(1,2*N)-u(1,2*N)/dt+...
(-4*u(1,2*N)+u(2,2*N)+u(Nx,2*N)+u(1,2*N+1)+u(1,2*N-1))/h^2;
for i = 2:N
d(i,2*N) = f(i,2*N)-u(i,2*N)/dt+...
(-4*u(i,2*N)+u(i+1,2*N)+u(i-1,2*N)+u(i,2*N+1)+u(i,2*N-1))/h^2;
end
for i = N+1:2*N
d(i,2*N) = f(i,2*N)-u(i,2*N)/dt+...
(-4*u(i,2*N)+u(i+1,2*N)+u(i-1,2*N)+u(N,N+i)+u(i,2*N-1))/h^2;
```

```
end
for i = 2*N+1:3*N
d(i,2*N) = f(i,2*N)-u(i,2*N)/dt+...
(-4*u(i,2*N)+u(i+1,2*N)+u(i-1,2*N)+u(N+2*N+1-i,Ny)+u(i,2*N-1))/h^2;
end
for i = 3*N+1:Nx-1
d(i,2*N) = f(i,2*N)-u(i,2*N)/dt+...
(-4*u(i,2*N)+u(i+1,2*N)+u(i-1,2*N)+u(1,3*N+3*N+1-i)+u(i,2*N-1))/h^2;
end
d(Nx,2*N) = f(Nx,2*N)-u(Nx,2*N)/dt+...
(-4*u(Nx,2*N)+u(1,2*N)+u(Nx-1,2*N)+u(1,2*N+1)+u(Nx,2*N-1))/h^2;
% j = 2N+1,...,Ny-1
for j = 2*N+1:Ny-1
d(1,j) = f(1,j)-u(1,j)/dt+...
(-4*u(1,j)+u(2,j)+u(4*N+2*N+1-j,2*N)+u(1,j+1)+u(1,j-1))/h^2;
for i = 2:N-1
d(i,j) = f(i,j)-u(i,j)/dt+...
(-4*u(i,j)+u(i+1,j)+u(i-1,j)+u(i,j+1)+u(i,j-1))/h^2;
end
d(N,j) = f(N,j)-u(N,j)/dt+...
(-4*u(N,j)+u(j-N,2*N)+u(N-1,j)+u(N,j+1)+u(N,j-1))/h^2;
end
% j = Ny
d(1,Ny) = f(1,Ny)-u(1,Ny)/dt+...
(-4*u(1,Ny)+u(2,Ny)+u(3*N+1,2*N)+u(3*N,2*N)+u(1,Ny-1))/h^2;
for i = 2:N-1
d(i,Ny) = f(i,Ny)-u(i,Ny)/dt+...
(-4*u(i,Ny)+u(i+1,Ny)+u(i-1,Ny)+u(3*N+1-i,2*N)+u(i,Ny-1))/h^2;
end
d(N,Ny) = f(N,Ny)-u(N,Ny)/dt+...
(-4*u(N,Ny)+u(2*N,2*N)+u(N-1,Ny)+u(2*N+1,2*N)+u(N,Ny-1))/h^2;
end
```

References

[1] Z. Cao, Z. Weng, S. Zhai, An effective operator splitting scheme for general motion by mean curvature using a modified Allen–Cahn equation, Appl. Math. Lett. 163 (2025) 109472.
[2] Z. Weng, S. Zhai, J. Kim, A novel shape transformation method via the nonlocal modified Allen-Cahn model: analysis and numerical simulations, Comput. Methods Appl. Mech. Eng. 450 (2026) 118639.
[3] S. Su, R. Gao, J. Yang, Second-order accurate, maximum principle-preserving, and convergent schemes for the phase-field shape transformation model, Eng. Comput. 41 (2025) 4665–4697.
[4] Z. Weng, S. Zhai, Numerical approximation of the multi-component conservative Allen–Cahn model based on boundary corrected strang splitting method, Comput. Methods Appl. Mech. Eng. 456 (2026) 118968.
[5] Z. Weng, S. Zhai, X. Feng, Numerical approximation of the nonlocal ternary viscous Cahn–Hilliard model based on a high order explicit hybrid algorithm, SIAM J. Sci. Comput. 47 (2025) A2221–A2247.
[6] B. Fu, D. Cai, X. Kong, R. Gao, J. Yang, On the numerical approximation of a phase-field volume reconstruction model: linear and energy-stable leap-frog finite difference scheme, Commun. Nonlinear Sci. Numer. Simul. 151 (2025) 109104.
[7] J. Yang, Y. Li, J. Kim, Energy-stable, L^2-stable, and weakly H^1-stable BDF2 viscosity splitting method for the penalized incompressible Navier–Stokes equations, J. Comput. Phys. 559 (2026) 114887.
[8] F. Zhang, Y. Xu, F. Chen, R. Guo, Interior penalty discontinuous Galerkin based isogeometric analysis for Allen–Cahn equations on surfaces, Commun. Comput. Phys. 18 (5) (2015) 1380–1416.
[9] R. Kusdiantara, F.T. Akbar, N. Nuraini, B.E. Gunara, H. Susanto, Snakes on lieb lattice, J. Nonlinear Sci. 32 (4) (2022) 59.
[10] X. Xiao, R. He, X. Feng, Unconditionally maximum principle preserving finite element schemes for the surface Allen–Cahn type equations, Numer. Methods Partial Differ. Equ. 36 (2) (2020) 418–438.
