

Kernel Principal Component Analysis for Allen–Cahn Equations

Yusuf Çakır *  and Murat Uzunca 

Department of Mathematics, Sinop University, Sinop 57000, Türkiye; muzunca@sinop.edu.tr

* Correspondence: ycakir@sinop.edu.tr

Abstract: Different researchers have analyzed effective computational methods that maintain the precision of Allen–Cahn (AC) equations and their constant security. This article presents a method known as the reduced-order model technique by utilizing kernel principle component analysis (KPCA), a nonlinear variation of traditional principal component analysis (PCA). KPCA is utilized on the data matrix created using discrete solution vectors of the AC equation. In order to achieve discrete solutions, small variations are applied for dividing up extraterrestrial elements, while Kahan’s method is used for temporal calculations. Handling the process of backmapping from small-scale space involves utilizing a non-iterative formula rooted in the concept of the multidimensional scaling (MDS) method. Using KPCA, we show that simplified sorting methods preserve the dissipation of the energy structure. The effectiveness of simplified solutions from linear PCA and KPCA, the retention of invariants, and computational speeds are shown through one-, two-, and three-dimensional AC equations.

Keywords: Allen–Cahn equation; kernel principle component analysis; multidimensional scaling; energy dissipation

MSC: 68T99; 65F55



Citation: Çakır, Y.; Uzunca, M. Kernel Principal Component Analysis for Allen–Cahn Equations. *Mathematics* **2024**, *12*, 3434. <https://doi.org/10.3390/math12213434>

Received: 8 October 2024

Revised: 30 October 2024

Accepted: 30 October 2024

Published: 2 November 2024



Copyright: © 2024 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

1. Introduction

Differential equations, particularly partial differential equations (PDEs), are commonly used to mathematically represent most real-life problems in different scientific fields. Among them, the Allen–Cahn (AC) equation is a popular nonlinear PDE for modeling phase transition issues. The AC equation takes the form of a reaction–diffusion equation, where the phase field evolves over time to minimize a free energy functional. It is characterized by a double-well potential, which encourages the formation of distinct phases and sharp interfaces between them. As time progresses, the solutions to the AC equation exhibit intricate patterns and structures, making it a crucial tool for understanding phenomena such as spinodal decomposition and the evolution of microstructures in alloys. The AC equation was firstly presented in [1] to simulate the movement of antiphase boundaries in crystalline solids. The AC equation is commonly applied to represent different natural phenomena and has become a fundamental equation for the diffuse interface method used to investigate phase transitions and interfacial dynamics in various fields such as materials science [2], image analysis [3], fluid dynamics [4], and mean curvature flow [5]. In materials science, it is used to simulate phase transformations in polymer, metal, and ceramic materials [6]. AC equation is further used in multi-reconstruction [7] and predicting the evolution of microstructures in various materials and also in biophysics [8].

The AC equation defined by

$$u_t = \mu^2 \Delta u - f(u), \quad (x, t) \in \Omega \times (0, T], \quad (1)$$

explains the movement of the anti-phase borders of a dual metal mixture at a set temperature. In Equation (1), $u(x, t)$ is used to denote the amount of one metal component in

the alloy while the positive parameter μ represents the narrow inter-facial width that is smaller than the lab scale length. We examine the AC Equation (1) on a domain $\Omega \subset \mathbb{R}^d$, ($d = 1, 2, 3$), enforced with homogeneous Neumann or periodic boundary conditions, and with the double-well potential $F(u) = (u^2 - 1)^2/4$. The nonlinear functional in (1) is given as $f(u) = F'(u) = u - u^3$, and the free energy functional of the AC equation is defined by

$$\mathcal{E}(u) = \int_{\Omega} \left(\frac{\mu^2}{2} |\nabla u|^2 + F(u) \right) dx. \quad (2)$$

In this form, the AC Equation (1) is a dissipative gradient system

$$u_t = -\frac{\delta \mathcal{E}}{\delta u},$$

where $\delta \mathcal{E} / \delta u$ represents the variational derivative of the free energy. In other words, the AC equation meets a non-linear stability condition, which means that the free energy functional $\mathcal{E}(u)$ decreases over time, i.e., $\mathcal{E}(u(x, t_n)) \leq \mathcal{E}(u(x, t_m))$ for $t_n > t_m$.

The AC equation often requires long-time simulations due to the nature of the phase separation and interface dynamics it models. As the system evolves, the formation and evolution of interfaces between different phases can lead to intricate patterns and structures that take time to develop fully. Capturing these dynamics accurately involves tracking the evolution of the order parameter over extended periods, especially when the processes involve slow kinetics or the emergence of complex morphologies. Additionally, the interplay between diffusion and reaction processes can result in long-lived transients, making it essential to run simulations for sufficient time to observe the stable states and equilibrium configurations. Long-time simulations allow researchers to gain deeper insights into the stability, coarsening behavior, and final patterns that emerge in the system, providing a comprehensive understanding of the underlying physical phenomena.

Long-time simulations in complex systems, like the AC equation, often require significant computational resources, making them time-consuming and costly. Dimensionality reduction for dynamic systems is a crucial process aimed at simplifying complex models while retaining the essential features of their behavior. In many real-world applications, such as fluid dynamics, climate modeling, and biological systems, the state space can be incredibly high-dimensional, making simulations computationally expensive and difficult to analyze. Intrusive and data-driven reduced order modeling (ROM) are two distinct approaches used for dimensionality reduction. Intrusive ROM relies on integrating reduced models directly into the numerical algorithms of the original system, often requiring detailed knowledge of the governing equations. In contrast, data-driven ROM leverages discrete data to construct models, utilizing techniques like machine learning and statistical analysis to capture system behavior without explicit knowledge of the underlying physics. Among them, principal component analysis (PCA) is a robust statistical method used for reducing dimensionality and visualizing data across fields like image processing, genetics, finance, and social sciences. It transforms original variables into principal components that maximize variance, aiming to create a lower-dimensional representation while retaining original variability [9,10]. PCA's effectiveness diminishes when data distributions exhibit low variance, making it less suitable for linear subspace decomposition.

Recently, kernel principal component analysis (KPCA) has emerged as an advanced alternative, which is an extension of traditional PCA that enables the analysis of nonlinear relationships within high-dimensional data. By employing a kernel function, KPCA implicitly maps the original data into a higher-dimensional feature space where linear separability is often achieved, allowing for more effective dimensionality reduction and feature extraction. This method excels in capturing complex patterns that conventional PCA might overlook, making it particularly useful in fields such as exploratory data analysis, pattern recognition, face recognition, manifold learning across fields like computer vision, bio-informatics, broad learning systems, meshfree approaches, and signal processing [11–19].

