Mathematics

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

AIMS Mathematics, 10(3): 7319–7338.
DOI: 10.3934/math.2025335
Received: 31 December 2024
Revised: 06 March 2025
Accepted: 26 March 2025
Published: 28 March 2025

Research article

Benchmark problems for physics-informed neural networks: The Allen–Cahn equation

Hyun Geun Lee[1], **Youngjin Hwang**[2], **Yunjae Nam**[3], **Sangkwon Kim**[2] **and Junseok Kim**[2,*]

[1] Department of Mathematics, Dongguk University, Seoul 04620, Republic of Korea
[2] Department of Mathematics, Korea University, Seoul 02841, Republic of Korea
[3] Program in Actuarial Science and Financial Engineering, Korea University, Seoul 02841, Republic of Korea

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

Abstract: In this paper, we present accurate and well-designed benchmark problems for evaluating the effectiveness and precision of physics-informed neural networks (PINNs). The presented problems were generated using the Allen–Cahn (AC) equation, which models the mechanism of phase separation in binary alloy systems and simulates the temporal evolution of interfaces. The AC equation possesses the property of motion by mean curvature, which means that, in the sharp interface limit, the evolution of the interface described by the equation is governed by its mean curvature. Specifically, the velocity of the interface is proportional to its mean curvature, which implies the tendency of the interface to minimize its surface area. This property makes the AC equation a powerful mathematical model for capturing the dynamics of interface motion and phase separation processes in various physical and biological systems. The benchmark source codes for the 1D, 2D, and 3D AC equations are provided for interested researchers.

Keywords: benchmark problems; Allen–Cahn equation; physics-informed neural networks
Mathematics Subject Classification: 35K57, 65M22, 82C26

1. Introduction

We present accurate and well-designed benchmark problems for evaluating the effectiveness and precision of physics-informed neural networks (PINNs). The presented problems are generated using the Allen–Cahn (AC) equation [1]:

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

$$= -\frac{\phi^3(\mathbf{x},t) - \phi(\mathbf{x},t)}{\epsilon^2} + \Delta\phi(\mathbf{x},t), \quad \mathbf{x} \in \Omega, \; t > 0. \tag{1.2}$$

where $\phi(\mathbf{x},t)$ is a phase-field function, $F(\phi) = 0.25\left(\phi^2 - 1\right)^2$ is potential energy (see Figure 1), and ϵ is a positive parameter. Here, we use the zero Neumann boundary condition.

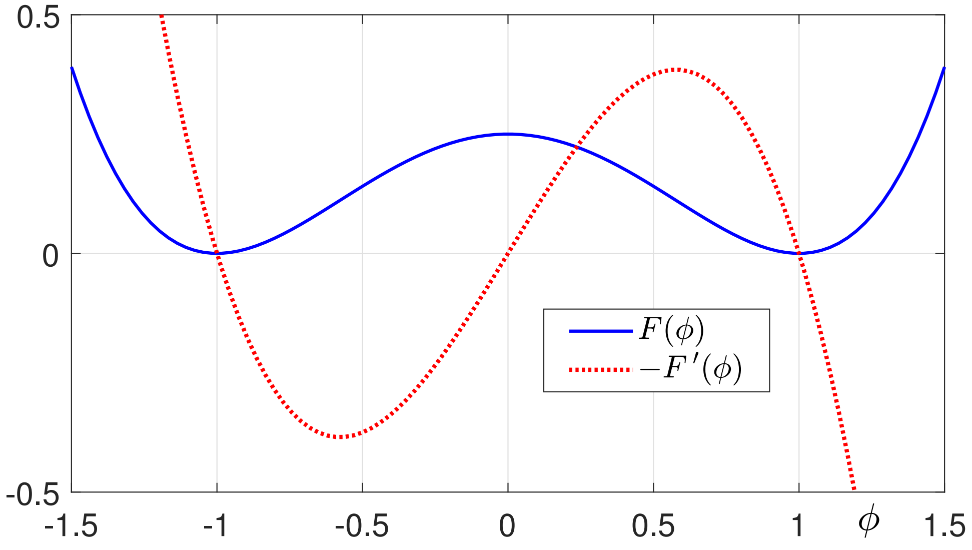


Figure 1. Polynomial double-well potential energy and its first derivative.

The AC equation can be derived as the L^2-gradient flow of the associated free energy functional, which is defined as follows:

$$\mathcal{E}(\phi) = \int_\Omega \left(\frac{F(\phi)}{\epsilon^2} + \frac{1}{2}|\nabla\phi|^2 \right) d\mathbf{x}. \tag{1.3}$$

The AC equation, a fundamental equation in the phase-field model [2], is extensively adopted in modeling phase transitions and interface dynamics across various disciplines. In material science, it describes phase separation in binary mixtures, grain boundary evolution in polycrystals, and solidification processes in crystal growth [3,4]. In image processing, it aids in image segmentation [5] by detecting edges and performs image inpainting by evolving interfaces to fill missing regions. In biology, it models morphogenesis and cellular membrane dynamics, while in fluid dynamics, it captures two-phase flows [6,7], such as oil-water interfaces, and bubble dynamics. In medical signal analysis, the AC energy function is used to extract nonlinear features from electroencephalogram data. Lu and Wang [8] accomplished high accuracy in epilepsy classification through AC support vector machine and AC multi-complexity support vector machine classifiers and incorporated measures like Shannon entropy, which offered excellent performance, reduced computational costs, and significant potential for medical applications. Chemical engineering applications include catalysis interface modeling and reaction-diffusion systems, where it explains chemical pattern formation. In physics, AC simulates quantum dot behavior and magnetic flux distribution in superconductors [9]. Geophysics uses it for fracture propagation in brittle materials and landslide modeling [10]. The equation also serves as a benchmark in numerical analysis, testing adaptive time-stepping methods and energy-preserving numerical schemes [11,12]. Further applications include clustering and feature extraction in AI, modeling minimal surfaces in geometry, and supporting studies on curvature-driven motion. Its versatility lies in modeling dynamics where phase boundaries and interface motion are key phenomena, which makes it invaluable across scientific and engineering domains.

Raissi et al. [13] presented physics-informed neural networks (PINNs) to incorporate physical laws into the training process for solving differential equations, demonstrating success with Burgers' and Schrödinger equations. However, the method encounters difficulties with phase-field equations, as it relies on static, randomly selected training points that cannot adapt to the dynamic sharp transition layers characteristic of these models. Recent advancements have focused on solving these challenges through innovations in sampling and training strategies. In [14], the authors proposed a spatio-temporal adaptive PINN for the AC and Cahn–Hilliard equations [15–17]. The key algorithm employs adaptive temporal sampling, which dynamically adjusts training points to regions with high error. By reformulating the problem as a second-order equation and using adaptive sampling based on energy evolution, this method reduces computational costs and improves accuracy. Techniques such as parareal PINN (PPINN) [18] and other domain decomposition approaches [19] enhance computational efficiency by dividing the problem into smaller subdomains, while conservative PINNs (cPINNs) [19] ensure physical continuity across these subdomains. To further improve accuracy, constraints such as mass preservation and energy dissipation have been incorporated into the loss function, ensuring that solutions adhere to physical laws. Residual-based adaptive refinement (RAR) [20, 21] dynamically allocates training points to regions with large residuals, improving accuracy in critical areas.

Roy and Castonguay [22] introduced benchmark problems to validate the effectiveness of PINNs in solving complex time-dependent partial differential equations. Monaco and Apiletti [23] investigated the performance of state-of-the-art PINNs training methods across various benchmark problems, including the AC equation. They highlighted their limitations and proposed enhanced strategies for improved convergence and accuracy. Huang and Alkhalifah [24] proposed an efficient PINNs framework by integrating multi-resolution hash encoding that demonstrates up to a 10-fold acceleration in training speed for benchmark problems such as the AC equation. McClenny and Braga-Neto [25] introduced self-adaptive PINNs (SA-PINNs), which improve the training process of PINNs by dynamically weighting individual training points, demonstrating superior performance on various benchmark problems such as the AC equation. Kopaničáková et al. [26] introduced nonlinear additive and multiplicative preconditioning strategies for the limited-memory Broyden–Fletcher–Goldfarb–Shanno (L-BFGS) optimizer to enhance the training of PINNs. The authors validated the proposed method on benchmark problems such as the Klein–Gordon and AC equations.

