Computer Physics Communications 304 (2024) 109301

Contents lists available at ScienceDirect

Computer Physics Communications

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

Computational Physics

A modified Allen–Cahn equation with a mesh size-dependent interfacial parameter on a triangular mesh ☆

Junxiang Yang [a], Jian Wang [b], Soobin Kwak [c], Seokjun Ham [c], Junseok Kim [c,*]

[a] *School of Computer Science and Engineering, Faculty of Innovation Engineering, Macau University of Science and Technology, Macao Special Administrative Region of China*
[b] *School of Mathematics and Statistics, Nanjing University of Information Science and Technology, Nanjing, 210044, China*
[c] *Department of Mathematics, Korea University, Seoul, 02841, Republic of Korea*

ARTICLE INFO

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

Keywords:
Modified Allen–Cahn equation
Triangular mesh
Mesh size-dependent parameter

ABSTRACT

In this article, we propose a modified Allen–Cahn (AC) equation with a space-dependent interfacial parameter. When numerically solving the AC equation with a constant interfacial parameter over large domains, a substantial number of grid points are essential, which leads to significant computational costs. To effectively resolve this problem, numerous adaptive mesh techniques have been developed and implemented. These methods use locally refined meshes that adaptively track the interfacial positions of the phase field throughout the simulation. However, the data structures for adaptive algorithms are generally complex, and the problems to be solved may involve challenges at multiple scales. In this article, we present a modified AC equation with a mesh size-dependent interfacial parameter on a triangular mesh to efficiently solve multi-scale problems. In the proposed method, a triangular mesh is used, and the interfacial parameter value at a node point is defined as a function of the average length of the edges connected to the node point. The proposed algorithm effectively uses large and small values of the interfacial parameter on coarse and fine meshes, respectively. To demonstrate the efficiency and superior performance of the proposed method, we conduct several representative numerical experiments. The computational results indicate that the proposed interfacial function can adequately evolve the multi-scale phase interfaces without excessive relaxation or freezing of the interfaces. Finally, we provide the main source code for the methodology, including mesh generation as described in this paper, so that interested readers can use it.

1. Introduction

In this article, we present a modified Allen–Cahn (AC) equation with a space-dependent interfacial parameter:

$$\frac{\partial \phi(\mathbf{x},t)}{\partial t} = -\frac{F'(\phi(\mathbf{x},t))}{\epsilon^2(\mathbf{x})} + \Delta\phi(\mathbf{x},t),\ \mathbf{x}\in\Omega,\ t>0, \tag{1}$$

where $\phi(\mathbf{x},t)$ is the phase-field function at spatial coordinates $\mathbf{x}=(x,y)$ and time t, $F(\phi(\mathbf{x},t)) = 0.25(\phi^2(\mathbf{x},t)-1)^2$, which is a polynomial approximation of a logarithmic free energy [1], and $\epsilon(\mathbf{x})$ is the space-dependent interfacial parameter. We numerically solve Eq. (1) with appropriate boundary and initial conditions. We note that if $\epsilon(\mathbf{x}) = \epsilon_0$ for a constant ϵ_0, then Eq. (1) transforms into the standard AC equation [2,3], which was initially proposed as a phenomenological equation for anti-phase domain coarsening in binary alloys. Equation (1) can be derived from the L^2-gradient flow of the total free energy functional:

$$\mathcal{E}(\phi) = \int_\Omega \left[\frac{F(\phi(\mathbf{x},t))}{\epsilon^2(\mathbf{x})} + \frac{1}{2}|\nabla\phi(\mathbf{x},t)|^2\right] d\mathbf{x}. \tag{2}$$

The AC equation is an important mathematical model due to its capacity to describe phase transitions and interface dynamics across various physical systems, including materials science, physics, and biology [4]. It plays a crucial role in understanding phenomena such as crystal growth, triply periodic structure formation [5], grain boundary motion, solidification in metallic alloys [6], tumor growth, microstructure evo-

☆ The review of this paper was arranged by Prof. David W. Walker.
* 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.cpc.2024.109301
Received 23 February 2024; Received in revised form 12 June 2024; Accepted 26 June 2024
Available online 2 July 2024
0010-4655/© 2024 Elsevier B.V. All rights are reserved, including those for text and data mining, AI training, and similar technologies.

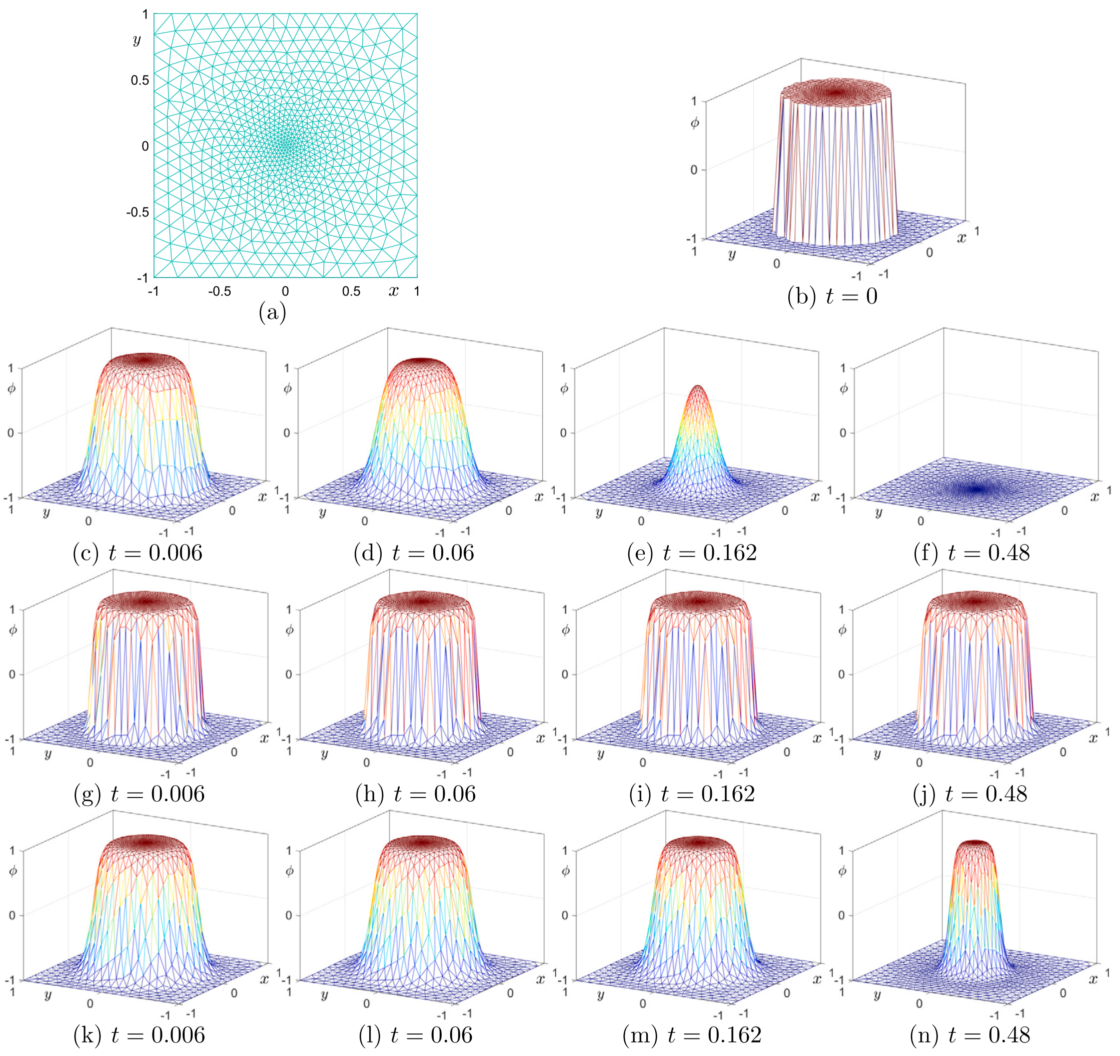

Fig. 1. Effect of constant interfacial parameter ϵ_0: (a) a non-uniform triangular mesh, (b) the initial condition. (c)–(f), (g)–(j), and (k)–(n) are the temporal evolutions of the phase fields with a large $\epsilon_0 = 0.1$, a small $\epsilon_0 = 0.032$, and the proposed $\epsilon(\mathbf{x})$, respectively. The times are indicated beneath each figure.

lution, and pattern formation, making it a building block in modeling complex processes and phenomena across diverse scientific disciplines. Recently, the AC equation was used as a benchmark problem for machine learning algorithms [7,8].

When numerically solving the AC equation with a constant interfacial parameter over large domains, a substantial number of grid points are essential, which leads to significant computational costs. To resolve this problem, various adaptive finite difference methods (FDM) [9], finite element methods (FEM) [10–12], and finite volume methods (FVM) [13] have been developed. These adaptive methods use locally refined meshes that adaptively track the interfacial positions of the phase field. Liu et al. [10] considered the scalar auxiliary variable (SAV) weak Galerkin FEM and the time-space adaptive algorithm for the AC equation. Chen et al. [11] proposed an adaptive, second-order accurate, and unconditionally stable FEM for the AC equation. Joshi and Jaiman [12] proposed a stable and robust adaptive variational partitioned procedure to numerically solve the Navier–Stokes (NS) and AC equations for multi-phase fluid flows. Li et al. [13] presented the discontinuous finite volume element method (DFVEM) and the backward Euler scheme to solve the AC equation. Poochinapan and Wongsaijai [14] developed a fourth-order FDM for solving the AC equation in both 1D and 2D. Hwang et al. [15] developed a simple and efficient computational scheme for the AC equation on effective symmetric triangular meshes. Celiker and Lin [16] presented an efficient triangular finite element method (FEM) with exponential mesh refinement to solve the AC equation in non-convex polygons, which overcame the effect of strong corner singularities. Li et al. [19] presented a polygonal mesh adaptation technique for a fully implicit method using discontinuous Galerkin FEMs in space and an Euler scheme to numerically solve the AC equation.