However, KPCA faces challenges in interpreting results in the original input space and the computational cost associated with kernel evaluations, with multidimensional scaling (MDS) being a notable approach for this purpose [14,20,21]. MDS allows for the iterative computation of the pre-image by employing various distance metrics between the data vectors. Despite these challenges, KPCA remains a powerful tool for exploratory data analysis and is widely utilized in applications.

This paper explores the application of KPCA to data that are derived from discrete solution vectors of the AC equation, a novel approach in ROM that preserves solution accuracy and conservation of the energy dissipation property of the AC equation. In the literature, there are rare papers dealing with the ROM for AC equation. An energy-stable ROM using finite differences and convex splitting is derived for the AC equation in [22], and a non-linear POD/Galerkin ROM is applied in [23]. In [24], the authors derive an energy-stable ROM for AC equation using discontinuous Galerkin and average vector field methods, utilizing proper orthogonal decomposition-greedy adaptive sampling and a discrete empirical interpolation method.

Below, the paper outlines the AC equation's mathematical formulation with space/time discretization in Section 2; details the KPCA process in Section 3, showcases results related to reduced solution accuracy and computational efficiency for the one-, two- and three-dimensional AC equations in Section 4; and concluding with key findings.

2. Discrete Data from AC Equations

In this section, we present the spatio-temporal (full) discrete representation of the AC equation based on its free energy (2), where the solution vectors are utilized to construct the data set for further analysis.

2.1. Space Discretization

The AC equation is first discretized in space. With this goal in mind, we employ finite difference discretization utilizing a tensor framework. To be more clear, consider a partition of the domain $\Omega \subset \mathbb{R}^d$ ($d = 1, 2, 3$) into m discrete grid nodes, and with the spatial variable $x = (x_1, \dots, x_d)$. On this partition, we establish a time-varying semi-discrete solution vector $\mathbf{u}(t) : [0, T] \mapsto \mathbb{R}^m$ as

$$\mathbf{u}(t) = (u_1(t), \dots, u_m(t))^T, \quad (3)$$

with $u_i(t)$ representing the approximate solution at time t and the i -th grid node of the partitioned domain. Also let the matrix $A \in \mathbb{R}^{m \times m}$ denote the discrete Laplace operator approximating the second-order derivative term in the AC equation. Under the given spatial discretization, the m -dimensional semi-discrete (dynamical) system has the form

$$\dot{\mathbf{u}} = \mu^2 A \mathbf{u} - \mathbf{f}(\mathbf{u}), \quad (4)$$

where $\dot{\mathbf{u}}$ is the ordinary differentiation with respect to time variable t , and the m -dimensional vector $\mathbf{f} : \mathbb{R}^m \mapsto \mathbb{R}^m$ is such that $f_i(\mathbf{u}) = u_i - u_i^3$, $i = 1, \dots, m$.

The form of the matrix A , together with the ordering of the discrete data within the semi-discrete solution vector $\mathbf{u}$ and hence within the vector $\mathbf{f}(\mathbf{u})$, is effected by both the ordering of the grid nodes on the mesh and the boundary conditions. Here, we first construct the discrete setting for a one-dimensional domain (interval) $\Omega \subset \mathbb{R}$, and then we use an extension to two- and three-dimensional domains utilizing a tensor framework.

The partitioned mesh of the domain $\Omega \subset \mathbb{R}^d$ is constructed by the m_i grid nodes in either spatial axis x_i , $i = 1, 2, 3$. Let, on some x_i -axis, the matrix $D_i \in \mathbb{R}^{m_i \times m_i}$ stand for the matrix corresponding to the discretization of the second-order partial differentiation with respect to x_i , by finite differences, for a one-dimensional spatial interval $[x_i^l, x_i^r]$ with a mesh size Δx_i . If a periodic boundary condition is imposed, we have $\Delta x_i = (x_i^r - x_i^l)/m_i$, and the mesh consists of the grid nodes at which the solution is unknown, which are the first m_i nodes $x_{i,j} = x_i^l + (j-1)\Delta x_i$, $j = 1, 2, \dots, m_i$, excluding the right boundary node x_i^r , because

of the periodicity. On the other hand, if homogeneous Neumann boundary is imposed, then solutions are unknown at each m_i grid node $x_{i,j} = x_i^l + (j-1)\Delta x_i$, $j = 1, 2, \dots, m_i$, including both left and right boundary nodes, i.e., the mesh size is $\Delta x_i = (x_i^r - x_i^l)/(m_i - 1)$. The corresponding matrix form of $D_i \in \mathbb{R}^{m_i \times m_i}$ under periodic and homogeneous Neumann boundary conditions are given, respectively, by

$$D_i := \frac{1}{\Delta x_i^2} \begin{pmatrix} -2 & 1 & & & 1 \\ 1 & -2 & 1 & & \\ & \ddots & \ddots & \ddots & \\ & & 1 & -2 & 1 \\ 1 & & & 1 & -2 \end{pmatrix}, \quad D_i := \frac{1}{\Delta x_i^2} \begin{pmatrix} -1 & 1 & & & 0 \\ 1 & -2 & 1 & & \\ & \ddots & \ddots & \ddots & \\ & & 1 & -2 & 1 \\ 0 & & & 1 & -1 \end{pmatrix}.$$

2.1.1. One-Dimensional Domain

Consider a one-dimensional domain $[x_1^l, x_1^r] \subset \mathbb{R}$ whose uniformly partitioned mesh consists of m_1 grid nodes $x_{1,1}, \dots, x_{1,m_1}$ on the x_1 -axis and has a mesh size Δx_1 (as discussed above, the mesh size formula depends on the boundary condition, just to be able to fix the number of nodes). On such a one-dimensional grid, we simply have in the dynamical system (4) that $A = D_1$ and $m = m_1$. In addition, the entries of the solution form in (3) are given simply by $u_i(t) = u(x_{1,i}, t)$.

2.1.2. Extension to Higher Space Dimension

The form of the dynamical system (4) for a spatial domain with dimension $d > 1$ can be easily constructed by the use of matrix D_i defined on a single x_i -axis and utilizing a tensor framework [6].