Qiu et al. [27] applied PINNs to the phase-field model of two-phase flow based on the Navier–Stokes and Cahn–Hilliard equations. To resolve the significant challenges posed by the high-order nonlinear differential terms in these equations, the authors used adaptive techniques such as the time-marching strategy, which allows PINNs to preserve high accuracy while reducing computational costs. Ghaffari Motlagh et al. [28] proposed a deep learning phase-field model for brittle fracture problems using PINNs. The authors compared this model with various methods such as traditional and variational PINNs. The developed model effectively captures fracture initiation and propagation with better computational efficiency. However, it has a significant issue of being highly sensitive to parameter choices within the neural network. Li et al. [29] developed phase-field DeepONet to overcome the computational cost issue in simulating real pattern-forming systems. By incorporating the minimizing movement scheme, the authors constructed a physics-informed deep operator neural network framework that predicts the dynamic responses of systems governed by the gradient flows of free-energy functionals without directly solving the governing equations. The authors validated the proposed method using reactive phase-field models, specifically the AC and Cahn–Hilliard equations.

The remainder of this paper is organized as follows: Section 2 introduces a series of benchmark problems for the AC equation. It begins with the conventional benchmark problem in one-dimensional (1D) space, followed by another benchmark problem of the traveling wave solution in one-dimensional space. Subsequently, the motion by mean curvature is presented in two-dimensional (2D) and three-dimensional (3D) spaces, which is an important feature of the AC equation. Finally, Section 3 presents the conclusions, which include a summary of the findings as well as a discussion of potential applications and future directions.

2. Benchmark problems

To provide a straightforward explanation, we focus on the AC equation in the 1D domain $\Omega = (L_x, R_x)$. The discretization approach can be extended similarly to 2D and 3D spaces. Let $h = (R_x - L_x)/N$ be the grid size with an integer N and $\phi_i^n = \phi(x_i, n\Delta t)$, where $x_i = L_x + (i - 0.5)h$ and Δt is the time step size. The AC equation (1.1) can be discretized as follows:

$$\begin{aligned} \frac{\phi_i^{n+1} - \phi_i^n}{\Delta t} + O(\Delta t) &= -\frac{(\phi_i^n)^3 - \phi_i^n}{\epsilon^2} + \Delta_h \phi_i^n + O(h^2) \\ &= \frac{\phi_i^n - (\phi_i^n)^3}{\epsilon^2} + \frac{\phi_{i-1}^n - 2\phi_i^n + \phi_{i+1}^n}{h^2} + O(h^2), \end{aligned} \tag{2.1}$$

which is approximated as

$$\frac{\phi_i^{n+1} - \phi_i^n}{\Delta t} = \frac{\phi_i^n - (\phi_i^n)^3}{\epsilon^2} + \frac{\phi_{i-1}^n - 2\phi_i^n + \phi_{i+1}^n}{h^2}, \tag{2.2}$$

which can be rewritten explicitly:

$$\phi_i^{n+1} = \phi_i^n + \Delta t \left(\frac{\phi_i^n - (\phi_i^n)^3}{\epsilon^2} + \frac{\phi_{i-1}^n - 2\phi_i^n + \phi_{i+1}^n}{h^2} \right). \tag{2.3}$$

The 2D and 3D numerical schemes for the AC equations are presented as follows, respectively:

$$\phi_{ij}^{n+1} = \phi_{ij}^n + \Delta t \left(\frac{\phi_{ij}^n - (\phi_{ij}^n)^3}{\epsilon^2} + \frac{\phi_{i-1,j}^n + \phi_{i+1,j}^n + \phi_{i,j-1}^n + \phi_{i,j+1}^n - 4\phi_{ij}^n}{h^2} \right), \tag{2.4}$$

$$\begin{aligned} \phi_{ijk}^{n+1} = \phi_{ijk}^n + \Delta t \Bigg(&\frac{\phi_{ijk}^n - (\phi_{ijk}^n)^3}{\epsilon^2} \\ &+ \frac{\phi_{i-1,jk}^n + \phi_{i+1,jk}^n + \phi_{i,j-1,k}^n + \phi_{i,j+1,k}^n + \phi_{ij,k-1}^n + \phi_{ij,k+1}^n - 6\phi_{ijk}^n}{h^2} \Bigg), \end{aligned} \tag{2.5}$$

where $\phi_{ij}^n = \phi(x_i, y_j, n\Delta t)$ and $\phi_{ijk}^n = \phi(x_i, y_j, z_k, n\Delta t)$. The numerical schemes for the AC equation in 1D, 2D, and 3D are stable if the time step size Δt satisfies the following inequality [30].

$$\Delta t \le \frac{\epsilon^2 h^2}{2h^2 + 2d\epsilon^2},$$

where $d = 1, 2,$ and 3 correspond to the respective dimensions. For a detailed stability and convergence analysis of the 1D, 2D, and 3D explicit schemes for the AC equation, refer to [30].

2.1. *Conventional benchmark problem in 1D space*

In [31], the authors considered a benchmark problem with different coefficient scales in the AC equation:

$$\frac{\partial\phi(x,t)}{\partial t} = -5[\phi^3(x,t) - \phi(x,t)] + 0.0001\frac{\partial^2\phi(x,t)}{\partial x^2}, \quad x \in [-1,1],\ t > 0 \tag{2.6}$$

with periodic boundary conditions

$$\phi(-1,t) = \phi(1,t),\ \phi_x(-1,t) = \phi_x(1,t) \tag{2.7}$$

and an initial condition

$$\phi(x,0) = x^2\cos(\pi x),\ x \in [-1,\ 1]. \tag{2.8}$$

Figure 2(a) illustrates the dynamic evolution of the numerical solution $\phi(x,t)$ to the AC equation over time and space under the given initial and boundary conditions. Figure 2(b) depicts the initial condition $\phi(x,0)$, defined as Eq (2.8), which serves as the starting profile for the AC equation. Figure 2(c) presents the solution $\phi(x,0.5)$.

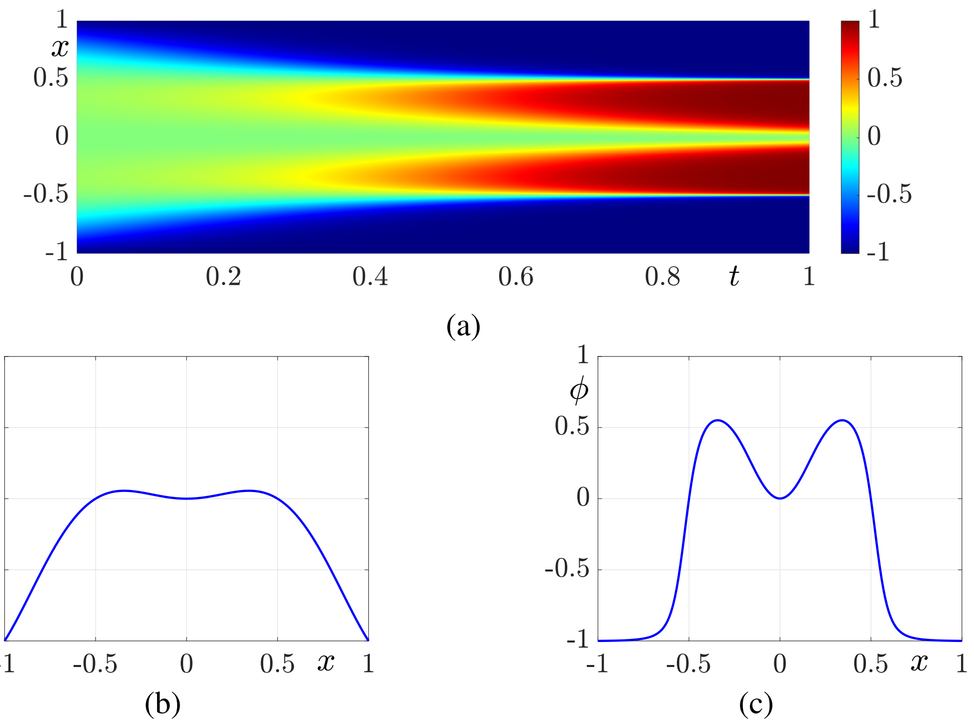


Figure 2. (a) Solutions of $\phi(x,t)$. (b) Initial condition Eq (2.8). (c) Solution of the AC equation at $t = 0.5$.