[11] Z. Sun, S. Zhang, A radial basis function approximation method for conservative Allen–Cahn equations on surfaces, Appl. Math. Lett. 143 (2023) 108634.
[12] H. Zamani-Gharaghoshi, M. Dehghan, M. Abbaszadeh, Numerical solution of Allen–Cahn model on surfaces via an effective method based on generalized moving least squares (GMLS) approximation and the closest point approach, Eng. Anal. Bound. Elem. 152 (2023) 575–581.
[13] Q. Pan, C. Chen, Y.J. Zhang, X. Yang, A novel hybrid IGA-EIEQ numerical method for the Allen–Cahn/Cahn–Hilliard equations on complex curved surfaces, Comput. Methods Appl. Mech. Eng. 404 (2023) 115767.
[14] B. Sjögreen, High order finite difference and finite volume methods for advection on the sphere, J. Sci. Comput. 51 (3) (2012) 703–732.
[15] S. Chen, C. Li, ExoCubed: a Riemann-solver-based cubed-sphere dynamic core for planetary atmospheres, Astrophys. J. 966 (1) (2024) 123.
[16] X. Chen, The LMARS based shallow-water dynamical core on generic gnomonic cubed-sphere geometry, J. Adv. Model. Earth Syst. 13 (1) (2021) e2020MS002280.
[17] J. Zhang, L.L. Wang, Z. Rong, A prolate-element method for nonlinear PDEs on the sphere, J. Sci. Comput. 47 (1) (2011) 73–92.
[18] D. Lee, H. Kim, The time-fractional Allen–Cahn equation on geometric computational domains, Commun. Nonlinear Sci. Numer. Simul. 141 (2025) 108455.
[19] Y. Hwang, J. Yang, G. Lee, S. Ham, S. Kang, S. Kwak, J. Kim, Fast and efficient numerical method for solving the Allen–Cahn equation on the cubic surface, Math. Comput. Simul. 215 (2024) 338–356.
[20] L. Ge, Y. Zhao, S. Zhong, Z. Shan, F. Ma, Z. Han, K. Guo, Efficient and integration stable nonlinear model predictive controller for autonomous vehicles based on the stabilized explicit integration method, Nonlinear Dyn. 111 (5) (2023) 4325–4342.
[21] Y. Li, H.G. Lee, D. Jeong, J. Kim, An unconditionally stable hybrid numerical method for solving the Allen–Cahn equation, Comput. Math. Appl. 60 (6) (2010) 1591–1606.
[22] Q. Zhang, J. Liu, X. Wang, F. Xie, S. Wu, Study on fault mechanism and dynamics of spur gear pair with time-varying backlash, Nonlinear Dyn. 112 (17) (2024) 14853–14877.
[23] J. Teunissen, F. Schiavello, Geometric multigrid method for solving Poisson's equation on octree grids with irregular boundaries, Comput. Phys. Commun. 286 (2023) 108665.
[24] X. Li, Y. Ge, Sixth-order compact difference scheme and multigrid method for solving the Poisson equation, Math. Sci. 18 (4) (2024) 655–668.

[25] K. Pan, H.W. Sun, Y. Xu, Y. Xu, An efficient multigrid solver for two-dimensional spatial fractional diffusion equations with variable coefficients, Appl. Math. Comput. 402 (2021) 126091.
[26] H.L. Liao, T. Tang, T. Zhou, On energy stable, maximum-principle preserving, second-order BDF scheme with variable steps for the Allen–Cahn equation, SIAM J. Numer. Anal. 58 (4) (2020) 2294–2314.
[27] J. Shen, T. Tang, J. Yang, On the maximum principle preserving schemes for the generalized Allen–Cahn equation, Commun. Math. Sci. 14 (6) (2016) 1517–1534.
[28] L.C. Evans, Partial Differential Equations, 19 of *Grad. Stud. Math.*, Amer. Math. Soc., 2nd ed., 2010.
[29] D. Li, C. Quan, J. Xu, Stability and convergence of strang splitting. Part I: scalar Allen–Cahn equation, J. Comput. Phys. 458 (2022) 111087.
[30] C. Yoo, M. Ji, J. Shin, Second-order operator splitting method for solving heterogeneous diffusion equations, J. Korean Soc. Ind. Appl. Math. 29 (3) (2025) 229–245.
[31] K.W. Morton, D.F. Mayers, Numerical Solution of Partial Differential Equations, Cambridge Univ. Press, 1994.
[32] M. Francisquez, B. Zhu, B.N. Rogers, Multigrid treatment of implicit continuum diffusion, Comput. Phys. Commun. 236 (2019) 104–117.
[33] U. Trottenberg, C.W. Oosterlee, A. Schuller, Multigrid, Elsevier, 2000.
[34] A. Stuart, A.R. Humphries, Dynamical Systems and Numerical Analysis, 2, Cambridge Univ. Press, 1998.
[35] J.W. Choi, H.G. Lee, D. Jeong, J. Kim, An unconditionally gradient stable numerical method for solving the Allen–Cahn equation, Phys. A Stat. Mech. Appl. 388 (9) (2009) 1791–1803.
[36] S. Ham, J. Kim, Stability analysis for a maximum principle preserving explicit scheme of the Allen–Cahn equation, Math. Comput. Simul. 207 (2023) 453–465.