However, the data structure for the adaptive algorithm is generally complex, and the problems to be solved may involve multiple scales. If a stationary triangular mesh with significantly varying triangle sizes is used, then it can lead to various issues. When a large interfacial parameter is used, the interfacial profile becomes overly diffuse on the fine mesh. On the contrary, using a small interfacial parameter results in an excessively steep interfacial profile on the coarse mesh and causes the temporal evolution to become pinned. Fig. 1 illustrates how different values of the interfacial parameter affect the evolution dynamics of the phase fields in the numerical solutions of the AC equation.

Figs. 1(a) and (b) display a non-uniform triangular domain and an initial profile on $\Omega = (-1,1)\times(-1,1)$:

$$\phi(x,y,0) = \begin{cases} 1, & \text{if } x^2+y^2 < 0.6, \\ -1, & \text{otherwise.} \end{cases}$$

Fig. 1(c)–(e), (f)–(h), and (i)–(k) are the temporal evolutions of the phase fields with a large $\epsilon_0 = 0.1$, a small $\epsilon_0 = 0.032$, and the proposed interfacial parameter $\epsilon(\mathbf{x})$, respectively. A large interfacial parameter,

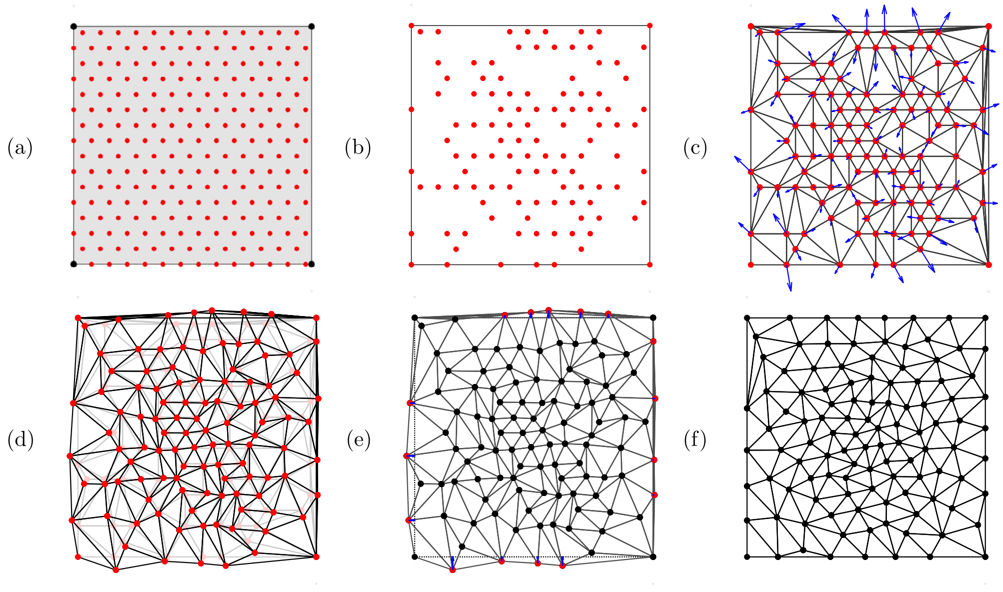

Fig. 2. Schematic illustration of generating the DistMesh. (a) Generated uniform nodes in the domain. (b) Nodes are removed by applying a weight function w. (c) Net force $\mathbf{F}$ in current triangulation. (d) Arrangement of nodes via $\Delta t\mathbf{F}$. (e) Projection of the nodes located outside into the boundary using Eq. (8). (f) Final result of unstructured mesh by using the DistMesh algorithm.

$\epsilon_0 = 0.1$, is appropriate when the mesh size is large, as shown in Fig. 1(c). However, the same $\epsilon_0 = 0.1$ is relatively too large when the mesh size is small, as displayed in Fig. 1(e), which leads to the interfacial profile becoming overly diffuse on the fine mesh. On the contrary, using a small interfacial parameter, $\epsilon_0 = 0.032$, results in an excessively steep interfacial profile on the coarse mesh and causes the temporal evolution to become pinned as shown in Fig. 1(h).

To resolve these problems, in this article, we present a modified AC equation with a mesh size-dependent interfacial parameter on a triangular mesh to efficiently solve multi-scale problems. In the proposed method, a triangular mesh is used, and the interfacial parameter value at a node point is defined as a function of the average length of the edges connected to the node point. The proposed algorithm effectively uses large values of the interfacial parameter at coarse meshes and small values at fine meshes. To highlight this feature of the proposed method, we conduct several computational experiments such as data classification and multi-scale simulations.

The structure of this document is outlined as follows. In Section 2, we present a computational algorithm for the proposed modified AC equation with a mesh size-dependent interfacial parameter on a triangular mesh. In Section 3, several computational experiments are conducted to validate the high performance of the proposed modified AC model.

2. Numerical method

Now, we present a numerical method for the proposed modified AC equation with a mesh size-dependent interfacial parameter on a triangular mesh. We adopt the operator splitting method for the AC equation [17], where the authors solved the linear term using an explicit Euler method and the nonlinear term using a closed-form solution on a uniform Cartesian mesh. In this study, we apply the operator splitting method on a nonuniform triangular mesh. To apply the operator splitting method [18,20–22], we first express the modified AC equation (1) as follows:

$$\frac{\partial \phi(\mathbf{x},t)}{\partial t} = \mathcal{N}\phi(\mathbf{x},t) + \mathcal{L}\phi(\mathbf{x},t), \tag{3}$$

where $\mathcal{N}\phi(\mathbf{x},t) = -F'(\phi(\mathbf{x},t))/\epsilon^2(\mathbf{x})$ and $\mathcal{L}\phi(\mathbf{x},t) = \Delta\phi(\mathbf{x},t)$. Then, we sequentially solve the following two equations:

$$\frac{\partial u(\mathbf{x},t)}{\partial t} = \mathcal{L}u(\mathbf{x},t), \tag{4}$$

$$\frac{\partial v(\mathbf{x},t)}{\partial t} = \mathcal{N}v(\mathbf{x},t). \tag{5}$$

Given a time step Δt, let $u(\mathbf{x},0) = \phi(\mathbf{x},t)$ and we solve Eq. (4) to get the solution $u(\mathbf{x},\Delta t)$. Next, let $v(\mathbf{x},0) = u(\mathbf{x},\Delta t)$ and we solve Eq. (5) to get the solution $v(\mathbf{x},\Delta t)$, which is $\phi(\mathbf{x},t+\Delta t)$. Throughout this study, we use the following conventions for Eqs. (4) and (5) unless there exists ambiguity:

$$\frac{\partial \phi(\mathbf{x},t)}{\partial t} = \Delta\phi(\mathbf{x},t), \tag{6}$$

$$\frac{\partial \phi(\mathbf{x},t)}{\partial t} = -\frac{F'(\phi(\mathbf{x},t))}{\epsilon^2(\mathbf{x})}. \tag{7}$$

2.1. *DistMesh method for generating a triangular mesh*

To generate a triangular mesh, we use the DistMesh algorithm [23], which is a computational method for generating unstructured triangular and tetrahedral meshes. This method optimally distributes mesh nodes based on prescribed quality criteria and it enables efficient numerical simulations in various engineering and scientific applications. The subsequent process describes the complete algorithm of the DistMesh for generating an unstructured triangular mesh. A function $w(x,y) = 1+\sqrt{x^2+y^2}$ is adopted as the weight function on domain $\Omega = (L_x, R_x)\times(L_y, R_y)$. A function $d(x,y) = -\min\left\{x-L_x, R_x-x, y-L_y, R_y-y\right\}$ is adopted as the signed distance function. Fig. 2 shows the entire procedure of the DistMesh method, and the followings are the detailed steps.

Step 1. Generate initial nodes $\mathbf{X}^0$ in domain and remove nodes by applying a weight function w.
Step 2. Compute the Delaunay triangulation with nodes $\mathbf{X}^n$ and compute the net force $\mathbf{F}$.
Step 3. Update the current position of nodes $\mathbf{X}^n$ to $\mathbf{X}^{n+1/2}$ by adding $\Delta t\mathbf{F}$ to $\mathbf{X}^n$.
Step 4. Push back the nodes that lie beyond the boundary onto the boundary using the following equation.

$$\mathbf{X}_i^{n+1} = \mathbf{X}_i^{n+\frac{1}{2}} - d\left(\mathbf{X}_i^{n+\frac{1}{2}}\right)\frac{\nabla d\left(\mathbf{X}_i^{n+\frac{1}{2}}\right)}{\left|\nabla d\left(\mathbf{X}_i^{n+\frac{1}{2}}\right)\right|^2}. \tag{8}$$

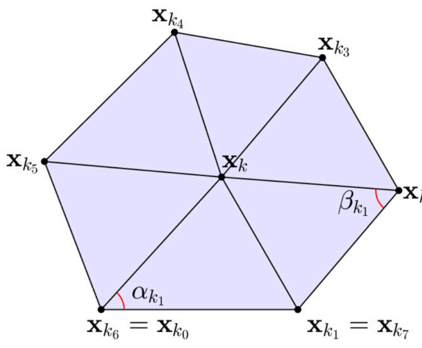

Fig. 3. Schematic illustration of the triangles that are contact with the node point $\mathbf{x}_k$.

Step 5. Iterate **Step 2–5** until the level of the total movement of nodes decreases below a specified tolerance threshold.

Additional information and comprehensive details regarding the mesh generation process can be found in the reference [23].