In addition, a second initial condition was considered:

$$\phi(x,0) = a\left(x^2 + \frac{1}{a}\right)\left(1 - x^2\right)^2 - 1. \tag{2.9}$$

Figure 3(a) demonstrates how the solution to the AC equation evolves dynamically across time and space under the specified initial and boundary conditions. The initial condition $\phi(x,0)$ defined by Eq (2.9) with $a = 3$ is shown in Figure 3(b) as the starting profile for the AC equation. The solution at $t = 0.5$, $\phi(x,0.5)$ is displayed in Figure 3(c).

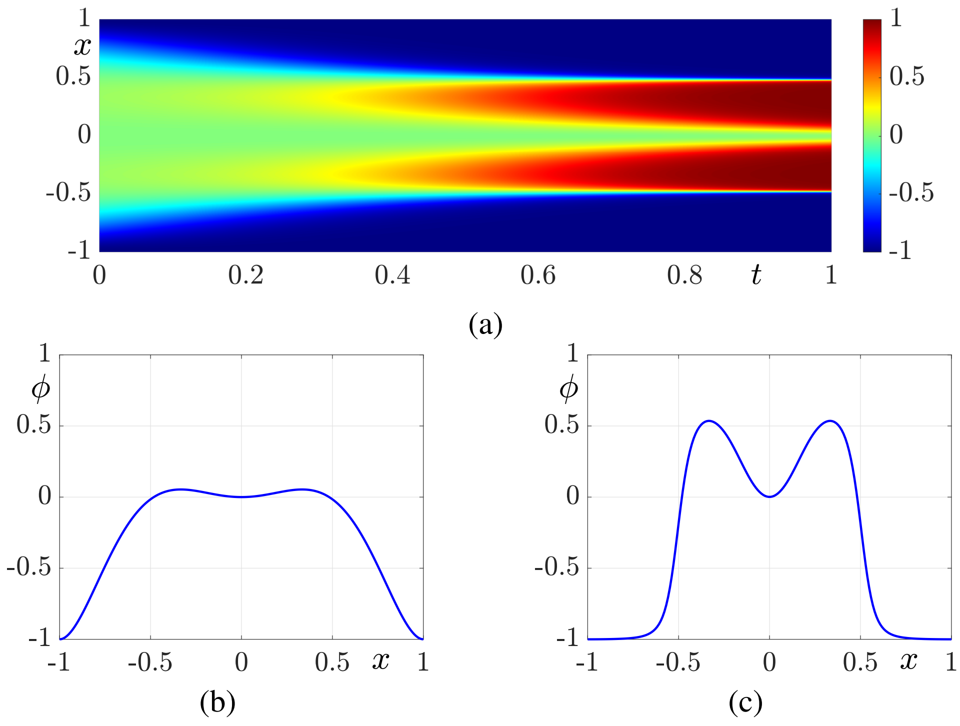


Figure 3. (a) Solutions of $\phi(x,t)$. (b) Initial condition Eq (2.9). (c) Solution of the AC equation at $t = 0.5$.

First, the benchmark problem 1 is solved using the standard-PINN. A standard-PINN is trained to minimize the residual of the given partial differential equation by iteratively updating the neural network parameters. Instead of directly approximating the solution, the neural network learns to reduce the residual error at various spatio-temporal points and ensures that the solution satisfies the underlying physical constraints. The neural network's parameters are initially assigned random values and are subsequently optimized through iterative updates to minimize the cost function, which incorporates the constraints imposed by the PDE. Denote the neural network's output as $\hat{\phi}(x,t)$. The standard-PINN's total cost function is given below:

$$\mathrm{MSE} = \alpha \mathrm{MSE}_i + \mathrm{MSE}_b + \mathrm{MSE}_r, \tag{2.10}$$

where

$$\mathrm{MSE}_i = \frac{1}{N_i}\sum_{k=1}^{N_i}\left(\hat{\phi}\left(x_k^i, 0\right) - \phi_k^i\right)^2, \tag{2.11}$$

$$\text{MSE}_b = \frac{1}{N_b}\sum_{k=1}^{N_b}\left(\hat{\phi}\left(-1,t_k^b\right)-\hat{\phi}\left(1,t_k^b\right)\right)^2+\frac{1}{N_b}\sum_{k=1}^{N_b}\left(\hat{\phi}_x\left(-1,t_k^b\right)-\hat{\phi}_x\left(1,t_k^b\right)\right)^2, \tag{2.12}$$

$$\text{MSE}_r = \frac{1}{N_r}\sum_{k=1}^{N_r}\left(R\left(x_k^r,t_k^r\right)\right)^2. \tag{2.13}$$

Here, $R(x,t) = \phi_t(x,t)+5[\phi^3(x,t)-\phi(x,t)]-0.0001\phi_{xx}(x,t)$ is the residual function for the AC equation and (x_k^i, t_k^i), (x_k^b, t_k^b), and (x_k^r, t_k^r) are the given initial, boundary, and residual data, respectively. These points were sampled according to a uniform distribution. To force initial condition learning, we applied a factor $\alpha = 20$ to the initial loss within the total loss.

To train the standard-PINN structure for the AC equation, we used the number of training data points: $N_i = 512$ points for the initial condition, $N_b = 200$ points for the boundary condition, and $N_r = 10,000$ points for the residual function. The neural network structure features four hidden layers, each containing 128 neurons. Both the adaptive moment estimation (ADAM) and L-BFGS optimizers are utilized during training. Initially, the model is trained using 50,000 iterations of the ADAM optimizer, followed by further training with the L-BFGS optimizer until one of the stopping criteria is satisfied.

Figure 4(a) shows the predicted solution using the standard-PINN for the AC equation and the dynamic evolution of the solution over time and space under the given initial condition Eq (2.9). Figure 4(b)–4(f) displays the predicted and numerical solutions $\hat{\phi}(x,t)$ at various times. Figure 4(g) displays the minimization of the loss function Eq (2.10) while training the standard-PINN for the AC equation.

As shown in Figure 4, we can see that the predicted solution is not close to the numerical solution of benchmark problem 1.