On a two-dimensional rectangular domain $\Omega = [x_1^l, x_1^r] \times [x_2^l, x_2^r] \subset \mathbb{R}^2$, let the mesh consist of $m = m_1 m_2$ grid nodes, which are the intersection nodes of m_1 grid nodes $x_{1,1}, \dots, x_{1,m_1}$ on the x_1 -axis and m_2 grid nodes $x_{2,1}, \dots, x_{2,m_2}$ on the x_2 -axis, with the uniform mesh sizes Δx_1 and Δx_2 , respectively. On this mesh, in order to determine the form of the solution entries in (3), we should determine an ordering of the nodes, as the way of the ordering in higher dimensions changes the form of the matrix A in the dynamical system (4). For the ordering of the nodes, we associate to each $m = m_1 m_2$ index i a distinct pair of indices as $i = \{j, l\}$ for $j = 1, \dots, m_1$ and $l = 1, \dots, m_2$. Under this setting, an individual entry of the solution form in (3) is given by $u_i(t) := u_{j,l}(t) = u((x_{1,j}, x_{2,l}), t)$, giving the approximate solution at time t and at the location $x = (x_{1,j}, x_{2,l}) \in \Omega$. Then, the ordering is taken by writing the solution vector in the following form:

$$\mathbf{u}(t) = (u_{1,1}(t), \dots, u_{m_1,1}(t), u_{1,2}(t), \dots, u_{m_1,2}(t), \dots, u_{m_1,m_2}(t))^T.$$

According to the given ordering, the matrix A in (4) can be defined by the formula

$$A = I_{m_2} \otimes D_1 + D_2 \otimes I_{m_1},$$

where $\otimes$ denotes the Kronecker product, and for some positive integer s , I_s is the identity matrix of size s .

Finally, on a three-dimensional cuboid $\Omega = [x_1^l, x_1^r] \times [x_2^l, x_2^r] \times [x_3^l, x_3^r] \subset \mathbb{R}^3$, let the mesh consist of $m = m_1 m_2 m_3$ grid nodes, which are the intersection nodes of m_1 grid nodes $x_{1,1}, \dots, x_{1,m_1}$ on the x_1 -axis, m_2 grid nodes $x_{2,1}, \dots, x_{2,m_2}$ on the x_2 -axis, and m_3 grid nodes $x_{3,1}, \dots, x_{3,m_3}$ on the x_3 -axis, with the uniform mesh sizes Δx_1 , Δx_2 and Δx_3 , respectively. For the ordering of the nodes, we associate with each $m = m_1 m_2 m_3$ index i a distinct pair of indices as $i = \{j, l, s\}$ for $j = 1, \dots, m_1$, $l = 1, \dots, m_2$ and $s = 1, \dots, m_3$. Under this setting, an individual entry of the solution form in (3) is given by $u_i(t) := u_{j,l,s}(t) = u((x_{1,j}, x_{2,l}, x_{3,s}), t)$, giving the approximate solution at time t and at the location

$x = (x_{1,j}, x_{2,l}, x_{3,s}) \in \Omega$. Then, the ordering is taken by writing the solution vector in the following form:

$$\mathbf{u}(t) = (u_{1,1,1}(t), \dots, u_{1,m_2,1}(t), u_{2,1,1}(t), \dots, u_{2,m_2,1}(t), \dots, u_{m_1,m_2,1}(t), \dots, u_{m_1,m_2,m_3}(t))^T,$$

which is the default ordering of the MatLab's *meshgrid* function on three-dimensional grids. According to the given ordering, the matrix A in (4) can be defined by the formula

$$A = I_{m_3} \otimes I_{m_2} \otimes D_1 + I_{m_3} \otimes D_2 \otimes I_{m_1} + D_3 \otimes I_{m_2} \otimes I_{m_1}.$$

We note that the given tensor framework with an appropriate ordering of the grid nodes can be easily used to construct the dynamical system (4) for an arbitrary space dimension d . Yet it is sufficient to give formulations for $d = 1, 2, 3$ in the case of the real-life problems.

2.2. Temporal Integration

The full discrete system for the AC equation is derived by applying a temporal integration technique to the semi-discrete system (4). Since the effective non-intrusive dimension reduction methods rely on accurate and reliable data, and the data set used in this paper is composed of the discrete solution vectors of the full discrete system of the AC equation, it is crucial to use a temporal integration that is both accurate and preserves the energy dissipation property of the AC equation. An energy-stable scheme is a numerical method that maintains the energy dissipation/conservation of a gradient/Hamiltonian flow at the discrete level. Extensive literature exists on gradient stable schemes for the AC equation. In this instance, we utilize Kahan's method [25], which is an advanced numerical method designed to enhance the accuracy of time integration for dynamic systems. The Kahan time integrator achieves higher precision compared to traditional integration methods, such as the standard Runge–Kutta or Euler methods. It is a second-order, time-reversal, and linearly implicit technique for ODEs [25].

For the full discrete system, we partition the time interval $[0, T]$ into $n - 1$ equal parts with time step-size $\Delta t = T/(n - 1)$ to create discrete time points $t_j = j\Delta t$, $j = 0, \dots, n - 1$. The solution vector at time t_j is denoted by $\mathbf{u}^j = \mathbf{u}(t_j) \in \mathbb{R}^m$. Then, the application of Kahan's method leads to the full discrete problem: given $\mathbf{u}^0 \in \mathbb{R}^m$ by the initial condition, we compute $\mathbf{u}^{j+1}$ from the linear system

$$\left(I_m - \frac{\Delta t}{2} J_r(\mathbf{u}^j)\right)(\mathbf{u}^{j+1} - \mathbf{u}^j) = \Delta t r(\mathbf{u}^j), \quad j = 0, 1, \dots, n - 1, \quad (5)$$

where $r(\mathbf{u}) := \mu^2 A \mathbf{u} - \mathbf{u} + \mathbf{u}^3$ is the right-hand side vector of the dynamical system (4), and $J_r(\mathbf{u})$ is its diagonal Jacobian matrix. Under the given discretization, the discrete form of the energy $\mathcal{E}$ reads as

$$E(\mathbf{u}) = \sum_{i=1}^m \left[\frac{\mu^2}{2} (D_f \mathbf{u})_i^2 + \frac{1}{2} (u_i^2 - 1)^2 \right],$$

where the matrix $D_f \in \mathbb{R}^{m \times m}$ mimics the first-order forward finite difference differentiation.

3. Reduced-Order Model

In this part, we will cover the linear/nonlinear ROM formulation of the AC equation. First, we briefly outline the conventional PCA method, followed by an explanation of KPCA, the nonlinear counterpart of the PCA. The process involves two steps, starting with vectors from the column space of a data matrix, known as the input space, moving to the reduced space via projection, and then mapping back from the reduced space to the input

space to get the reduced approximation. Because the ROM approach we are examining relies on the given data, we will focus in the following sections on the data matrix

$$U = [\mathbf{u}^1 \cdots \mathbf{u}^n] \in \mathbb{R}^{m \times n}, \quad (6)$$

where the j th column vector $\mathbf{u}^j \in \mathbb{R}^m$ represents the (full-order) solution vector of the AC equation at time t_j , derived from the full discrete system (5). We make a point to simplify notation by beginning the initial superscript of the snapshot vectors in Equation (6) at 1 instead of 0. This means that the column vector $\mathbf{u}^i \in \mathbb{R}^m$ in Equation (6) corresponds to the solution vector $\mathbf{u}^{i-1} \in \mathbb{R}^m$, $i = 1, \dots, n$, of the AC equation derived from the system (5).