2.2. Numerical solution algorithm

On the two-dimensional triangular mesh with N node points $\mathbf{x}_k = (x_k, y_k)$, for $1 \le k \le N$, we first solve Eq. (6):

$$\frac{\phi_k^* - \phi_k^n}{\Delta t} = \Delta_d \phi_k^n, \tag{9}$$

where $\phi_k^n = \phi(\mathbf{x}_k, n\Delta t)$, Δt is a time step, and $\Delta_d \phi_k^n$ is the discrete Laplace operator [24], which is defined as follows.

$$\Delta_d \phi_k = \frac{3}{A(\mathbf{x}_k)} \sum_{m=1}^{N_k} \frac{\cot \alpha_{k_m} + \cot \beta_{k_m}}{2} (\phi_{k_m} - \phi_k), \tag{10}$$

where $A(\mathbf{x}_k)$ is the sum of areas of triangles that are in contact with the node point $\mathbf{x}_k$ and N_k is the number of node points neighboring $\mathbf{x}_k$. For example, in Fig. 3, $\mathbf{x}_k$ has $\{\mathbf{x}_{k_1}, \mathbf{x}_{k_2}, \mathbf{x}_{k_3}, \mathbf{x}_{k_4}, \mathbf{x}_{k_5}, \mathbf{x}_{k_6}\}$ as one-ring neighboring points and $\alpha_{k_m} = \angle \mathbf{x}_k \mathbf{x}_{k_{m-1}} \mathbf{x}_{k_m}$ and $\beta_{k_m} = \angle \mathbf{x}_k \mathbf{x}_{k_{m+1}} \mathbf{x}_{k_m}$ for $m = 1, \cdots, N_k$, where $\mathbf{x}_{k_0} = \mathbf{x}_{k_{N_k}}$ and $\mathbf{x}_{k_{N_k+1}} = \mathbf{x}_{k_1}$. For example, when $m = 1$, $\alpha_{k_1} = \angle \mathbf{x}_k \mathbf{x}_{k_0} \mathbf{x}_{k_1}$ and $\beta_{k_1} = \angle \mathbf{x}_k \mathbf{x}_{k_2} \mathbf{x}_{k_1}$, as shown in Fig. 3.

From Eq. (9), we have

$$\phi_k^* = \phi_k^n + \Delta t \Delta_d \phi_k^n, \quad 1 \le k \le N. \tag{11}$$

In this paper, we propose a space-dependent interfacial parameter at a vertex $\mathbf{x}_k$, which is defined by averaging the lengths of edges that are in contact with the vertex as follows:

$$\epsilon(\mathbf{x}_k) = \frac{s}{N_k} \sum_{m=1}^{N_k} \left| \mathbf{x}_k - \mathbf{x}_{k_m} \right|, \tag{12}$$

where s is a scaling parameter. Next, we solve the nonlinear Eq. (7) analytically with the initial condition $\phi_k^0 = \phi_k^*$ and temporal step Δt:

$$\phi_k^{n+1} = \frac{\phi_k^*}{\sqrt{[1 - (\phi_k^*)^2] e^{-\frac{2\Delta t}{\epsilon^2(\mathbf{x}_k)}} + (\phi_k^*)^2}}. \tag{13}$$

For the boundary condition, we use the homogeneous Neumann boundary condition. Let $\mathbf{x}_b$ be a boundary point and $\mathcal{N}_b$ be the one-ring neighborhood of $\mathbf{x}_b$ excluding boundary points. For simplicity of exposition, we define the value of ϕ_b^n on the boundary point as the average of the neighboring points which are interior of the domain:

$$\phi_b^n = \frac{1}{\#\left(\mathcal{N}_b\right)} \sum_{\mathbf{x}_k \in \mathcal{N}_b} \phi_k^n,$$

where $\#(\mathcal{N}_b)$ is the number of points in $\mathcal{N}_b$. Fig. 4 shows the boundary points ($\star$) and the interior neighborhood ($\bullet$) of boundary points on a triangular mesh.

Because the proposed scheme is fully explicit, there is no requirement to solve a system of discrete equations implicitly. Therefore, implementation is straightforward and computational speed is very fast. Now, let us consider the maximum principle [25,26] for the proposed explicit numerical method, which implies stability. Let $\|\phi\|_\infty = \max_{1 \le k \le N} |\phi_k|$ be the discrete maximum norm, $\|\phi^n\|_\infty \le 1$, and all triangles be acute. From Eq. (9), we obtain

$$|\phi_k^*| = \left| \left(1 - \frac{3\Delta t}{A(\mathbf{x}_k)} \sum_{m=1}^{N_k} \frac{\cot \alpha_{k_m} + \cot \beta_{k_m}}{2} \right) \phi_k^n + \frac{3\Delta t}{A(\mathbf{x}_k)} \sum_{m=1}^{N_k} \frac{\cot \alpha_{k_m} + \cot \beta_{k_m}}{2} \phi_{k_m}^n \right| \tag{14}$$

$$\le \left(1 - \frac{3\Delta t}{A(\mathbf{x}_k)} \sum_{m=1}^{N_k} \frac{\cot \alpha_{k_m} + \cot \beta_{k_m}}{2} \right) \|\phi^n\|_\infty + \frac{3\Delta t}{A(\mathbf{x}_k)} \sum_{m=1}^{N_k} \frac{\cot \alpha_{k_m} + \cot \beta_{k_m}}{2} \|\phi^n\|_\infty \tag{15}$$

$$\le 1, \qquad \text{for } k = 1, \ldots, N, \tag{16}$$

where we have used the condition

$$1 - \frac{3\Delta t}{A(\mathbf{x}_k)} \sum_{m=1}^{N_k} \frac{\cot \alpha_{k_m} + \cot \beta_{k_m}}{2} \ge 0, \ k = 1, \ldots, N, \tag{17}$$

which can be rewritten as

$$\Delta t \le \min_{1 \le k \le N} \frac{2A(\mathbf{x}_k)}{3 \sum_{m=1}^{N_k} (\cot \alpha_{k_m} + \cot \beta_{k_m})}. \tag{18}$$

Therefore, $\|\phi^*\|_\infty \le 1$ holds for Δt satisfying Eq. (18). Next, from Eq. (13), we obtain

$$|\phi_k^{n+1}| = \frac{|\phi_k^*|}{\sqrt{\left[1 - \left(\phi_k^*\right)^2\right] e^{-\frac{2\Delta t}{\epsilon^2(\mathbf{x})}} + \left(\phi_k^*\right)^2}} \le 1, \ k = 1, \ldots, N, \tag{19}$$

where we have used the non-negative coefficient condition, $1 - \left(\phi_k^*\right)^2 \ge 0$ because of $\|\phi^*\|_\infty \le 1$. Hence, the maximum principle $\|\phi^{n+1}\|_\infty \le 1$ holds for Δt satisfying Eq. (18).

We note that the only restriction on time stepping in the numerical solution algorithm arises from the diffusion Eq. (9) because the second numerical step Eq. (13) of the operator splitting method is unconditionally stable. Please refer to [17] for details on an operator splitting method for the AC equation on Cartesian grids. Furthermore, this operator splitting scheme has a better stability condition compared to the fully explicit time-stepping method like that considered in [27].

In this study, we focused on proposing a novel modified AC equation with a mesh size-dependent interfacial parameter on a triangular mesh. For simplicity of exposition, we solve the linear diffusion equation by using an explicit method, which has time step restrictions. We may use implicit methods [28] to numerically solve the linear diffusion equation.

We also note that the AC equation is a gradient flow of the total energy functional (2). Numerous numerical methods based on discrete energies have energy stability, where discrete energy decreases, which implies the stability of the numerical schemes [28]. In this study, we use the operator splitting method, where we directly verify stability by demonstrating the stabilities of each splitting step. Recently, authors in [29] investigated Strang operator splitting methods for AC equations

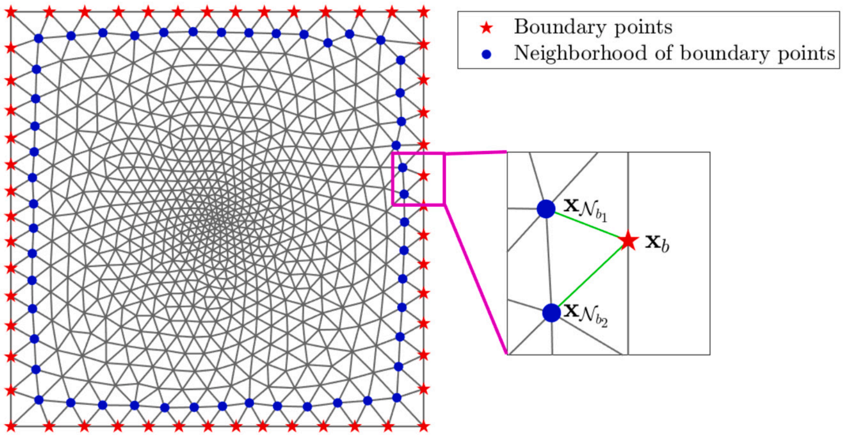
★ Boundary points
● Neighborhood of boundary points
$\mathbf{x}_{\mathcal{N}_{b_1}}$
$\mathbf{x}_b$
$\mathbf{x}_{\mathcal{N}_{b_2}}$

Fig. 4. Schematic illustration of triangular mesh and boundary points.