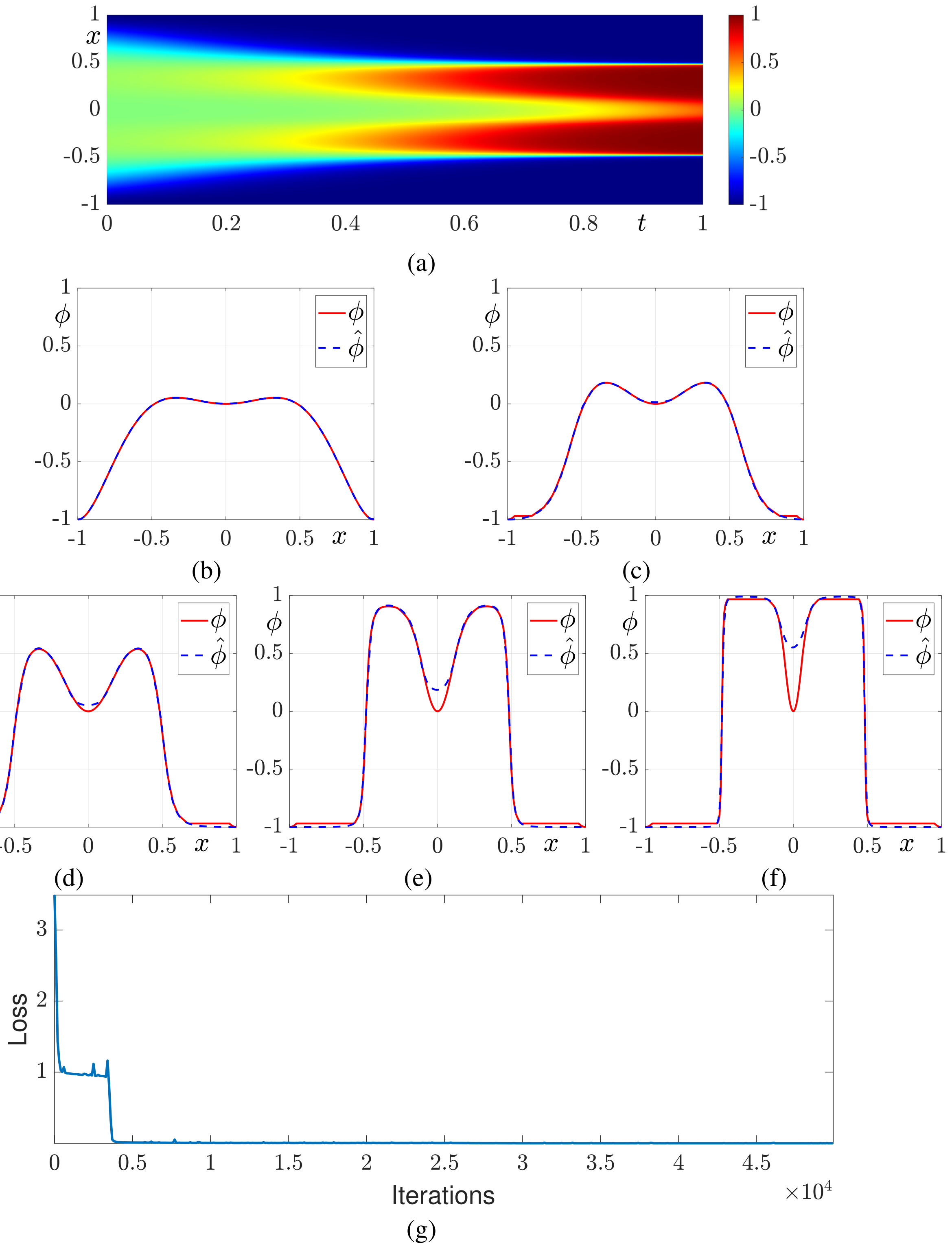


Figure 4. (a) Solutions $\hat{\phi}(x, t)$ obtained using the standard-PINN to learn the AC equation. From (b) to (f), the predicted solution $\hat{\phi}(x, t)$ and numerical solution $\phi(x, t)$ are at $t = 0,\ 0.25,\ 0.5,\ 0.75$, and 1.0. (g) Loss using ADAM optimizer.

2.2. Travelling wave solution in 1D space

The next benchmark problem is the traveling wave solution [30]:

$$\phi_{\text{exact}}(x,t) = \frac{1}{2}\left[1 - \tanh\left(\frac{x - 3t/(\sqrt{2}\epsilon)}{2\sqrt{2}\epsilon}\right)\right], \tag{2.14}$$

which is one of the analytic solutions of the following one-dimensional AC equation [1] on an infinite domain:

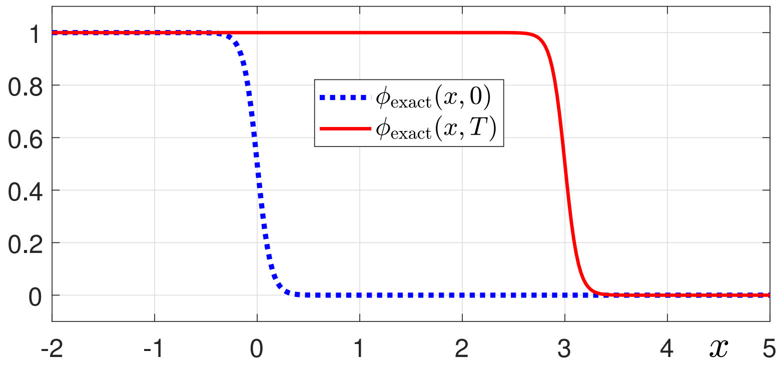
$$\frac{\partial\phi(x,t)}{\partial t} = -\frac{F'(\phi(x,t))}{\epsilon^2} + \frac{\partial^2\phi(x,t)}{\partial x^2}, \;\; x \in \Omega = (-\infty,\ \infty),\; t > 0. \tag{2.15}$$

Figure 5(a) shows Eq (2.14) at times $t = 0$ and $t = T$. Here, $T = \sqrt{2}\epsilon$ and $\epsilon = 2h$. It is straightforward to verify that Eq (2.14) satisfies Eq (2.15) by substituting it in the AC equation. We note that this exact solution is only valid on an infinite domain without boundary conditions. In the numerical methods, a truncated finite domain, $\Omega = (L_x,\ R_y)$, is used instead of an infinite domain, and a homogeneous Neumann boundary condition, $\partial\phi(L_x,t)/\partial x = \partial\phi(R_x,t)/\partial x = 0$, is applied. The rationale behind this homogeneous Neumann boundary condition for the traveling wave benchmark problem is as follows: The first derivative of Eq (2.14) is given as follows:

$$\frac{\partial\phi_{\text{exact}}}{\partial x}(x,t) = -\frac{1}{4\sqrt{2}\epsilon}\text{sech}^2\left(\frac{x - 3t/(\sqrt{2}\epsilon)}{2\sqrt{2}\epsilon}\right), \tag{2.16}$$

which is strictly negative for all x values. However, if the transition layer is away from the domain boundary, the first derivatives at the domain boundary, $\partial\phi(L_x,t)/\partial x$ and $\partial\phi(R_x,t)/\partial x$, are close to zero as shown in Figure 5(b). Hence, the homogeneous Neumann boundary condition is a good approximation of the AC equation in this case. The basic mechanism of the traveling wave from the AC equation is as follows: The initial condition is positive and has a transition layer ranging from approximately one to approximately zero. The AC equation consists of two terms: One is the diffusion term, $\partial^2\phi(x,t)/\partial x^2$, and the other is the nonlinear reaction term, $-F'(\phi(x,t))/\epsilon^2$. The diffusion term broadens the transition layer, and the nonlinear term increases the phase-field values because the nonlinear term is positive when the phase-field values are positive, as can be seen in Figure 1, resulting in the forward propagation of the transition layer.

(a)



(b)

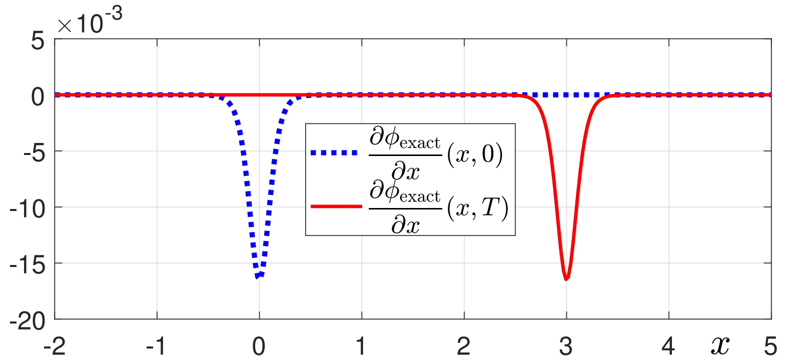


Figure 5. (a) Traveling wave solutions for the AC equation at times $t = 0$ and $t = T$. (b) First spatial derivatives of the traveling wave solutions for the AC equation at times $t = 0$ and $t = T$. Here, $\epsilon = 2h$.

Figure 6 illustrates a real-world application of solving a data classification problem using a 2D traveling wave solution. For more details, please refer to [32].

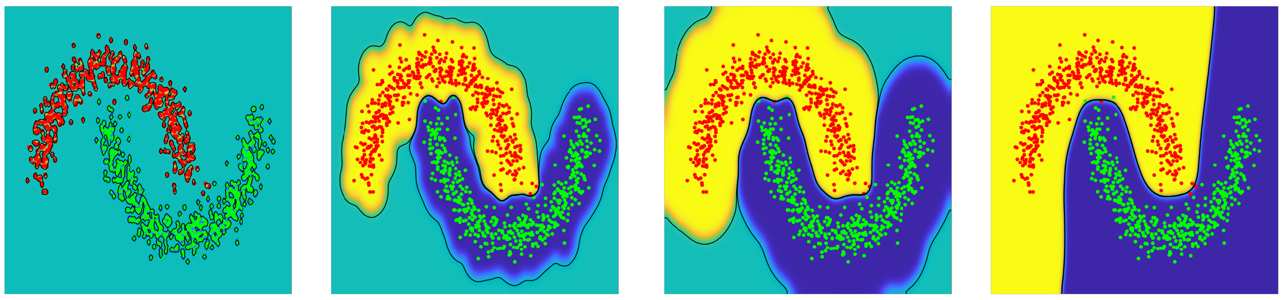


Figure 6. Application of traveling wave for data classification. The process of constructing the decision boundary (or decision surface) for classifying data points with different labels (red and green points) from left to right.