3.1. Linear Dimension Reduction (PCA)

In the case of a data matrix (6) containing the solution vectors for the AC equation, PCA aims to find a linear model of a reduced dimension $k \ll m$, which can accurately represent the variance in the vectors $\mathbf{u}^i$. We assume that the matrix U has zero column sum; otherwise, we can achieve this by subtracting the average column value $\bar{\mathbf{u}} = (1/n) \sum \mathbf{u}^i$ from each column. The covariance matrix C of the matrix U is defined by

$$C = \sum_{i=1}^n \mathbf{u}^i (\mathbf{u}^i)^T = UU^T \in \mathbb{R}^{m \times m},$$

whose diagonalization is given by

$$C = P\Lambda P^T,$$

where the column vectors of the orthogonal matrix $P = [\mathbf{p}^1 \cdots \mathbf{p}^m] \in \mathbb{R}^{m \times m}$ represent the eigenvectors related to the (sorted) eigenvalues $\lambda_1 > \cdots > \lambda_m \geq 0$ lined up on the diagonal entries of the diagonal matrix $\Lambda \in \mathbb{R}^{m \times m}$. Next, the eigenvectors $\{\mathbf{p}^1, \dots, \mathbf{p}^k\} \subset \mathbb{R}^m$ of the covariance matrix C with the k largest eigenvalues $\lambda_1 > \cdots > \lambda_k$ are considered as the basis of the k -dimensional linear subspace in a straightforward manner [18]. Ultimately, any random vector $\mathbf{u}^* \in \mathbb{R}^m$ in the input space can be roughly expressed by the pre-image $\hat{\mathbf{u}}^* \in \mathbb{R}^m$ as a linear combination of the eigenvectors

$$\mathbf{u}^* \approx \hat{\mathbf{u}}^* = \sum_{i=1}^k z_i^* \mathbf{u}^i = P_k \mathbf{z}^*, \quad (7)$$

with the matrix $P_k = [\mathbf{p}^1 \cdots \mathbf{p}^k] \in \mathbb{R}^{m \times k}$ being composed of the initial k columns of the orthogonal matrix P . The real coefficients z_i^* represent the components of the projection vector $\mathbf{z}^* = [z_1^*, \dots, z_k^*] \in \mathbb{R}^k$, obtained by projecting $\mathbf{u}^*$ onto the smaller linear subspace. The $k \ll m$ -dimensional vector $\mathbf{z}^* = P_k^T \mathbf{u}^*$ lies in the reduced space, which is the projection of the m -dimensional vector $\mathbf{u}^*$ from the input space. For more details on PCA, we refer the reader to the works [9,10,12–14,20,21,26] and references therein.

3.2. Nonlinear Dimension Reduction (KPCA)

PCA is restricted to reducing dimensionality in a linear manner. However, standard PCA may be inefficient when confronted with data featuring intricate structures that cannot be accurately represented within a linear subspace. Yet, KPCA provides a resolution by allowing us to expand the linear PCA for nonlinear dimensionality reduction [12,14,16].

By utilizing KPCA, one maps the vectors from the m -dimensional input space to a larger $M \gg m$ -dimensional (potentially infinite-dimensional) space known as the feature space, through a nonlinear map $\Phi(\cdot) : \mathbb{R}^m \mapsto \mathbb{R}^M$. Then, the traditional PCA is subsequently implemented on the vectors within this feature space. In order to achieve this, we label as $\tilde{U}$ the converted data matrix created as

$$\tilde{U} = [\Phi(\mathbf{u}^1) \cdots \Phi(\mathbf{u}^n)] \in \mathbb{R}^{M \times n},$$

where the columns consist of the converted vectors $\Phi(\mathbf{u}^l) \in \mathbb{R}^M$ corresponding to the input space vectors $\mathbf{u}^l$. In most cases, the matrix $\tilde{U}$ associated with the arbitrary map $\Phi(\cdot)$ does not necessarily possess zero column sum for PCA. Subtracting the mean $\bar{\Phi} = (1/n) \sum \Phi(\mathbf{u}^i)$ from each column results in a data matrix

$$\tilde{U} = [\tilde{\Phi}(\mathbf{u}^1) \cdots \tilde{\Phi}(\mathbf{u}^n)] = \tilde{U}H \in \mathbb{R}^{M \times n},$$

with zero column sum, where the columns are $\tilde{\Phi}(\mathbf{u}^l) = \Phi(\mathbf{u}^l) - \bar{\Phi}$, and H is the centering matrix defined by

$$H = I_n - \frac{1}{n} \mathbf{1}\mathbf{1}^T \in \mathbb{R}^{n \times n}, \quad (8)$$

where I_n is the n -dimensional identity matrix and $\mathbf{1} = [1, \dots, 1]^T \in \mathbb{R}^n$ is the vector of ones. Next, we apply the standard PCA steps described earlier to the data matrix $\tilde{U}$, which has columns that represent the feature space. This involves identifying the eigenvectors of the covariance matrix

$$\tilde{C} = \sum_{l=1}^n \tilde{\Phi}(\mathbf{u}^l) \tilde{\Phi}(\mathbf{u}^l)^T = \tilde{U} \tilde{U}^T \in \mathbb{R}^{M \times M}. \quad (9)$$

Currently, the KPCA has two significant disadvantages. In the first place, the size M could be so large that it becomes nearly impossible to calculate the eigenvectors of the M -dimensional covariance matrix $\tilde{C}$. Second, the random nonlinear function $\Phi(\cdot)$ is often inaccessible. To tackle these problems, a kernel trick is utilized [16,18]. To comprehend this technique, let us examine the eigenvalue problem

$$\tilde{C} \mathbf{v}^i = \tilde{\lambda}_i \mathbf{v}^i, \quad i = 1, \dots, M, \quad (10)$$

where $\{\tilde{\lambda}_i, \mathbf{v}^i\}$ represents the eigenpair of the covariance matrix $\tilde{C}$. It is important to understand that, by definition, the eigenvectors cover the feature space, which is equivalent to the column space of the altered data matrix $\tilde{U} = [\tilde{\Phi}(\mathbf{u}^1) \cdots \tilde{\Phi}(\mathbf{u}^n)]$. Consequently, there are real coefficients a_{ij} for each eigenvector $\mathbf{v}^i$ that satisfy the linear combination

$$\mathbf{v}^i = \sum_{j=1}^n a_{ij} \tilde{\Phi}(\mathbf{u}^j), \quad i = 1, \dots, M. \quad (11)$$