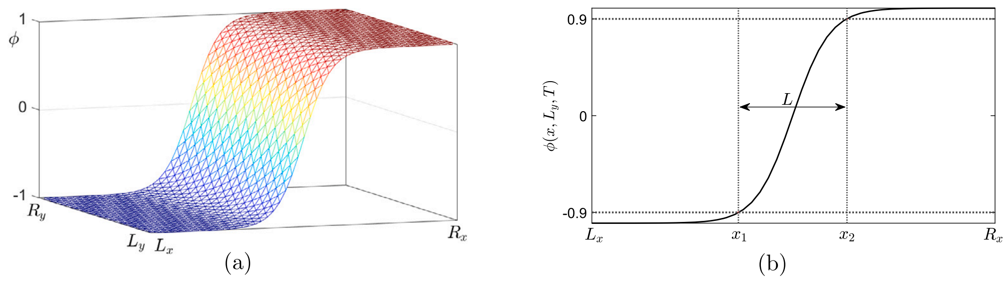

Fig. 5. Schematic illustration for the equilibrium solution (a) and the thickness of the transition layer (b).

and proved strict energy dissipation with a judiciously modified energy. Zhang et al. [30] developed a unified method of a fourth-order inequality-preserving scheme satisfying forward Euler conditions by introducing a stabilization parameter dependent on the time-step.

3. Computational experiments

In this section, we present several computational experiments to demonstrate the performance of the proposed modified AC equation and its numerical solution.

3.1. Relation between the thickness of the transition layer and ϵ_0 value

To investigate the relationship between the thickness of the transition layer and the $\epsilon(\mathbf{x}) = \epsilon_0$ value, we consider the following initial condition on the computational domain $\Omega = (-5, 5) \times (2, 2)$:

$$\phi(x, y, 0) = \begin{cases} -1, & \text{if } x < 0, \\ 1, & \text{otherwise.} \end{cases}$$

Here, we use a constant interfacial parameter $\epsilon(\mathbf{x}) = \epsilon_0$ and $\Delta t = 0.012$ on a nearly uniform triangular mesh consisting of 1214 points. The average length of edges of all triangles is denoted by h_{ave}, and its value is approximately 0.2019. We define a numerical equilibrium solution as $\phi^\infty = \phi^{n+1}$ if $\|\phi^{n+1} - \phi^n\|_\infty < 1e\text{-}6$ for some n, see Fig. 5(a). For the equilibrium solution ϕ^∞, we define the thickness of the transition layer as $L = |x_2 - x_1|$, where $\phi(x_1, 0, (n+1)\Delta t) = -0.9$ and $\phi(x_2, 0, (n+1)\Delta t) = 0.9$ as displayed in Fig. 5(b).

Fig. 6 shows the initial condition; equilibrium solutions with $\epsilon_0 = h_{ave}$, $3h_{ave}$, and $5h_{ave}$; and relationship between the thickness of the transition layer and the constant interfacial parameter ϵ_0. From Fig. 6(b)–(d), we can observe that the interfacial transition layer widens as the value of ϵ_0 increases. Fig. 6(e) shows the relationship between the thickness of the transition layer and the constant interfacial parameter ϵ_0 along with a linear fit to the data. The linear fitting function is given as

$$L(\epsilon_0) = 4.1498\epsilon_0 + 0.0879, \tag{20}$$

where we used the data with $\epsilon_0 = h_{ave}$, $2h_{ave}$, $3h_{ave}$, $4h_{ave}$, and $5h_{ave}$. By inverting Eq. (20), we can derive the following formula for ϵ_0:

$$\epsilon_0(L) = (L - 0.0879)/4.1498, \tag{21}$$

which can be used when we want to control the thickness of the interfacial layer L.

3.2. Convergence test

In this section, we investigate the convergence of temporal accuracy for the proposed scheme. To evaluate the errors and convergence rates, we compare the reference and numerical solutions [31]. The reference solution $\phi_{\text{ref}}^{N_t}$ is obtained with a sufficiently small reference time step $\Delta t_{\text{ref}} = 3.124 \times 10^{-7}$, where N_t is the total number of temporal iterations and $T = N_t \Delta t$ is the final time. In the following convergence test, we set $T = 0.016$ for the reference solution. The discrete error at time T is defined as $\mathbf{e}(T) = (e_1(T), e_2(T), \ldots, e_N(T))$, where $e_k(T) = \phi_k^{\frac{\Delta t_{\text{ref}}}{\Delta t} N_t} - (\phi_{\text{ref}})_k^{N_t}$ for $k = 1, 2, \ldots, N$. We define discrete l_2-error as $\|\mathbf{e}^{N_t}\|_2 = \sqrt{\sum_{k=1}^{N} (e_k(T))^2 A(\mathbf{x}_k)/3}$, where $A(\mathbf{x}_k)$ is the sum of areas of triangles that contact the node point $\mathbf{x}_k$. We conduct a convergence test with the following initial condition on the computational domain $(-1, 1) \times (-1, 1)$:

$$\phi(x, y, 0) = \tanh\left(\frac{0.4 - \sqrt{x^2 + y^2}}{0.015\sqrt{2}}\right).$$

The triangulated computational domain and given initial condition are illustrated in Figs. 7(a) and (b), respectively. Here, we use the space-

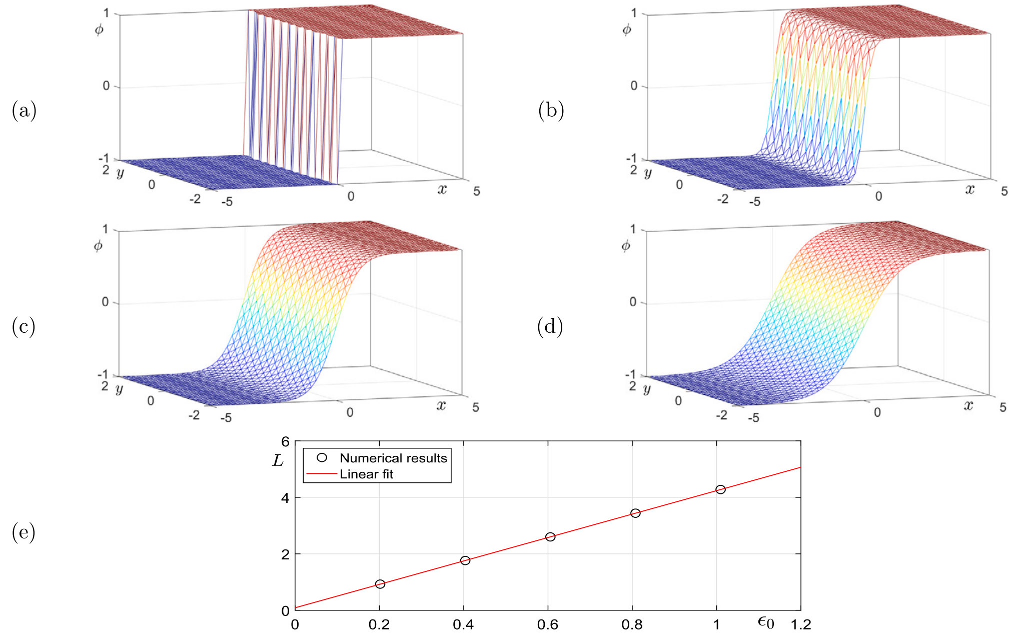

Fig. 6. (a) Initial condition. (b)–(d) Equilibrium solutions with $\epsilon_0 = h_{ave}$, $3h_{ave}$, and $5h_{ave}$, respectively. (e) Relation between the thickness of the transition layer and the constant interfacial parameter ϵ_0 along with a linear fit to the data.

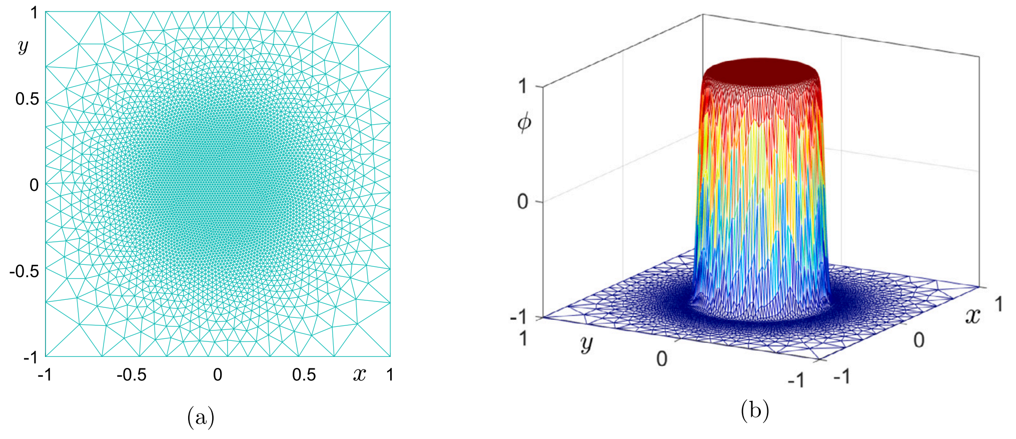

Fig. 7. (a) Triangulated computational domain and (b) initial condition for the convergence test.

Table 1
Temporal errors and convergence rates at $T = 0.016$.

Δt	$8\Delta t_{\text{ref}}$		$16\Delta t_{\text{ref}}$		$32\Delta t_{\text{ref}}$		$64\Delta t_{\text{ref}}$
l_2-error	0.0136		0.0293		0.0616		0.1292
rate		1.1105		1.0696		1.0688	

dependent interfacial parameter with $s = 1$. Table 1 lists the temporal errors and convergence rates between the reference and numerical solutions with time steps of $\Delta t = 8\Delta t_{\text{ref}}$, $16\Delta t_{\text{ref}}$, $32\Delta t_{\text{ref}}$, and $64\Delta t_{\text{ref}}$. We can verify that the temporal accuracy of the proposed algorithm is first-order.