2.3. *Motion by mean curvature in 2D space*

The evolution of the solution to the AC equation under motion by mean curvature is a principal characteristic of the AC equation. Figure 7 shows a schematic diagram of the motion by mean curvature for the AC equation in the domain $\Omega = (-2, 2) \times (-1, 1)$. The dashed line denotes the initial condition, which is an ellipse. As the evolution progresses, the regions with higher curvature, such as the leftmost and rightmost points of the ellipse, contract more rapidly. Conversely, regions with lower curvature, located near the top and bottom of the ellipse, exhibit slower contraction. This behavior reflects the dependence of the motion speed on the curvature magnitude.

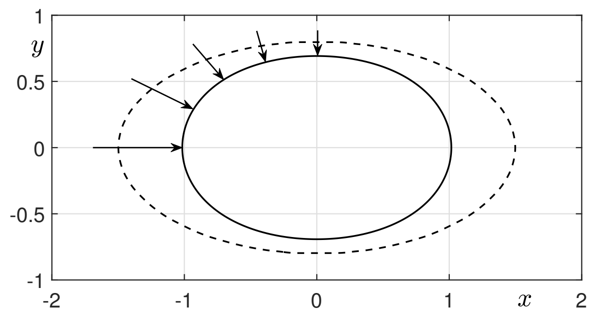


Figure 7. Schematic illustration of the motion by mean curvature using the AC equation.

As $\epsilon \to 0$, the zero level set of the solution to the AC equation converges to the motion by mean curvature [30]. For a sphere with an initial radius R_0, the mean curvature flow in d-dimensional space is governed by the following ordinary differential equation:

$$\frac{d}{dt}R(t) = -\frac{d-1}{R(t)}, \quad R(0) = R_0,$$

with the analytical solution given by

$$R(t) = \sqrt{R_0^2 - 2(d-1)t}.$$

Therefore, the second benchmark problem examines the capability of the AC equation to approximate motion by mean curvature. Specifically, we perform numerical tests to compare the temporal evolution of the radius derived from the analytical solution of the mean curvature flow with the evolution of the radius of the zero-contour obtained from the numerical solution of the AC equation. This comparison serves to validate the accuracy and reliability of the numerical approach. In 2D space, the initial condition is defined as

$$\phi(x, y, 0) = \tanh\left(\frac{R_0 - \sqrt{x^2 + y^2}}{\sqrt{2}\epsilon}\right) \tag{2.17}$$

on the computational domain $\Omega = (-1, 1) \times (-1, 1)$ with 256×256 mesh. Here, $R_0 = 0.5$, $h = 2/256$, $\epsilon = \epsilon_{13} = 13h/(2\sqrt{2}\tanh^{-1}(0.9))$, $\Delta t = 0.1h^2$, and the final time $T = 20000\Delta t$ are used. The computational solution for the radius is defined as the average of the distances from the center point $(0, 0)$ to the zero-contour of the numerical solution.

Figures 8(a)–8(c) show the computational solutions of the AC equation obtained using the fully explicit method at $t = 0$, $10000\Delta t$, and $17500\Delta t$. Here, for visualization purposes, only a subset of the entire discrete domain was used, specifically $\Omega_h = (x_i = L_x + (i - 0.5)h, y_j = L_y + (j - 0.5)h)$ for $i = 4, 8, 12, \ldots, 256$ and $j = 4, 8, 12, \ldots, 256$. Figure 8(d) shows the temporal evolution of the zero-contour of the computational solution at intervals of $2500\Delta t$. Figure 8(e) shows the temporal evolution of the analytical and numerical solutions for the radius $R(t)$, driven by the motion by mean curvature of the AC equation, up to $t = 20000\Delta t$. We observed that the computational solution obtained using the fully explicit method for solving the AC equation agrees well with the analytical solution. This indicates its suitability for benchmark problems.

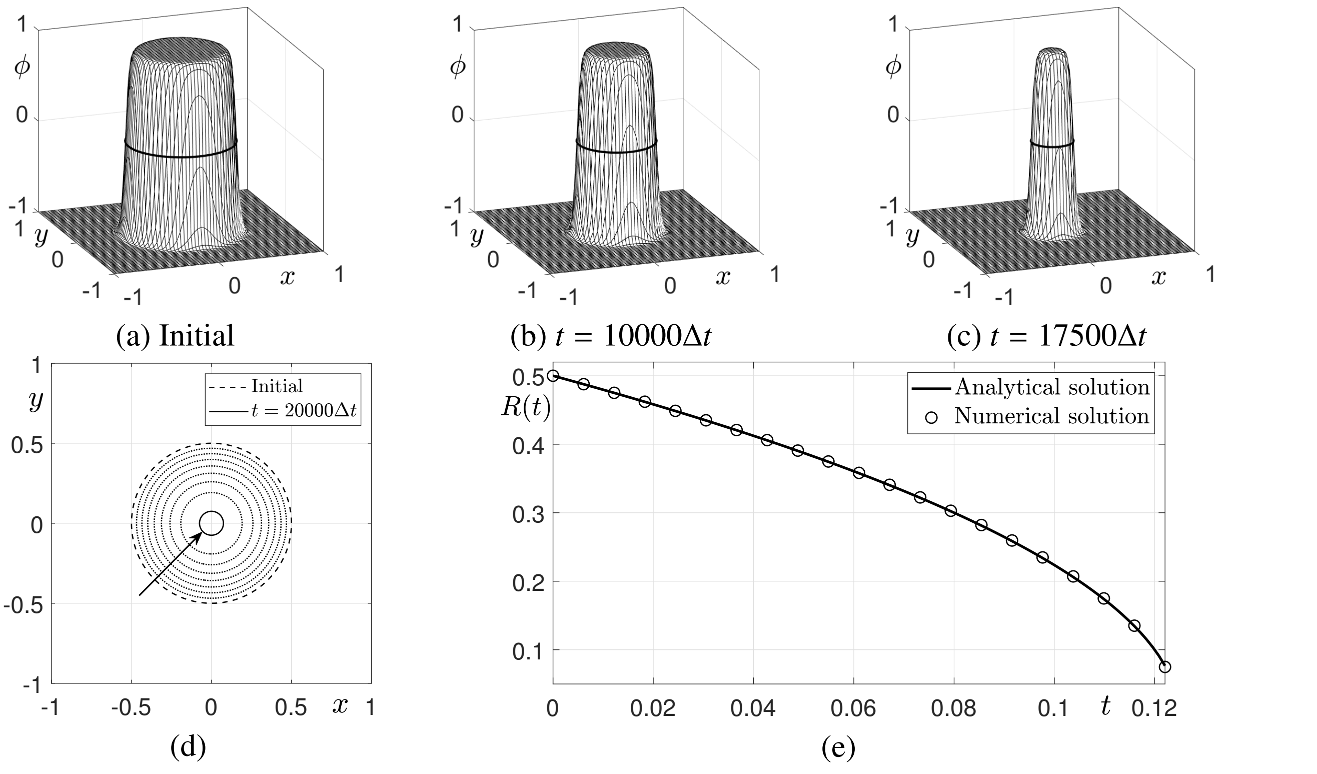
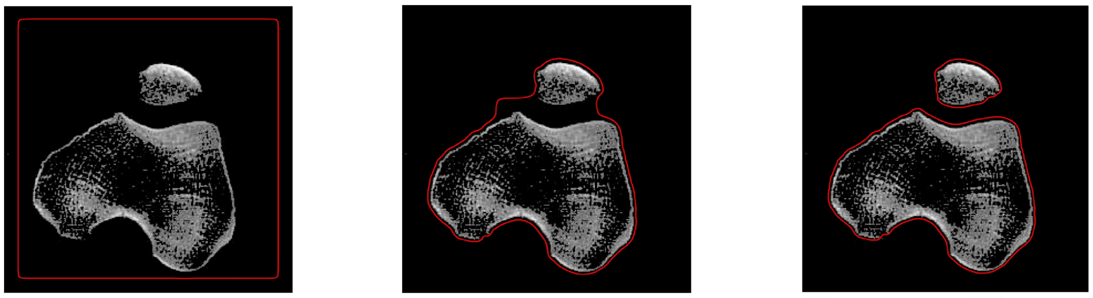


Figure 8. (a)–(c) Computational solutions at times $t = 0$, $10000\Delta t$, and $17500\Delta t$. (d) Time evolution of the radius of the zero-contour for the numerical solution at intervals of $2500\Delta t$. (e) Time evolution of the radius for the analytical and numerical solutions.