By replacing the connection (11) and the identity (9) into the Equation (10), we obtain

$$\sum_{l=1}^n \tilde{\Phi}(\mathbf{u}^l) \sum_{j=1}^n a_{ij} \tilde{\Phi}(\mathbf{u}^l)^T \tilde{\Phi}(\mathbf{u}^j) = \tilde{\lambda}_i \sum_{j=1}^n a_{ij} \tilde{\Phi}(\mathbf{u}^j).$$

All eigenvectors are within the range of the altered vectors $\{\tilde{\Phi}(\mathbf{u}^s)\}_{s=1}^n$, so for $s = 1, \dots, n$, we can examine the equivalent equations by projecting onto the vectors $\tilde{\Phi}(\mathbf{u}^s)$, resulting in

$$\sum_{l=1}^n \tilde{\Phi}(\mathbf{u}^s)^T \tilde{\Phi}(\mathbf{u}^l) \sum_{j=1}^n a_{ij} \tilde{\Phi}(\mathbf{u}^l)^T \tilde{\Phi}(\mathbf{u}^j) = \tilde{\lambda}_i \sum_{j=1}^n a_{ij} \tilde{\Phi}(\mathbf{u}^s)^T \tilde{\Phi}(\mathbf{u}^j). \quad (12)$$

At this stage, a kernel function $\kappa(\cdot, \cdot) : \mathbb{R}^m \times \mathbb{R}^m \mapsto \mathbb{R}$ is defined so that

$$\kappa(\mathbf{u}^s, \mathbf{u}^l) = \langle \Phi(\mathbf{u}^s), \Phi(\mathbf{u}^l) \rangle = \Phi(\mathbf{u}^s)^T \Phi(\mathbf{u}^l), \quad s, l = 1, \dots, n,$$

with the goal of depicting the Euclidean inner products $\Phi(\mathbf{u}^s)^T \Phi(\mathbf{u}^l)$ of non-centered transformed vectors in the feature space using input space vectors. Different types of kernel functions are utilized in various studies, including linear, polynomial, and Gaussian kernels [16,26]. On the other hand, in order to be able to construct a non-iterative solution the formula, it is crucial to take into account a kernel function that can be purely written in

terms of the distance metric $\|\cdot\|$. For this reason, we employ the Gaussian kernel defined for any vector $\mathbf{x}, \mathbf{y}$ by

$$\kappa(\mathbf{x}, \mathbf{y}) = \exp\left(-\frac{\|\mathbf{x} - \mathbf{y}\|^2}{2\sigma^2}\right),$$

where $\|\cdot\|$ represents the Euclidean norm (distance metric) and σ is a parameter. The Gaussian kernel, also known as the radial basis function kernel, is the most used kernel function among the others, because of its nice properties such as smoothness and easy computation. During the simulations, we take the value of the parameter σ so that we have $2\sigma^2 = 1$. Indeed, any positive value of the parameter σ can be taken, since the results are invariant under transformation [27–29]. This invariance is also demonstrated numerically for the one-dimensional example in the numerical results section.

In addition, to depict the Euclidean inner products $\tilde{\Phi}(\cdot)^T \tilde{\Phi}(\cdot)$ of transformed vectors that are centered in the feature space, we utilize the framework [14]

$$\tilde{\kappa}(\mathbf{u}^s, \mathbf{u}^l) = \kappa(\mathbf{u}^s, \mathbf{u}^l) - \frac{1}{n} \mathbf{1}^T \mathbf{k}_{\mathbf{u}^s} - \frac{1}{n} \mathbf{1}^T \mathbf{k}_{\mathbf{u}^l} + \frac{1}{n^2} \mathbf{1}^T \mathbf{K} \mathbf{1},$$

where the kernel matrix $\mathbf{K} \in \mathbb{R}^{n \times n}$ and the vector $\mathbf{k}_{\mathbf{u}} \in \mathbb{R}^n$ are defined as

$$K_{ij} = \kappa(\mathbf{u}^i, \mathbf{u}^j), \quad \mathbf{k}_{\mathbf{u}} = (\kappa(\mathbf{u}, \mathbf{u}^1), \dots, \kappa(\mathbf{u}, \mathbf{u}^n))^T.$$

We note that it is not necessary to compute all the vectors $\mathbf{k}_{\mathbf{u}^i}$, as they are simply the i th columns of the symmetric kernel matrix $\mathbf{K}$. The use of the kernel function results in an equivalent equation to (12) being

$$\sum_{l=1}^n \kappa(\mathbf{u}^s, \mathbf{u}^l) \sum_{j=1}^n a_{ij} \kappa(\mathbf{u}^l, \mathbf{u}^j) = \tilde{\lambda}_i \sum_{j=1}^n a_{ij} \kappa(\mathbf{u}^s, \mathbf{u}^j). \quad (13)$$

The Equation (13) can be represented in matrix-vector form as

$$\tilde{\mathbf{K}}^2 \mathbf{a}^i = \tilde{\lambda}_i \tilde{\mathbf{K}} \mathbf{a}^i, \quad \text{or} \quad \tilde{\mathbf{K}} \mathbf{a}^i = \tilde{\lambda}_i \mathbf{a}^i, \quad i = 1, \dots, M, \quad (14)$$

for the coefficient vector $\mathbf{a}^i = (a_{i1}, \dots, a_{in})^T \in \mathbb{R}^n$, where $\tilde{\mathbf{K}} = \mathbf{H} \mathbf{K} \mathbf{H}$ and $\mathbf{H}$ is the centering matrix given in (8). We note here that the Gaussian kernel leads to a kernel matrix $\mathbf{K}$ being symmetric and positive definite. As a result, it is guaranteed that the eigenvalue problem (14) has stable solutions with positive eigenvalues. We also note that since we use only the first $k \ll m \ll M$ eigenvectors related to the first k dominant eigenvalue, no eigenvalue clustering or near-degeneracy issues occur. In simulations, we compute the eigenvalues using the MatLab's built-in function *eig*.