3.3. Separation of domain

Separation of domain is important in classification to make distinct categories remain independent. This prevents overlap and ambiguity and increases the accuracy and reliability of classification models by clearly defining boundaries between different classes or groups of data. We consider the separation of domain using the proposed modified AC equation in two-dimensional space $\Omega = (-1,1) \times (-1,1)$ with a space-dependent interfacial parameter and non-uniform triangular mesh of 1786 points. The triangulated computational domain and its interfacial parameter $\epsilon(\mathbf{x}_k)$ are shown in Fig. 8(a). The triangles are smaller at the center of the domain and gradually increase in size as they move away from the center. As shown in Fig. 8(b), the initial configuration is given as

$$\phi(x,y,0) = \begin{cases} 1, & \text{if } x^2 + y^2 < 0.2 \text{ and } y \geq 0, \\ -1, & \text{if } x^2 + y^2 < 0.2 \text{ and } y < 0, \\ 0, & \text{otherwise.} \end{cases} \quad (22)$$

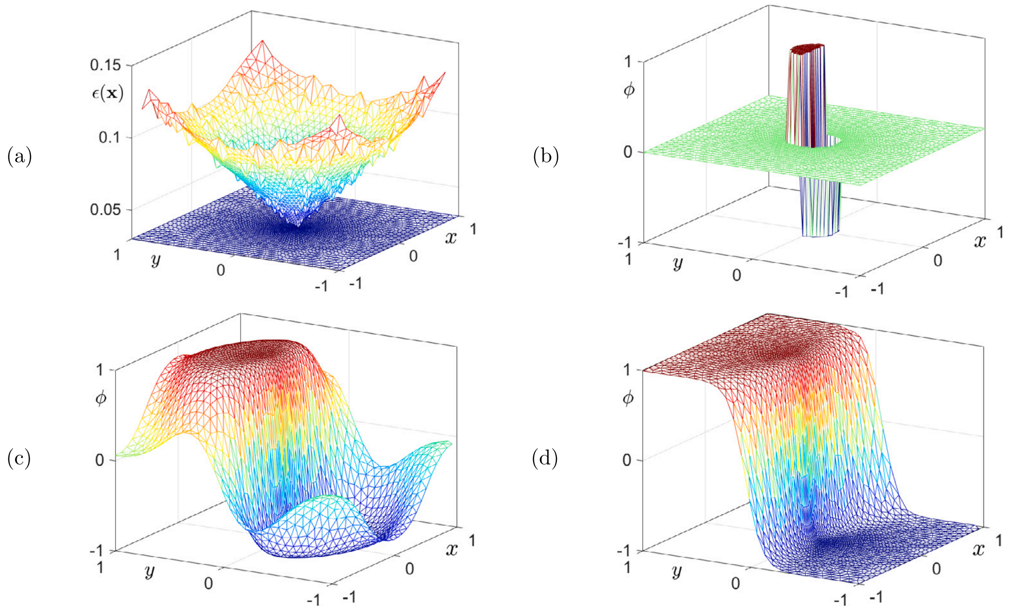

Fig. 8. (a) Triangulated computational domain and its interfacial parameter $\epsilon(\mathbf{x})$. (b)–(d) Temporal evolution of the proposed modified AC equation with the space-dependent interfacial parameter at $t = 0$, $600\Delta t$, and $1200\Delta t$, respectively.

Figs. 8(b)–(d) show the temporal evolution with the given initial condition (22). Here, we use $s = 1.5$ and a time-step size of $\Delta t = 7.8695e\text{-}5$. As is shown in Fig. 8(d), the phases of the numerical solution are well separated throughout the entire domain. Initially, the non-zero phase-field values are defined in the fine mesh at the center of the domain, and these non-zero phase-field values propagate through the entire domain as time progresses. This approach is highly efficient because it allows for a detailed mesh placement initially and a coarser mesh away from that region to optimize computational efficiency.

3.4. Multi-scale problem

Multi-scale problems are important because they involve phenomena occurring at various scales and provide insights into complex systems such as materials, fluids, and biological structures, which enable accurate modeling and understanding of complex behaviors and interactions within these systems. Let us consider the following initial condition with two disks on a two-dimensional space $\Omega = (-1, 1) \times (-1, 1)$:

$$\phi(x, y, 0) = \begin{cases} 1, & \text{if } (x-0.3)^2 + (y-0.3)^2 < 0.4 \text{ or} \\ & (x+0.4)^2 + (y+0.4)^2 < 0.1, \\ -1, & \text{otherwise,} \end{cases} \tag{23}$$

$$\phi(x, y, t) = -1 \text{ for } (x, y) \in \partial\Omega,\ t \geq 0. \tag{24}$$

Here, we use Dirichlet boundary conditions. The non-uniform triangulated computational domain becomes coarser as it moves away from the center of the small disk. Fig. 9(a) shows the non-uniform triangulated domain. The given initial condition with the non-uniform triangulated domain is shown in Fig. 9(b). The columns in Figs. 9(c)–(e) show the temporal evolutions of the AC equation under various ϵ values: large constant epsilon $\epsilon(\mathbf{x}) = 0.08$, small constant epsilon $\epsilon(\mathbf{x}) = 0.01$, and the mesh size-dependent interfacial parameter with $s = 1$ in Eq. (12). Here, the time-step size is taken as $\Delta t = 7.6360 \times 10^{-5}$.

As shown in Fig. 9(c), when a large constant $\epsilon(\mathbf{x}) = 0.08$ is used, it is appropriate for the coarse mesh region. However, it is too large in the fine mesh region and results in a widened interfacial transition layer. Meanwhile, as displayed in Fig. 9(d), when using a small constant $\epsilon(\mathbf{x}) = 0.01$, it is suitable for the fine mesh region. However, it is too small in the coarse mesh region, and results in a steep interfacial transition layer and pins the evolution of the phase-field in that region. Fig. 9(d) demonstrates the superior performance of the proposed mesh size-dependent interfacial parameter with $s = 1$, which adaptively adjusts its values based on the mesh sizes. Specifically, $\epsilon(\mathbf{x})$ becomes small and large on fine and coarse meshes, respectively.

3.5. Traveling wave solution

We investigate a traveling wave solution to the AC equation [32], where the interface of the phases evolves over time. We generate a nonuniform triangulated domain that coarsens linearly with respect to the x-direction. On a 2D computational domain $\Omega = (0, 5) \times (0, 1)$, we take the following initial condition for a traveling wave solution to the AC equation:

$$\phi(x, y, 0) = \begin{cases} 1, & \text{if } x < 0.2 \\ 0, & \text{otherwise.} \end{cases}$$

Fig. 10 shows the temporal evolution of the traveling wave solution to the AC equation on a nonuniform triangulated mesh. Here, we use a time step of $\Delta t = 4.5918 \times 10^{-4}$ and a scaling parameter of $s = 0.5$. In the context of the AC equation, the traveling wave solution describes the evolution of phase transitions, such as the formation of domain boundaries between distinct phases. These solutions illustrate how local interactions between neighboring points lead to the propagation of these phase transitions over time. These solutions are essential for understanding phenomena like phase separation and pattern formation in diverse fields such as physics, materials science, and biology.

3.6. Data classification

Next, we consider an application of the proposed method: data classification, where our proposed algorithm demonstrates efficiency and superior performance. Data classification is a very important method in data analysis and involves the organization of unstructured data into consistent structures. Classification involves assigning new data to one of the predetermined classes. One of the classification techniques involves generating a decision boundary that separates classes. There are various methods for generating decision boundaries in real-world applications, including: support vector machine [33–35], neural network

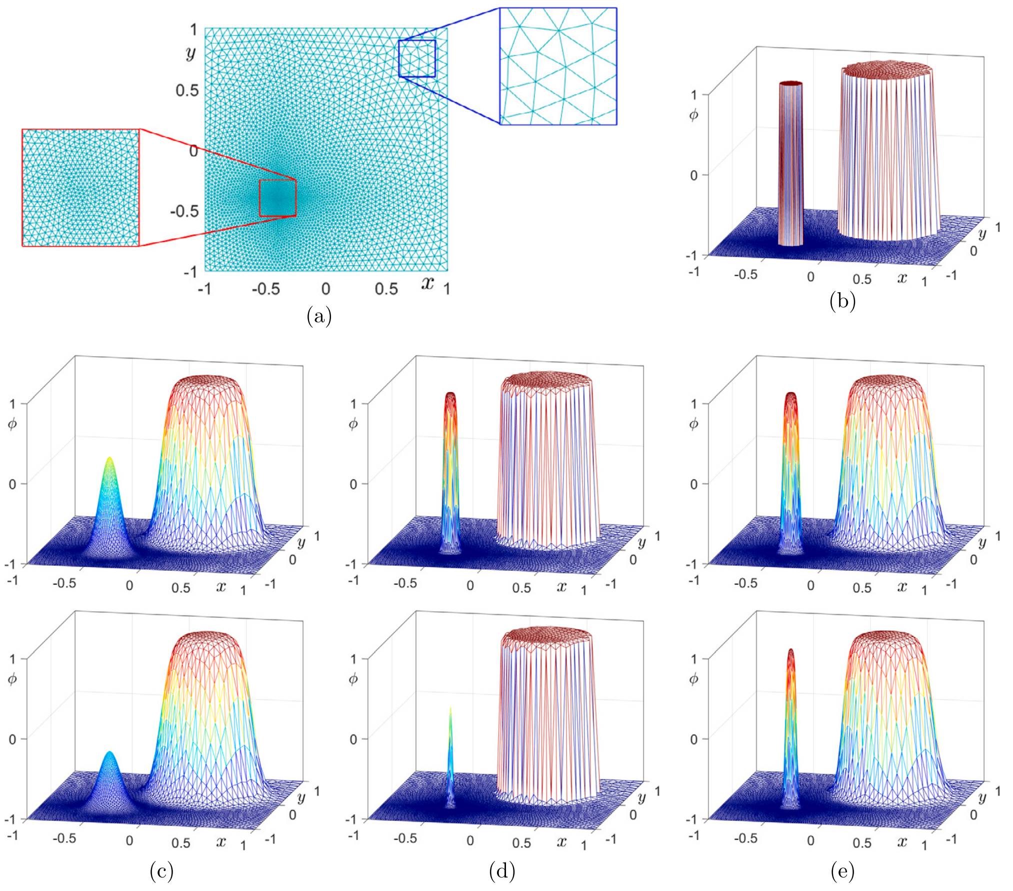
Fig. 9. (a) Non-uniform triangular mesh. (b) Given initial condition. (c)–(e) Temporal evolutions of the AC equation under different $\epsilon(\mathbf{x})$ values: (c) a large constant $\epsilon(\mathbf{x}) = 0.08$; (d) a small constant $\epsilon(\mathbf{x}) = 0.01$; and (e) a mesh size-dependent interfacial parameter with $s = 1$. From the second to third row, the time values are $t = 150\Delta t$ and $t = 250\Delta t$, respectively.