Figure 9 illustrates a real-world application of solving image segmentation and 3D volume reconstruction using the slice image problem using the motion by mean curvature property. Figure 9(a) shows the process of segmenting bones from CT slice image data, and Figure 9(b) illustrates the process of 3D volume reconstruction using the segmented slice data. For more details, please refer to [33].

(a)



(b)

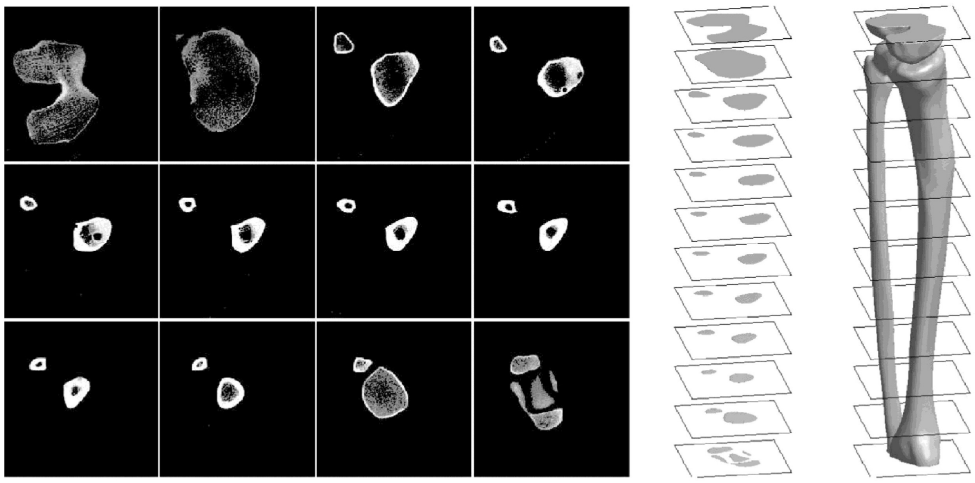


Figure 9. Application of the property of motion by mean curvature. (a) Image segmentation; (b) volume reconstruction. Reprinted from Li et al. [33] with permission from Elsevier.

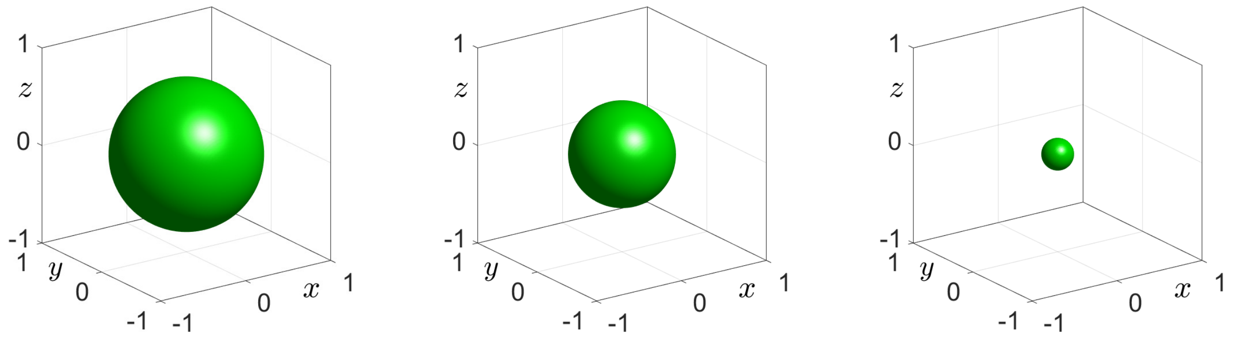
2.4. Motion by mean curvature in 3D space

Next, the initial condition on $\Omega = (-1, 1) \times (-1, 1) \times (-1, 1)$ is given by

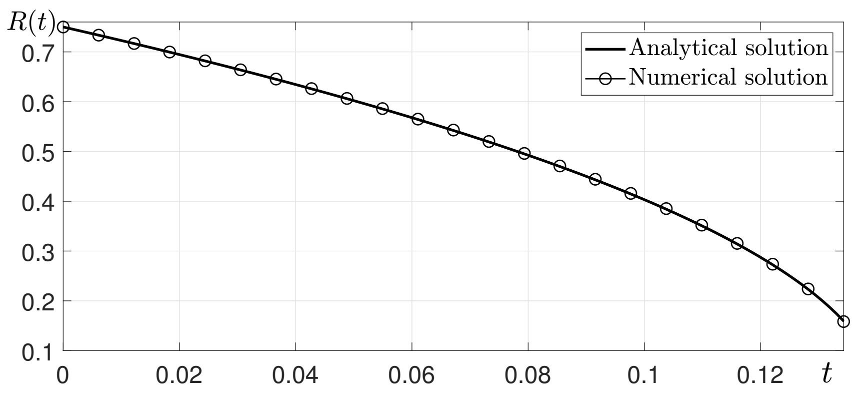
$$\phi(x, y, z, 0) = \tanh\left(\frac{R_0 - \sqrt{x^2 + y^2 + z^2}}{\sqrt{2}\epsilon}\right),$$

where the initial radius is $R_0 = 0.75$. Therefore, we obtain the analytical solution for the radius of sphere by the motion by mean curvature as $R(t) = \sqrt{R_0^2 - 4t}$. The parameters used are $N_x = N_y = N_z = 128$, $\epsilon = \epsilon_{11} = 11h/(2\sqrt{2}\tanh^{-1}(0.9))$, $\Delta t = 0.1h^2$, and the final time $T = 5500\Delta t$. We define the numerical radius in 3D space as the average distance from the center point $(0, 0, 0)$ to the zero-level isosurface of the numerical solution.

Figures 10(a)–10(c) show snapshots of the computational solutions of the AC equation in 3D space at $t = 0$, $3000\Delta t$, and $5500\Delta t$. The temporal evolution of the radius for both the analytical and numerical solutions is presented in Figure 10(d).



(a) Initial (b) t = 3000Δt (c) t = 5500Δt



(d)

Figure 10. (a)–(c) Computational solutions at times $t = 0$, $3000\Delta t$, and $5500\Delta t$. (d) Temporal evolution of the radius for the analytical and numerical solutions.

3. Conclusions

In this work, we have developed and presented a set of benchmark problems based on the AC equation to evaluate the effectiveness and precision of physics-informed neural networks. These benchmarks include 1D, 2D, and 3D cases such as conventional problem, traveling wave solution, and motion by mean curvature. By providing benchmark problems with numerical approximations, we have demonstrated their ability to serve as reliable and robust tools for assessing PINNs. The results indicate that the AC equation, with its intrinsic properties such as travelling wave and motion by mean curvature, provides an excellent framework for generating diverse benchmark problems. The availability of source codes for these benchmarks further facilitates the adoption and validation of PINN methodologies in the broader scientific community. Future research can build upon these benchmarks by exploring more complex systems, incorporating additional physical constraints, or extending the benchmarks to other reaction-diffusion equations. These enhancements will contribute to the ongoing development and refinement of PINNs as a computational tool for solving partial differential equations.