Ultimately, for any vector $\mathbf{u}^* \in \mathbb{R}^m$ in the input space, its transformed vector $\tilde{\Phi}(\mathbf{u}^*) \in \mathbb{R}^M$ in the feature space is found. The projection vector $\mathbf{z}^* = (z_1^*, \dots, z_k^*)^T \in \mathbb{R}^k$ represents the projection of $\tilde{\Phi}(\mathbf{u}^*)$ onto the reduced k -dimensional space ($k \ll m \ll M$) spanned by the eigenvectors $\{\mathbf{v}^1, \dots, \mathbf{v}^k\}$ associated with the largest k eigenvalues $\{\tilde{\lambda}_1, \dots, \tilde{\lambda}_k\}$. After calculating the coefficients $\mathbf{a}^i$ from the eigenvalue problem (14) and utilizing the identity (11) of the eigenvectors $\mathbf{v}^i$, the projection vectors' entries z_i^* can be determined using the kernel function as the following

$$z_i^* = \tilde{\Phi}(\mathbf{u}^*)^T \mathbf{v}^i = \sum_{j=1}^n a_{ij} \tilde{\kappa}(\mathbf{u}^*, \mathbf{u}^j), \quad i = 1, \dots, k.$$

Reconstructing the original image of any random vector $\mathbf{u}^* \in \mathbb{R}^m$ in the input space can be achieved by approximating its pre-image $\hat{\mathbf{u}}^* \in \mathbb{R}^m$ using standard PCA as shown in Equation (7). Nonetheless, KPCA does not follow this pattern. We denote $P_k \tilde{\Phi}(\mathbf{u}^*) \in \mathbb{R}^k$ as

the projection of $\Phi(\mathbf{u}^*)$ onto the reduced space spanned by the eigenvectors $\{v^1, \dots, v^k\}$, namely,

$$P_k \Phi(\mathbf{u}^*) = \sum_{i=1}^k z_i^* v^i + \bar{\Phi}.$$

Next, we can find an estimated pre-image $\hat{\mathbf{u}}^*$ such that the vector $\Phi(\hat{\mathbf{u}}^*)$ after transformation is the vector closest to the projected vector $P_k \Phi(\mathbf{u}^*)$. This involves determining the minimum of the objective functional $\rho(\hat{\mathbf{u}}^*) = \|\Phi(\hat{\mathbf{u}}^*) - P_k \Phi(\mathbf{u}^*)\|^2$ using a least-squares approach. By imposing $\nabla_{\hat{\mathbf{u}}^*} \rho = 0$, we obtain the following equation to be solved

$$\hat{\mathbf{u}}^* = \frac{\sum_{i=1}^k \tilde{\gamma}_i \exp\left(-\frac{\|\hat{\mathbf{u}}^* - \mathbf{u}^i\|^2}{2\sigma^2}\right) \mathbf{u}^i}{\sum_{i=1}^k \tilde{\gamma}_i \exp\left(-\frac{\|\hat{\mathbf{u}}^* - \mathbf{u}^i\|^2}{2\sigma^2}\right)}, \quad (15)$$

where we set

$$\tilde{\gamma}_i = \gamma_i + \frac{1}{n} \left(1 - \sum_{j=1}^n \gamma_j\right), \quad \gamma_i = \sum_{l=1}^k z_l^* a_{li}.$$

The implicit Equation (15) can be solved by fixed-point iteration or other Newton-type nonlinear iterative solvers. But the nonlinear iteration technique can be unstable and depends highly on the initial estimation [14,26]. In the KPCA method presented in this study, the pre-image is calculated using a non-iterative approach [14], which relies on the correlation between the distance of vectors in the input space and the distance of the vectors in the transformed space, employing the Gaussian kernel function as a distance metric function $\|\cdot\|$. By utilizing the MDS concept [14,20], we can roughly discover a pre-image $\hat{\mathbf{u}}^*$ where the differences in distances between $\mathbf{u}^*$ and each input vector $\mathbf{u}^i$ and between projected vector $P_k \Phi(\mathbf{u}^*)$ and each transformed feature vector $\Phi(\mathbf{u}^i)$ remain constant; see Figure 1. To achieve this goal, let

$$d_{ij}^2 := d^2(\mathbf{u}^i, \mathbf{u}^j) = \|\mathbf{u}^i - \mathbf{u}^j\|^2, \quad \tilde{d}_{ij}^2 := \tilde{d}^2(\Phi(\mathbf{u}^i), \Phi(\mathbf{u}^j)) = \|\Phi(\mathbf{u}^i) - \Phi(\mathbf{u}^j)\|^2,$$

represent the distance metrics between input space vectors and their transformed feature space vectors. After manipulating the Gaussian kernel function as $f(d_{ij}^2) = \exp(-d^2(\mathbf{u}^i, \mathbf{u}^j)/2\sigma^2)$, we can establish a relation between input space distance and feature space distance, and also its inverse map, as [14]

$$f(d_{ij}^2) = \frac{1}{2}(K_{ii} + K_{jj} - \tilde{d}_{ij}^2), \quad d_{ij}^2 = -2\sigma^2 \ln \left(\frac{K_{ii} + K_{jj} - \tilde{d}_{ij}^2}{2} \right). \quad (16)$$

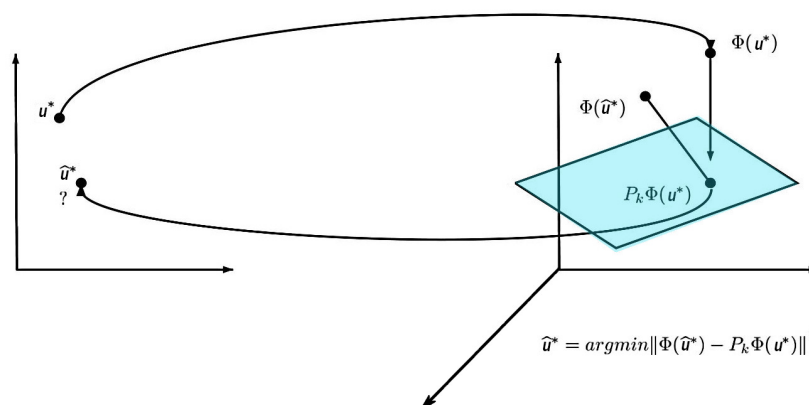


Figure 1. Reconstruction with KPCA.