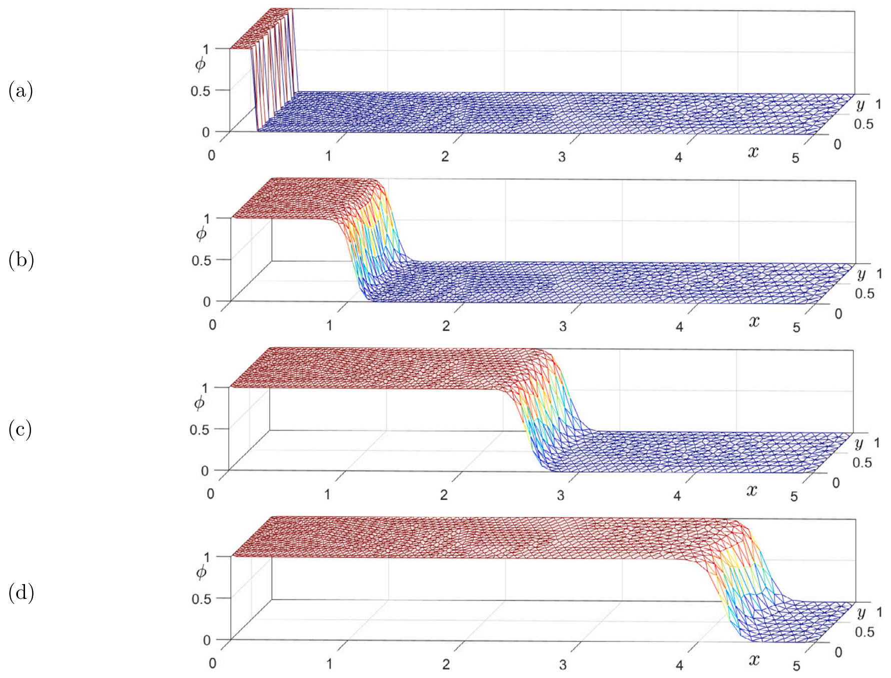
Fig. 10. Temporal evolution of the traveling wave solution to the AC equation on a nonuniform triangular mesh at (a) $t = 0$, (b) $t = 30\Delta t$, (c) $t = 90\Delta t$, and (d) $t = 170\Delta t$.

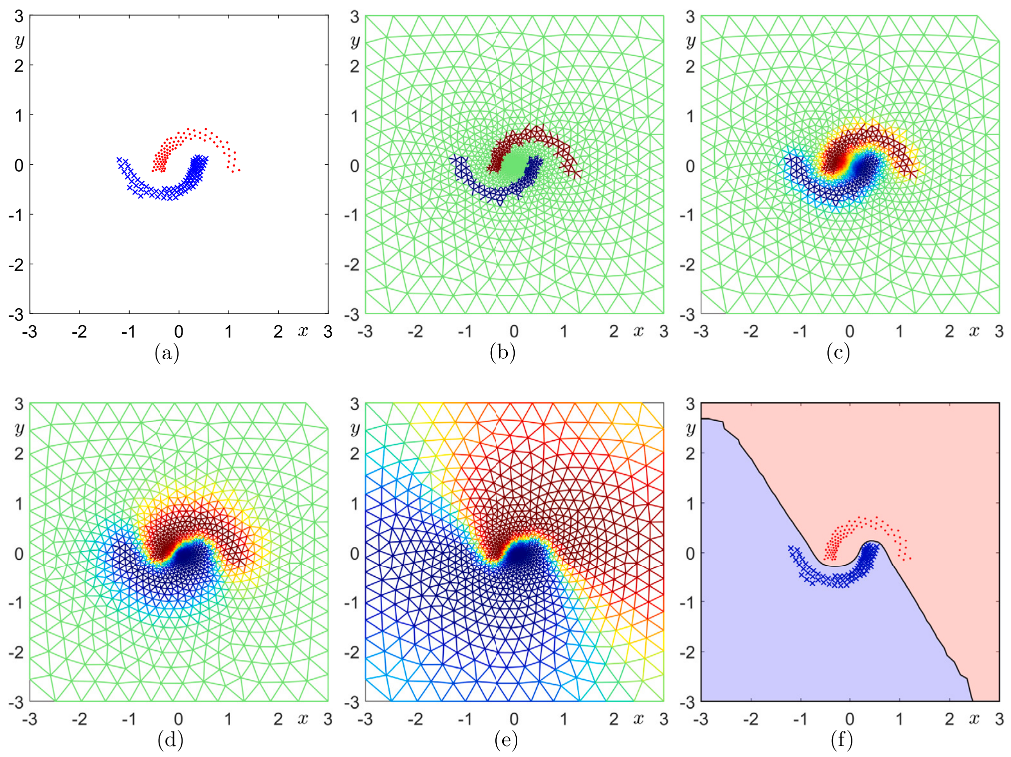

Fig. 11. (a) Given initial data. (b) Fidelity data f_k are given as one half circle takes plus one (red color), the other half circle takes negative one (blue color), the rest of the domain takes zero (green color). (c)–(e) Process of the classification at times $t = 100\Delta t,\ 1000\Delta t,\ 10000\Delta t$, respectively. (f) The zero-level filled contour of classification result. (For interpretation of the colors in the figure(s), the reader is referred to the web version of this article.)

[36–38], clustering [39], random forest [40,41]. Cervantes et al. [33] described many applications in real-world problems: image classification, bioinformatics, and face detection. Behkami et al. [36] proposed artificial neural network for classification of cow milk using the spectral data. In [39], semi-supervised algorithm based on embedded clustering method was proposed for image classification and segmentation. Gu et al. [41] demonstrated the efficiency of their classification method by using real-life data sets from the University of California, Irvine. Let us consider the following modified AC equation for data classification:

$$\frac{\partial \phi(\mathbf{x},t)}{\partial t} = -\frac{F'(\phi(\mathbf{x},t))}{\epsilon^2(\mathbf{x})} + \Delta\phi(\mathbf{x},t) + \lambda(\mathbf{x})(f(\mathbf{x}) - \phi(\mathbf{x},t)),\ \mathbf{x}\in\Omega,\ t>0, \tag{25}$$

where the fidelity term $\lambda(\mathbf{x})(f(\mathbf{x}) - \phi(\mathbf{x},t))$ is added to the proposed method for data classification. We note that if $\epsilon(\mathbf{x})$ is constant, then Eq. (25) becomes a phase-field model for binary data classification based on the AC equation and further details can be found in [42]. To numerically solve Eq. (25) using the operator splitting method, we need one more step in addition to Eqs. (6) and (7):

$$\frac{\partial \phi(\mathbf{x},t)}{\partial t} = \lambda(\mathbf{x})(f(\mathbf{x}) - \phi(\mathbf{x},t)). \tag{26}$$

After numerically solving Eqs. (6) and (7) using the following procedures,

$$\phi_k^* = \phi_k^n + \Delta t \Delta_d \phi_k^n,\quad 1 \le k \le N. \tag{27}$$

$$\phi_k^{**} = \frac{\phi_k^*}{\sqrt{[1-(\phi_k^*)^2]e^{-\frac{2\Delta t}{\epsilon^2(\mathbf{x}_k)}} + (\phi_k^*)^2}}. \tag{28}$$

The fidelity Eq. (26) is solved implicitly as follows:

$$\frac{\phi_k^{n+1} - \phi_k^{**}}{\Delta t} = \lambda_k(f_k - \phi_k^{n+1}),\qquad k = 1,\dots,N, \tag{29}$$

where λ is the fidelity coefficient and f is the given fidelity data. From Eq. (29), we obtain

$$\phi_k^{n+1} = \frac{\phi_k^{**} + \lambda_k \Delta t f_k}{1+\lambda_k\Delta t},\qquad k = 1,\dots,N.$$

For efficient computation in data classification, we consider a nonuniform triangular mesh on $\Omega = (-3,\ 3)\times(-3,\ 3)$, where the mesh structure is finer near the given data and gradually coarser away from the data. The given data is observed in Fig. 11(a). The fidelity data f_k are defined such that one half-circle takes a value of plus one (red color), the other half-circle takes negative one (blue color), and the rest of the domain takes zero (green color) as shown in Fig. 11(b). We set the values of λ_k as follows: $\lambda_k = 1000$ if $f_k \neq 0$; otherwise $\lambda_k = 0$. We use scaling parameter $s = 1.5$ and time step size $\Delta t = 7.5\text{e-}5$. Figs. 11(c)–(e) display the temporal evolution of the classification process at times $t = 100\Delta t,\ 1000\Delta t,\ 10000\Delta t$, respectively. Fig. 11(f) is the zero-level filled contour of the data classification result, as shown in Fig. 11(e). We can observe that the classification efficiently processes across the entire domain over time. Furthermore, we conducted a numerical comparison experiment with a uniform triangular mesh using the smallest length of the nonuniform mesh. Compared to using a fine triangular mesh, the CPU time for the proposed method is two orders of magnitude faster, which demonstrates its superior performance over conventional algorithms.