Author contributions

Hyun Geun Lee: Conceptualization, Software, Data curation, Resources, Formal analysis, Visualization, Writing – original draft, and Writing – review & editing. Youngjin Hwang: Formal analysis, Software, Resources, Validation, Visualization, Writing – original draft, and Writing – review & editing. Yunjae Nam: Validation, Investigation, Visualization, and Writing – original draft, and Writing – review & editing. Sangkwon Kim: Data curation, Validation, Investigation, Software, Resources, Writing – original draft, and Writing – review & editing. Junseok Kim: Conceptualization, Resources, Formal analysis, Supervision, Funding acquisition, Methodology, Project administration, Writing – original draft, and Writing – review & editing.

Acknowledgments

The first author (H.G. Lee) was supported by the National Research Foundation of Korea (NRF) grant funded by the Korea government (MSIT) (RS-2022-NR069708). We are grateful to the reviewers for their constructive feedback and thoughtful suggestions, which have greatly improved the original manuscript.

Conflict of interest

The authors declare no conflicts of interest.

Use of Generative AI tools declaration

The authors declare they have not used Artificial Intelligence (AI) tools in the creation of this article.

Appendix

The MATLAB codes are provided below. Listing 1 provides the code for computing the benchmark problem of the 1D AC equation with the travelling wave solution. Listings 2 and 3 each provide a code that approximates the motion by mean curvature flow, which is one of the properties of the AC equation in 2D and 3D spaces, respectively. These codes are used as benchmark problems to evaluate the efficiency and accuracy of PINNs.

Listing 1. Travelling wave solution.

```
numx=301; x=linspace(-2,5,numx); h=x(2)-x(1); t=0; epsil=2*h;
phi=@(t) 0.5*(1-tanh((x-3*t/(sqrt(2)*epsil))/(2*sqrt(2)*epsil)));
plot(x,phi(t),'b:','linewidth',3); hold on
t=sqrt(2)*epsil; plot(x,phi(t),'r','linewidth',2);
a=0.05; axis([x(1) x(end) 0-a 1+a]); xlabel('x')
legend('\phi_{exact}(x,0)','\phi_{exact}(x,T)');
box on; grid on; figure(2); clf;
```

```
dphi=@(t) -(1/4*sqrt(2)*epsil) ...
    *(sech((x-3*t/(sqrt(2)*epsil))/(2*sqrt(2)*epsil))).^2;
t=0; plot(x,dphi(t),'b:','linewidth',3); hold on
t=sqrt(2)*epsil; plot(x,dphi(t),'r','linewidth',2);
axis([x(1) x(end) -0.02 0.005])
xlabel('x'); legend('\partial \phi_{\rm exact}(x,0)/\partial x', ...
    '\partial \phi_{\rm exact}(x,T)/\partial x'); box on; grid on;
```

Listing 2. Two-dimensional AC equation.

```
Numx = 256; Numy = 256; x1=-1; x2=1; y1=-1; y2=1; h=(x2-x1)/Numx;
xi=x1-0.5*h:h:x2+0.5*h; yj=y1-0.5*h:h:y2+0.5*h;
dt = 0.1*h^2; Numt = 20000; m = 13; epsil = 0.5*m*h/(sqrt(2)*atanh(0.9))
r0 = 0.5;
for ii=[1:Numx+2]
for jj=[1:Numy+2]
    osol(ii,jj) = tanh((r0-sqrt(xi(ii)^2+yj(jj)^2))/(sqrt(2)*epsil));
end
end
nsol = osol; ct = 1;
[M,y1] = contour(xi(2:Numx+1),yj(2:Numy+1), ...
    osol(2:Numx+1,2:Numy+1),[0 0],'k-');
nR(ct) = mean(sqrt(sum((M(:,2:end)).^2)));
aR(ct) = sqrt(r0^2);
for it = 1:Numt
osol(1,:)=osol(2,:); osol(Numx+2,:)=osol(Numx+1,:);
osol(:,1)=osol(:,2); osol(:,Numy+2)=osol(:,Numy+1);
for ii = 2:Numx+1
for jj = 2:Numy+1
    nsol(ii,jj) = osol(ii,jj)+dt*((osol(ii,jj)-osol(ii,jj)^3)/epsil^2 ...
        +(osol(ii-1,jj)+osol(ii+1,jj)+osol(ii,jj-1)+osol(ii,jj+1) ...
        -4*osol(ii,jj))/h^2);
end
end
osol = nsol;
if mod(it,200) == 0
    ct = ct+1;
    [M,y1] = contour(xi(2:Numx+1),yj(2:Numy+1), ...
        osol(2:Numx+1,2:Numy+1),[0 0],'k-');
    nR(ct) = mean(sqrt(sum((M(:,2:end)).^2)));
    aR(ct) = sqrt(r0^2-2*dt*it);
end
end
```

```
axisfs = 18; fs = 23; lw = 2; ms = 12;
t = [0:200:it]*dt;
plot(t,aR,'k-'); hold on; plot(t,nR,'ko');
legend('Analytical solution','Numerical solution')
xlabel('t'); ylabel('R(t)');
```

Listing 3. Three-dimensional AC equation.

```
Numx = 128; Numy = 128; Numz = 128;
x1=-1; x2=1; y1=-1; y2=1; z1=-1; z2=1; h=(x2-x1)/Numx;
xi=x1-0.5*h:h:x2+0.5*h; yj=y1-0.5*h:h:y2+0.5*h;
zk=z1-0.5*h:h:z2+0.5*h;
dt = 0.1*h^2; Numt = 5500; m = 11;
epsil = 0.5*m*h/(sqrt(2)*atanh(0.9)); r0 = 0.75;
for ii = 1:Numx+2
for jj = 1:Numy+2
for kk = 1:Numz+2
    osol(ii,jj,kk) = tanh((r0-sqrt(xi(ii)^2+yj(jj)^2 ...
        +zk(kk)^2))/(sqrt(2)*epsil));
end
end
end
nsol = osol;
pltp = isosurface(xi,yj,zk,osol,0);
patch(pltp,'facealpha',1,'facecolor','g','edgecolor','none');
axis image; axis([x1 x2 y1 y2 z1 z2]); view([-35 20]);
ct = 1; nR(ct) = mean(sqrt(sum(pltp.vertices.^2,2)));
aR(ct) = sqrt(r0^2);
for it = 1:Numt
osol(1,:,:)=osol(2,:,:); osol(Numx+2,:,:)=osol(Numx+1,:,:);
osol(:,1,:)=osol(:,2,:); osol(:,Numy+2,:)=osol(:,Numy+1,:);
osol(:,:,1)=osol(:,:,2); osol(:,:,Numz+2)=osol(:,:,Numz+1);
for ii = 2:Numx+1
for jj = 2:Numy+1
for kk = 2:Numz+1
    nsol(ii,jj,kk) = osol(ii,jj,kk) ...
    +dt*((osol(ii,jj,kk)-osol(ii,jj,kk).^3)/epsil^2 ...
    +(osol(ii-1,jj,kk)+osol(ii+1,jj,kk) ...
    +osol(ii,jj-1,kk)+osol(ii,jj+1,kk) ...
    +osol(ii,jj,kk-1)+osol(ii,jj,kk+1)-6*osol(ii,jj,kk))/h^2);
end
end
end
```

```
    osol = nsol;
    if mod(it,50) == 0
        ct = ct+1; pltp = isosurface(xi,yj,zk,osol,0);
        nR(ct) = mean(sqrt(sum(pltp.vertices.^2,2)));
        aR(ct) = sqrt(r0^2-4*dt*it);
    end
    end
    axisfs = 18; fs = 23; lw = 2; ms = 12; t = [0:50:it]*dt;
    plot(t,aR,'k-'); hold on; plot(t,nR,'k-o');
    legend('Analytical solution','Numerical solution')
    xlabel('t'); ylabel('R(t)');
```

References

1. S. M. Allen, J. W. Cahn, A microscopic theory for antiphase boundary motion and its application to antiphase domain coarsening, *Acta Metall.*, **27** (1979), 1085–1095. https://doi.org/10.1016/0001-6160(79)90196-2

2. W. Xie, Q. Xia, Q. Yu, Y. Li, An effective phase field method for topology optimization without the curvature effects, *Comput. Math. Appl.*, **146** (2023), 200–212. https://doi.org/10.1016/j.camwa.2023.06.037