In addition, it can be demonstrated using the kernel matrix that the distance in feature space between the projected vector $P_k\Phi(\mathbf{u}^*)$ and a transformed vector $\Phi(\mathbf{u}^i)$ is

$$\begin{aligned}\tilde{d}^2(\Phi(\mathbf{u}^i), P_k\Phi(\mathbf{u}^*)) &= \|\Phi(\mathbf{u}^i) - P_k\Phi(\mathbf{u}^*)\|^2 \\ &= \|\Phi(\mathbf{u}^i)\|^2 + \|P_k\Phi(\mathbf{u}^*)\|^2 - 2P_k\Phi(\mathbf{u}^*)^T\Phi(\mathbf{u}^i) \\ &= \left(k_{u^*} + \frac{1}{n}K\mathbf{1} - 2k_{u^i}\right)H^TC_aH\left(k_{u^*} - \frac{1}{n}K\mathbf{1}\right) + \frac{1}{n^2}\mathbf{1}^TK\mathbf{1} + K_{ii} - \frac{2}{n}\mathbf{1}^Tk_{u^*},\end{aligned}\quad (17)$$

for the matrix

$$C_a = \sum_{j=1}^k \frac{1}{\tilde{\lambda}_j} \mathbf{a}^j (\mathbf{a}^j)^T.$$

Ultimately, by taking $P_k\Phi(\mathbf{u}^*) \approx \Phi(\hat{\mathbf{u}}^*)$ and utilizing the metric relation (16) within Equation (15), we can derive the non-iterative solution formula [14]

$$\begin{aligned}\hat{\mathbf{u}}^* &= \frac{\sum_{i=1}^k \tilde{\gamma}_i \exp\left(-\frac{\|\hat{\mathbf{u}}^* - \mathbf{u}^i\|^2}{2\sigma^2}\right) \mathbf{u}^i}{\sum_{i=1}^k \tilde{\gamma}_i \exp\left(-\frac{\|\hat{\mathbf{u}}^* - \mathbf{u}^i\|^2}{2\sigma^2}\right)} \\ &= \frac{\sum_{i=1}^k \tilde{\gamma}_i \left(\frac{1}{2}(2 - \tilde{d}^2(\Phi(\hat{\mathbf{u}}^*), \Phi(\mathbf{u}^i)))\right) \mathbf{u}^i}{\sum_{i=1}^k \tilde{\gamma}_i \left(\frac{1}{2}(2 - \tilde{d}^2(\Phi(\hat{\mathbf{u}}^*), \Phi(\mathbf{u}^i)))\right)} \\ &\approx \frac{\sum_{i=1}^k \tilde{\gamma}_i \left(\frac{1}{2}(2 - \tilde{d}^2(P_k\Phi(\mathbf{u}^*), \Phi(\mathbf{u}^i)))\right) \mathbf{u}^i}{\sum_{i=1}^k \tilde{\gamma}_i \left(\frac{1}{2}(2 - \tilde{d}^2(P_k\Phi(\mathbf{u}^*), \Phi(\mathbf{u}^i)))\right)},\end{aligned}\quad (18)$$

where the distance $\|P_k\Phi(\mathbf{u}^*) - \Phi(\mathbf{u}^i)\|^2$ in the formula is determined through relation (17). Furthermore, the k vectors $\mathbf{u}^i$ mentioned in (18) are chosen from the n input space vectors that are closest to the given vector $\mathbf{u}^*$ [14,26].

4. Numerical Results

In this section, we investigate how well the KPCA nonlinear dimensionality reduction method performs on one-, two- and three-dimensional AC equations. In both examples, we construct the data matrix U whose columns are formed by the discrete solution vectors of the given AC equation, obtained by the full discrete system discussed in Section 2. Since all the norms are equivalent on finite-dimensional vector spaces, in order to measure the accuracy of a reduced approximation $\hat{\mathbf{u}}^*$ to a fixed full solution $\mathbf{u}^*$, we prefer to use the relative absolute error defined by

$$\|\mathbf{u}^* - \hat{\mathbf{u}}^*\|_R = \frac{\|\mathbf{u}^* - \hat{\mathbf{u}}^*\|_\infty}{\|\mathbf{u}^*\|_\infty}, \quad \|\mathbf{u}\|_\infty = \max_{1 \leq i \leq m} |u_i|,$$

which is more convenient for solutions in the form of the data vector/matrix.

4.1. One-Dimensional Problem

The first numerical example is reserved for all the relevant tests, and some of them are not shown for the remaining examples, as they are similar to each other. As the first example, the AC Equation (1) will be numerically modeled over the one-dimensional domain $\Omega = [0, 2\pi]$ with periodic boundary conditions and with the target time $T = 600$. The initial phase is

$$u(x, 0) = 0.8 + \sin(x).$$

We first begin by validating the numerical convergence of the space-time discretization methods used for the FOM (5) presented in Section 2. Since the given AC equation has no analytic solution, as the true solution, we take a reference solution matrix U_{ref} obtained through sufficiently fine spatial and temporal mesh sizes. For the reference solutions, we

simulate the given one-dimensional AC equation with the spatial mesh size $\Delta x \approx 0.0005$ ($m = 12,800$) and the temporal mesh size $\Delta t \approx 0.0078$ ($n = 76,800$). In order to check the convergence, we compute the Frobenius norm of the difference between the matrix $U^{(s)}$ of the computed solutions on the s -th spatio-temporal mesh and the projection of the matrix U_{ref} of reference solutions onto the current spatio-temporal mesh, for $s = 1, \dots, 6$. In each iteration, the mesh sizes of the current mesh are halved as $\Delta t = 2^{1-s}$ and $\Delta x = 6\Delta t$ so that the number of grid points in the space dimension is $m = 100 * 2^{s-1}$. We give the errors in Frobenius norm and the related convergence orders in Table 1. We further compute the errors and convergence orders for different values of inter-facial width parameters; we consider the values $\mu \in \{1, 0.5, 0.1, 0.05, 0.01\}$. The results validate numerically that not only both the spatial and temporal methods are second-order convergent as expected but also the numerical scheme is robust under the inter-facial width parameter μ .

Table 1. One-dimensional AC: Convergence orders for $\Delta t = 2^{1-s}$, $\Delta x = 6\Delta t$ and different values of μ .

	s	1	2	3	4	5	6
$\mu = 1$	Error	4.13×10^{-3}	1.25×10^{-3}	3.50×10^{-4}	9.23×10^{-5}	2.32×10^{-5}	5.54×10^{-6}
	Order	-	1.72	1.84	1.92	1.99	2.06
$\mu = 0.5$	Error	3.65×10^{-3}	8.95×10^{-4}	2.31×10^{-4}	5.90×10^{-5}	1.48×10^{-5}	3.54×10^{-6}
	Order	-	2.03	1.96	1.97	2.00	2.06
$\mu = 0.1$	Error	4.37×10^{-3}	9.91×10^{-4}	2.49×10^{-4}	6.27×10^{-5}	1.55×10^{-5}	3.70×10^{-6}
	Order	-	2.14	1.99	1.99	2.01	2.07
$\mu = 0.05$	Error	6.60×10^{-2}	2.31×10^{-3}	5.49×10^{-4}	1.36×10^{-4}	3.36×10^{-5}	7.99×10^{-6}
	Order	-	4.84	2.07	2.01	2.02	2.07
$\mu = 0.01$	Error	2.86×10^{-2}	1.42×10^{-2}	4.92×10^{-3}	1.35×10^{-3}	2.75×10^{-4}	6.46×10^{-5}
	Order	-	1.01	1.53	1.87	2.29	2.09

Next, the data matrix U to be used in ROM is created using discrete solutions at each time t_k taken from the instant information of the system with space-time discretization, with the mesh sizes $\Delta x = \pi/100$ and $\Delta t = 0.5$ [6]. The inter-facial width is fixed as $\mu = 0.16$. The discretized data matrix for space and time coordinates is written as $U = [u^1 \dots u^{1200}] \in \mathbb{R}^{200 \times 1200}$. The cpu time needed for solving the FOM is recorded as 6.15 s.