4. Conclusions

In this study, we proposed a modified AC equation with a mesh size-dependent interfacial parameter on a triangular mesh to efficiently solve multi-scale problems. In the proposed method, a triangular mesh is used, and the interfacial parameter value at a node point is defined as a function of the average length of the edges connected to the node point. The proposed algorithm effectively uses large values of the interfacial parameter at coarse meshes and small values at fine meshes. Computational tests were conducted to validate the efficiency and superior performance of the proposed algorithm. The computational results confirm that our interfacial function adequately evolves multi-scale phase interfaces without inducing excessive relaxation or freezing of the interfaces. This promising approach not only saves the computational costs associated with traditional methods but also provides a robust solution for handling multi-scale interfacial dynamics in various applications. As interesting future research directions, efficient adaptive time-step [43,44], and adapting high-order numerical schemes such as those presented in [45], where up to fourth-order unconditionally structure-preserving parametric single-step methods were presented, extending the model to three-dimensional space and multi-component systems, adding a local-nonlocal space-time dependent Lagrange multiplier to the modified AC equation to make it conservative [46], studying parabolic sine-Gordon equations [47], where multiple states can be modeled, and investigating novel applications in the field could significantly contribute to advancing our proposed model. Furthermore, the proposed method can be extended to achieve second-order accuracy in time and incorporate adaptive time-stepping methods [48].

CRediT authorship contribution statement

Junxiang Yang: Writing – original draft, Software, Resources, Methodology, Investigation, Formal analysis, Data curation, Conceptualization. **Jian Wang:** Writing – original draft, Methodology, Investigation, Formal analysis, Data curation, Conceptualization. **Soobin Kwak:** Writing – review & editing, Writing – original draft, Visualization, Validation, Software, Investigation, Data curation. **Seokjun Ham:** Writing – review & editing, Writing – original draft, Visualization, Validation, Software, Investigation, Data curation. **Junseok Kim:** Writing – review & editing, Writing – original draft, Validation, Supervision, Project administration, Methodology, Investigation, Formal analysis, Conceptualization.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

The software program and source codes are available upon request or can be downloaded from the following link: https://mathematicians.korea.ac.kr/cfdkim/open-source-codes/.

Acknowledgement

J. Yang is supported by Macau University of Science and Technology Faculty Research Grants (FRG) (No. FRG-24-026-FIE). Jian Wang is supported by the Open Project of Center for Applied Mathematics of Jiangsu Province (Nanjing University of Information Science and Technology). 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 gratitude to the referees whose feedback significantly enhanced the quality of the paper.

Appendix A

In this Appendix, we provide the main source code for the methodology so that interested readers can use it. The full codes including mesh generation as described in this paper are available at the corresponding author's repository link: https://mathematicians.korea.ac.kr/cfdkim/open-source-codes/

```
clear all; format long;
Lx=0; Rx=5; Ly=0; Ry=1; h0=0.05; fd=@(p) drectangle(p,Lx,Rx,Ly,Ry
    ); fh=@(p) h0+0.01*p(:,1);
[p,t]=distmesh2d(fd,fh,h0,[Lx,Ly;Rx,Ry],[Lx,Ly;Rx,Ly;Rx,Ry;Lx,Ry
    ]);
% Indexing and make one-rings
fid=fopen(sprintf('vring_data.m'),'wt');
indt=[find(p(:,1)<Lx+h0/2)]; indt=[indt;find(p(:,1)>Rx-h0/2)];
    indt=[indt;find(p(:,2)<Ly+h0/2)];
indt=[indt;find(p(:,2)>Ry-h0/2)]; id=[1:length(p)]; id(indt)=[];
for iter = 1:length(id)
   commonPT=id(iter); IndexN=find(t(:)==commonPT);
   % Ordering neighboring points
   PT = zeros(length(IndexN),2);
   for i=1:length(IndexN)
      if mod(IndexN(i),size(t,1))==0, tmpT = t(size(t,1),:);
      else tmpT = t(mod(IndexN(i),size(t,1)),:); end
      if tmpT(1) == commonPT, PT(i,:) = [tmpT(2) tmpT(3)];
      elseif tmpT(2) == commonPT, PT(i,:) = [tmpT(3) tmpT(1)];
      else PT(i,:) = [tmpT(1) tmpT(2)]; end
   end
   V=[]; V(1:2)=PT(1,:); PT(1,:)=[];
   for k=1:length(IndexN)-1, a=find(PT(:,1)==V(end)); V(end+1)=
       PT(a,2); PT(a,:)=[]; end
   V(end+1:8)=0; fprintf(fid,'%d %d %d %d %d %d %d %d %d \n',
       length(IndexN),V);
end
fclose(fid); D = load('vring_data.m'); if max(D(:,1))>7, error
    happen, end
% Setting of the initial conditions
for i=1:length(p), if p(i,1)<0.2, valP(i)=1.0; else valP(i)=0;
    end; end
nnLap=0*valP;
% For boundary condition
bdid=unique([find(abs(p(:,1)-Lx)<0.5*h0 | abs(p(:,1)-Rx)<0.5*h0 |
     abs(p(:,2)-Ly)<0.5*h0...
   | abs(p(:,2)-Ry)<0.5*h0)]);
for i=1:length(bdid)
   temp=[];
   for j=1:length(t), if sum(bdid(i)==t(j,:))>0, temp=[temp,t(j
       ,:)]; end; end
   temp=unique(temp); nb_in_bd{i,:}=[bdid(i),temp];
   for k=1:length(bdid), temp(temp==bdid(k))=[]; end
   nb_bd{i,:}=[bdid(i),temp];
end
for i=1:length(bdid), valP(bdid(i))=mean(valP(nb_bd{i}(2:end)));
    end
nvalP=valP;
% Space-dependent epsilon
deps=valP; deps(bdid)=nan;
for k=1:length(D)
   dd=0;
   for i=2:D(k,1)+1
     x0=p(id(k),1); y0=p(id(k),2); x=p(D(k,i),1); y=p(D(k,i),2)
         ; dd=dd+sqrt((x0-x)^2+(y0-y)^2);
   end
   epsilon(k)=0.5*dd/(D(k,1));
end
deps(id)=epsilon;
% stable time-step
alpA=zeros(size(D,1),size(D,2)); betA=alpA;
for k=1:length(D)
   Egrad=0; Allarea=0; triC=id(k);
   for i=2:D(k,1)+1
      if i==2, triL=D(k,D(k,1)+1); triM=D(k,i); triN=D(k,i+1);
      else triL=D(k,i-1); triM=D(k,i); triN=D(k,i+1); end
      Ph1=p(triC,:)-p(triL,:); Ph2=p(triM,:)-p(triL,:); Pi1=p(
          triC,:)-p(triN,:);
      Pi2=p(triM,:)-p(triN,:); Pj1=p(triM,:)-p(triC,:); Pj2=p(
          triN,:)-p(triC,:);
      alpA(k,i-1)=acos(dot(Ph1,Ph2)/(norm(Ph1)*norm(Ph2)));
      betA(k,i-1)=acos(dot(Pi1,Pi2)/(norm(Pi1)*norm(Pi2)));
      Egrad=Egrad+(cot(alpA(k,i-1))+cot(betA(k,i-1)));
      Earea=0.5*sqrt(norm(Pj1)^2*norm(Pj2)^2-dot(Pj1,Pj2)^2);
          Allarea=Allarea+Earea;
   end
   dtt(k)=2*Allarea/(3*Egrad);
end
dt=min(dtt);
% Main algorithm
err=1; iter=0;
while err>1e-6
   iter=iter+1;
   for k=1:length(D)
```

```
        Egrad=0; Eachgrad1=0; Allarea=0; sumalphaA=0; triC=id(k);
        for i=2:D(k,1)+1
            if i==2, triL=D(k,D(k,1)+1); triM=D(k,i); triN=D(k,i
                +1);
            else triL=D(k,i-1); triM=D(k,i); triN=D(k,i+1); end
            Ph1=p(triC,:)-p(triL,:); Ph2=p(triM,:)-p(triL,:); Pi1
                =p(triC,:)-p(triN,:);
            Pi2=p(triM,:)-p(triN,:); Pj1=p(triM,:)-p(triC,:); Pj2
                =p(triN,:)-p(triC,:);
            alpA=acos(dot(Ph1,Ph2)/(norm(Ph1)*norm(Ph2)));
            betA=acos(dot(Pi1,Pi2)/(norm(Pi1)*norm(Pi2)));
            Egrad=Egrad+0.5*(cot(alpA)+cot(betA))*(valP(triM)-
                valP(triC));
            Earea=0.5*sqrt(norm(Pj1)^2*norm(Pj2)^2-dot(Pj1,Pj2)
                ^2); Allarea=Allarea+Earea;
        end
        nLap(k) = valP(triC) + dt*3.0*Egrad/Allarea;
    end
    nvalP(id)=nLap./sqrt(exp(-2*dt./(epsilon.^2))+nLap.^2.*(1-exp
        (-2*dt./(epsilon.^2))));
    % update boundary value
    for bdi=1:length(bdid), nvalP(bdid(bdi))=mean(nvalP(nb_bd{bdi
        }(2:end))); end
    err=sqrt(sum((nvalP(id)-valP(id)).^2)/length(id)); valP=nvalP
        ;
    if (iter==1 || mod(iter,10)==0)
        fig=figure(1); clf; hold on; view(5,30); box on; grid on;
             fig.Position(3:4)=[1000,250];
        set(gca,'FontSize',13); colormap jet; clim([0,1]);
            trimesh(t,p(:,1),p(:,2),nvalP);
        text([4.5,5.2,-0.3],[-0.05,0.8,0],[-0.15,-0.15,0.75],{'$x
            $','$y$','$\phi$'},...
            'Interpreter','Latex','FontSize',15)
        zticks([0,0.5,1]); xticks(0:5); axis([Lx,Rx,Ly,Ry,0,1]);
            drawnow
    end
end
```

References

[1] J. Park, C. Lee, Y. Choi, H. Lee, S. Kwak, Y. Hwang, J. Kim, An unconditionally stable splitting method for the Allen–Cahn equation with logarithmic free energy, J. Eng. Math. 132 (18) (2022) 1–18.