3. Y. Wang, J. Yang, X. Xiao, X. Feng, Efficient numerical simulation for the dendritic crystal growth with melt convection in complex domains, *Int. J. Heat Mass Transfer,* **233** (2024), 126036. https://doi.org/10.1016/j.ijheatmasstransfer.2024.126036

4. Y. Li, K. Qin, Q. Xia, J. Kim, A second-order unconditionally stable method for the anisotropic dendritic crystal growth model with an orientation-field, *Appl. Numer. Math.,* **184** (2023), 512–526. https://doi.org/10.1016/j.apnum.2022.11.006

5. C. Liu, Z. Qiao, Q. Zhang, Multi-phase image segmentation by the Allen–Cahn Chan–Vese model, *Comput. Math. Appl.*, **141** (2023), 207–220. https://doi.org/10.1016/j.camwa.2022.12.020

6. Q. Xia, J. Kim, B. Xia, Y. Li, An unconditionally energy stable method for binary incompressible heat conductive fluids based on the phase-field model, *Comput. Math. Appl.*, **123** (2022), 26–39. https://doi.org/10.1016/j.camwa.2022.07.022

7. Q. Xia, Y. Liu, J. Kim, Y. Li, Binary thermal fluids computation over arbitrary surfaces with second-order accuracy and unconditional energy stability based on phase-field model, *J. Comput. Appl. Math.*, **433** (2023), 115319. https://doi.org/10.1016/j.cam.2023.115319

8. Z. Lu, J. Wang, A novel and efficient multi-scale feature extraction method for EEG classification, *AIMS Math.*, **9** (2024), 16605–16622. https://doi.org/10.3934/math.2024805

9. S. Sun, Q. Gong, Y. Ni, M. Yi, A micromagnetic-mechanically coupled phase-field model for fracture and fatigue of magnetostrictive alloys, *J. Mech. Phys. Solids*, **191** (2024), 105767. https://doi.org/10.1016/j.jmps.2024.105767

10. Z. Wang, G. Wang, A coupled MPM and CBFEM framework for large deformation simulation of porous media interacting with pore and free fluid, *Comput. Geotech.*, **163** (2023), 105746. https://doi.org/10.1016/j.compgeo.2023.105746

11. B. Xia, X. Xi, R. Yu, P. Zhang, Unconditional energy-stable method for the Swift–Hohenberg equation over arbitrarily curved surfaces with second-order accuracy, *Appl. Numer. Math.*, **198** (2024), 192–201. https://doi.org/10.1016/j.apnum.2024.01.005

12. X. Hu, Q. Xia, B. Xia, Y. Li, A second-order accurate numerical method with unconditional energy stability for the Lifshitz–Petrich equation on curved surfaces, *Appl. Math. Lett.*, **163** (2025), 109439. https://doi.org/10.1016/j.aml.2024.109439

13. M. Raissi, P. Perdikaris, G. E. Karniadakis, Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations, *J. Comput. Phys.*, **378** (2019), 686–707. https://doi.org/10.1016/j.jcp.2018.10.045

14. C. L. Wight, J. Zhao, Solving Allen–Cahn and Cahn–Hilliard equations using the adaptive physics informed neural networks, *Commun. Comput. Phys.*, **29** (2021), 930–954. https://doi.org/10.4208/cicp.OA-2020-0086

15. X. Song, B. Xia, Y. Li, An efficient data assimilation based unconditionally stable scheme for Cahn–Hilliard equation, *Comput. Appl. Math.*, **43** (2024), 121. https://doi.org/10.1007/s40314-024-02632-7

16. X. Song, Q. Xia, J. Kim, Y. Li, An unconditional energy stable data assimilation scheme for Navier–Stokes–Cahn–Hilliard equations with local discretized observed data, *Comput. Math. Appl.*, **164** (2024), 21–33. https://doi.org/10.1016/j.camwa.2024.03.018

17. Y. Li, R. Liu, Q. Xia, C. He, Z. Li, First- and second-order unconditionally stable direct discretization methods for multi-component Cahn–Hilliard system on surfaces, *J. Comput. Appl. Math.*, **401** (2022), 113778. https://doi.org/10.1016/j.cam.2021.113778

18. X. Meng, Z. Li, D. Zhang, G. E. Karniadakis, PPINN: Parareal physics-informed neural network for time-dependent PDEs, *Comput. Methods Appl. Mech. Eng.*, **370** (2020), 113250. https://doi.org/10.1016/j.cma.2020.113250

19. A. D. Jagtap, E. Kharazmi, G. E. Karniadakis, Conservative physics-informed neural networks on discrete domains for conservation laws: Applications to forward and inverse problems, *Comput. Methods Appl. Mech. Eng.*, **365** (2020), 113028. https://doi.org/10.1016/j.cma.2020.113028

20. L. Lu, X. Meng, Z. Mao, G. E. Karniadakis, Deepxde: A deep learning library for solving differential equations, *SIAM Rev.*, **63** (2021), 208–228. https://doi.org/10.1137/19M1274067

21. S. J. Anagnostopoulos, J. D. Toscano, N. Stergiopulos, G. E. Karniadakis, Residual-based attention in physics-informed neural networks, *Comput. Mech.*, **42** (2024), 1273–1285. https://doi.org/10.1016/j.cma.2024.116805

22. P. Roy, S. T. Castonguay, Exact enforcement of temporal continuity in sequential physics-informed neural networks, *Comput. Methods Appl. Mech. Eng.*, **430** (2024), 117197. https://doi.org/10.1016/j.cma.2024.117197

23. S. Monaco, D. Apiletti, Training physics-informed neural networks: One learning to rule them all, *Results Eng.*, **18** (2023), 101023. https://doi.org/10.1016/j.rineng.2023.101023

24. X. Huang, T. Alkhalifah, Efficient physics-informed neural networks using hash encoding, *J. Comput. Phys.*, **501** (2024), 112760. https://doi.org/10.1016/j.jcp.2024.112760

25. L. D. McClenny, U. M. Braga-Neto, Self-adaptive physics-informed neural networks, *J. Comput. Phys.*, **474** (2023), 111722. https://doi.org/10.1016/j.jcp.2022.111722

26. A. Kopaniaáková, H. Kothari, G. E. Karniadakis, R. Krause, Enhancing training of physics-informed neural networks using domain decomposition-based preconditioning strategies, *SIAM J. Sci. Comput.*, **46** (2024), S46–S67. https://doi.org/10.1137/23M1583375

27. R. Qiu, R. Huang, Y. Xiao, J. Wang, Z. Zhang, J. Yue, et al., Physics-informed neural networks for phase-field method in two-phase flow, *Phys. Fluids*, **34** (2022). https://doi.org/10.1063/5.0091063

28. Y. Ghaffari Motlagh, P. K. Jimack, R. de Borst, Deep learning phase-field model for brittle fractures, *Int. J. Numer. Methods Eng.*, **124**, (2023), 620–638. https://doi.org/10.1002/nme.7135

29. W. Li, M. Z. Bazant, J. Zhu, Phase-Field DeepONet: Physics-informed deep operator neural network for fast simulations of pattern formation governed by gradient flows of free-energy functionals, *Comput. Meth. Appl. Mech. Eng.*, **416** (2023), 116299. https://doi.org/10.1016/j.cma.2023.116299

30. 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. https://doi.org/10.1016/j.matcom.2023.01.016

31. M. Penwarden, A. D. Jagtap, S. Zhe, G. E. Karniadakis, R. M. Kirby, A unified scalable framework for causal sweeping strategies for Physics-Informed Neural Networks (PINNs) and their temporal decompositions, *J. Comput. Phys.*, **493** (2023), 112464. https://doi.org/10.1016/j.jcp.2023.112464

32. S. Kim, J. Kim, Automatic binary data classification using a modified Allen–Cahn equation, *Int. J. Pattern Recognit. Artif. Intell.*, **35** (2021), 2150013. https://doi.org/10.1142/S0218001421500130

33. Y. Li, J. Shin, Y. Choi, J. Kim, Three-dimensional volume reconstruction from slice data using phase-field models, *Comput. Vis. Image Underst.*, **137** (2015), 115–124. https://doi.org/10.1016/j.cviu.2015.02.001

AIMS Press