[2] S.M. Allen, J.W. Cahn, A microscopic theory for antiphase boundary motion and its application to antiphase domain coarsening, Acta Metall. 27 (6) (1979) 1085–1095.

[3] Y. Kim, G. Ryu, Y. Choi, Fast and accurate numerical solution of Allen–Cahn equation, Math. Probl. Eng. 2021 (2021) 5263989.

[4] B. Inan, M.S. Osman, T. Ak, D. Baleanu, Analytical and numerical solutions of mathematical biology models: the Newell–Whitehead–Segel and Allen–Cahn equations, Math. Methods Appl. Sci. 43 (5) (2020) 2588–2600.

[5] Y. Li, Q. Xia, S. Yoon, C. Lee, B. Lu, J. Kim, Simple and efficient volume merging method for triply periodic minimal structures, Comput. Phys. Commun. 264 (2021) 107956.

[6] J.L. Fattebert, S. DeWitt, A. Perron, J. Turner, Thermo4PFM: facilitating phase-field simulations of alloys with thermodynamic driving forces, Comput. Phys. Commun. 288 (2023) 108739.

[7] Z. Wu, H. Ye, H. Zhang, Y. Zheng, Seq-SVF: an unsupervised data-driven method for automatically identifying hidden governing equations, Comput. Phys. Commun. 292 (2023) 108887.

[8] S. Koohy, G. Yao, K. Rubasinghe, Numerical solutions to low and high-dimensional Allen–Cahn equations using stochastic differential equations and neural networks, Partial Differ. Equ. Appl. Math. 7 (2023) 100499.

[9] D. Jeong, Y. Li, Y. Choi, C. Lee, J. Yang, J. Kim, A practical adaptive grid method for the Allen–Cahn equation, Physica A 573 (2021) 125975.

[10] Y. Liu, X. Shen, Z. Guan, Y. Nie, The adaptive SAV weak Galerkin finite element method for the Allen–Cahn equation, Comput. Math. Appl. 151 (2023) 449–460.

[11] Y. Chen, Y. Huang, N. Yi, A SCR-based error estimation and adaptive finite element method for the Allen–Cahn equation, Comput. Math. Appl. 78 (1) (2019) 204–223.

[12] V. Joshi, R.K. Jaiman, An adaptive variational procedure for the conservative and positivity preserving Allen–Cahn phase-field model, J. Comput. Phys. 366 (2018) 478–504.

[13] J. Li, J. Zeng, R. Li, An adaptive discontinuous finite volume element method for the Allen–Cahn equation, Adv. Comput. Math. 49 (4) (2023) 55.

[14] K. Poochinapan, B. Wongsaijai, Numerical analysis for solving Allen–Cahn equation in 1D and 2D based on higher-order compact structure-preserving difference scheme, Appl. Math. Comput. 434 (2022) 127374.

[15] Y. Hwang, S. Ham, C. Lee, G. Lee, S. Kang, J. Kim, A simple and efficient numerical method for the Allen–Cahn equation on effective symmetric triangular meshes, Electron. Res. Arch. 31 (8) (2023) 4557–4578.

[16] E. Celiker, P. Lin, An efficient finite element method with exponential mesh refinement for the solution of the Allen–Cahn equation in non-convex polygons, Commun. Comput. Phys. 28 (4) (2020) 1536–1560.

[17] D. Jeong, J. Kim, Explicit hybrid finite difference scheme for the Allen–Cahn equation, J. Comput. Appl. Math. 340 (2018) 247–255.

[18] X. Xiao, X. Feng, A second-order maximum bound principle preserving operator splitting method for the Allen–Cahn equation with applications in multi-phase systems, Math. Comput. Simul. 202 (2022) 36–58.

[19] R. Li, Y. Gao, Z. Chen, Adaptive discontinuous Galerkin finite element methods for the Allen–Cahn equation on polygonal meshes, Numer. Algorithms (2023) 1–34.

[20] X. Yang, Error analysis of stabilized semi-implicit method of Allen–Cahn equation, Discrete Contin. Dyn. Syst., Ser. B 11 (4) (2009) 1057–1070.

[21] H.G. Lee, A new conservative Swift–Hohenberg equation and its mass conservative method, J. Comput. Appl. Math. 375 (2020) 112815.

[22] Y. Li, S. Lan, X. Liu, B. Lu, L. Wang, An efficient volume repairing method by using a modified Allen–Cahn equation, Pattern Recognit. 107 (2020) 107478.

[23] P.O. Persson, G. Strang, A simple mesh generator in MATLAB, SIAM Rev. 46 (2) (2004) 329–345.

[24] G. Xu, Convergence of discrete Laplace–Beltrami operators over surfaces, Comput. Math. Appl. 48 (2004) 347–360.

[25] H. Zhang, J. Yan, X. Qian, S. Song, Numerical analysis and applications of explicit high order maximum principle preserving integrating factor Runge–Kutta schemes for Allen–Cahn equation, Appl. Numer. Math. 161 (2021) 372–390.

[26] T. Hou, H. Leng, Numerical analysis of a stabilized Crank–Nicolson/Adams–Bashforth finite difference scheme for Allen–Cahn equations, Appl. Math. Lett. 102 (2020) 106150.

[27] 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.

[28] J. Shin, S.K. Park, J. Kim, A hybrid FEM for solving the Allen–Cahn equation, Appl. Math. Comput. 244 (2014) 606–612.

[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] H. Zhang, X. Qian, J. Xia, S. Song, Efficient inequality-preserving integrators for differential equations satisfying forward Euler conditions, ESAIM: Math. Model. Numer. Anal. 57 (3) (2023) 1619–1655.

[31] Z. Tan, L. Chen, J. Yang, Generalized Allen–Cahn-type phase-field crystal model with FCC ordering structure and its conservative high-order accurate algorithm, Comput. Phys. Commun. 286 (2023) 108656.

[32] C. Lee, J. Park, S. Kwak, S. Kim, Y. Choi, S. Ham, J. Kim, An adaptive time-stepping algorithm for the Allen–Cahn equation, J. Funct. Spaces 2022 (2022) 2731593.

[33] J. Cervantes, F. Garcia-Lamont, L. Rodríguez-Mazahua, A. Lopez, A comprehensive survey on support vector machine classification: applications, challenges and trends, Neurocomputing 408 (2020) 189–215.

[34] C. Jimenez-Castano, A. Alvarez-Meza, A. Orozco-Gutierrez, Enhanced automatic twin support vector machine for imbalanced data classification, Pattern Recognit. 107 (2020) 107442.

[35] W. Dudzik, J. Nalepa, M. Kawulok, Evolving data-adaptive support vector machines for binary classification, Knowl.-Based Syst. 227 (2021) 107221.

[36] S. Behkami, S.M. Zain, M. Gholami, M.F.A. Khir, Classification of cow milk using artificial neural network developed from the spectral data of single- and three-detector spectrophotometers, Food Chem. 294 (2019) 309–315.

[37] G. Amato, L. Palombi, V. Raimondi, Data–driven classification of landslide types at a national scale by using Artificial Neural Networks, Int. J. Appl. Earth Obs. Geoinf. 104 (2021) 102549.

[38] A. Mittal, A. Soorya, P. Nagrath, D.J. Hemanth, Data augmentation based morphological classification of galaxies using deep convolutional neural network, Earth Sci. Inform. 13 (2020) 601–617.

[39] J. Enguehard, P. ÓHalloran, A. Gholipour, Semi-supervised learning with deep embedded clustering for image classification and segmentation, IEEE Access 7 (2019) 11093–11104.

[40] Z. Qu, H. Li, Y. Wang, J. Zhang, A. Abu-Siada, Y. Yao, Detection of electricity theft behavior based on improved synthetic minority oversampling technique and random forest classifier, Energies 13 (8) (2020) 2039.

[41] Q. Gu, J. Tian, X. Li, S. Jiang, A novel Random Forest integrated model for imbalanced data classification problem, Knowl.-Based Syst. 250 (2022) 109050.

[42] S. Kim, J. Kim, Automatic binary data classification using a modified Allen–Cahn equation, Int. J. Pattern Recognit. Artif. Intell. 35 (04) (2021) 2150013.

[43] H.L. Liao, B. Ji, L. Zhang, An adaptive BDF2 implicit time-stepping method for the phase field crystal model, IMA J. Numer. Anal. 42 (1) (2022) 649–679.

[44] H.L. Liao, B. Ji, L. Wang, Z. Zhang, Mesh-robustness of an energy stable BDF2 scheme with variable steps for the Cahn–Hilliard model, J. Sci. Comput. 92 (2) (2022) 52.

[45] H. Zhang, J. Yan, X. Qian, S. Song, Up to fourth-order unconditionally structure-preserving parametric single-step methods for semilinear parabolic equations, Comput. Methods Appl. Mech. Eng. 393 (2022) 114817.

[46] H. Zhang, J. Yan, X. Qian, X. Chen, S. Song, Explicit third-order unconditionally structure-preserving schemes for conservative Allen–Cahn equations, J. Sci. Comput. 90 (2022) 1–29.

[47] H. Zhang, X. Qian, J. Xia, S. Song, Unconditionally maximum-principle-preserving parametric integrating factor two-step Runge–Kutta schemes for parabolic Sine-Gordon equations, CSIAM Trans. Appl. Math. 4 (1) (2023) 177–224.

[48] 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.