Figure 2 displays the plots of the precise solutions at $t = 0, 250, 400, 500$, and also the solution trajectory for the time span $[0, 600]$. The AC equation with the potential function $F(u) = (u^2 - 1)^2/4$ has one stable ($u = 0$) and two unstable ($u = \pm 1$) equilibrium points. This phenomenon, known as phase separation, involves solutions transitioning between the equilibrium points. The boundaries separating the two unstable equilibrium zones shift across areas over extended periods, known as meta-stability. Figure 2, left, shows this state of meta-stability.

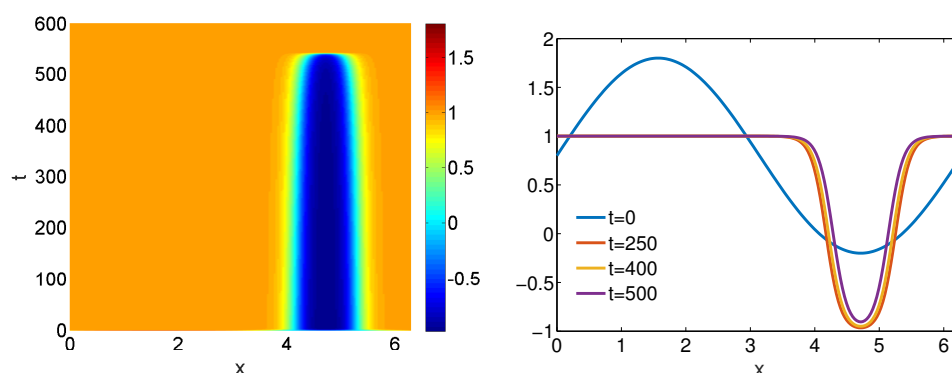


Figure 2. One-dimensional AC: Phase trajectory (left) and some phase plots (right).

Table 2 compares the PCA and KPCA methods in terms of accuracy and computational efficiency. For this comparison, the input space solution vector $u^* = u^{100}$ corresponding to the solution at time $t = 50$ is used. Results are presented for reduced dimensions $k = 1, \dots, 6$. The relative absolute errors between the full solution u^* and the reduced approximate solution $\hat{u}^*$ are shown when PCA and KPCA are used. When using the same number of reduced bases, more accurate results are obtained with KPCA.

Table 2. One-dimensional AC: Relative absolute errors $\|u^* - \hat{u}^*\|_R$ for different values of k .

k	1	2	3	4	5	6
PCA	3.09×10^{-1}	1.28×10^{-1}	1.50×10^{-2}	1.55×10^{-2}	2.34×10^{-3}	2.41×10^{-3}
KPCA	2.56×10^{-4}	1.71×10^{-5}	1.88×10^{-5}	2.28×10^{-4}	2.71×10^{-4}	5.14×10^{-4}

On the other hand, in terms of computational efficiency, Table 3 shows the solution times required by the KPCA method using Formula (15), which is solved by fixed point iteration, and the method using the algebraic Equation (18). The first two columns show the errors between the full and reduced solutions obtained with both formulae, while the last two columns show the processing time required to generate the preliminary images. It can be seen that the errors obtained with the solutions of the non-linear equation solved with iterations (2–3 iterations) the Formula (15) and without iterations with Formula (18) are almost the same, which shows that the solutions obtained without iteration are acceptable in terms of precision. It can be seen in the last two columns that the solution time required by the KPCA method without iterations is much less than the solution time required with iterations; i.e., the KPCA method with a non-iterative Formula (18) is quite fast.

Table 3. One-dimensional AC: Relative absolute errors and CPU times for different values of k .

k	$\ u^* - \hat{u}^*\ _R$		Cpu Time (s)	
	With Iteration	Without Iteration	With Iteration	Without Iteration
1	2.56×10^{-4}	2.56×10^{-4}	3.7850	0.3690
2	1.71×10^{-5}	1.28×10^{-4}	3.9650	0.3800
3	1.88×10^{-5}	2.56×10^{-4}	4.2110	0.4115
4	2.28×10^{-4}	1.29×10^{-4}	4.0152	0.4025
5	2.71×10^{-4}	1.09×10^{-3}	4.6500	0.4780
6	5.14×10^{-4}	4.92×10^{-3}	4.7968	0.4975

Figure 3 shows the reduced solutions together with the full solutions at times $t = 250$ and $t = 500$. For the reduced dimension $k = 2$, it can be seen that the full and reduced solutions coincide at both times. The full and reduced solution profiles are given in Figure 4. From the figures, it is observed that the full and reduced solutions show the same phase transition behavior.

The energy graphs obtained with full solutions and preliminary images are given in Figure 5. It can be seen that the same energy-reduction behavior is observed in both graphs.

The final test in this example is to check the effect of the scale parameter σ inside the Gaussian kernel function. Figure 6 shows, for different reduced space dimensions k , the relative absolute errors between the full solution u^* and the reduced approximation $\hat{u}^*$, when the value of the parameter σ varies in the range $[0.4, 5]$. We see that the scale parameter σ has no effect on the accuracy of the results, as we discussed in Section 3.2.

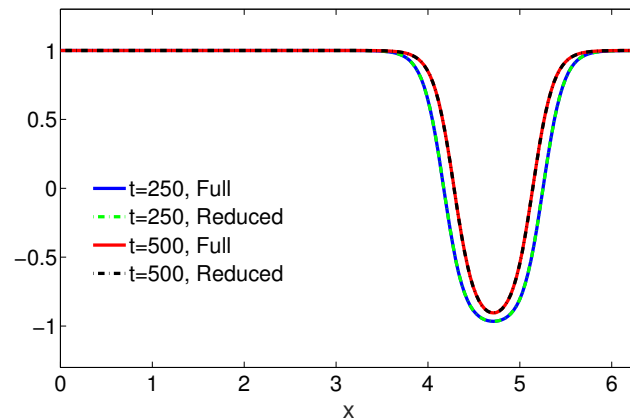


Figure 3. One-dimensional AC: Full/reduced-order solution profiles at $t = 250, 500$ for $k = 2$.

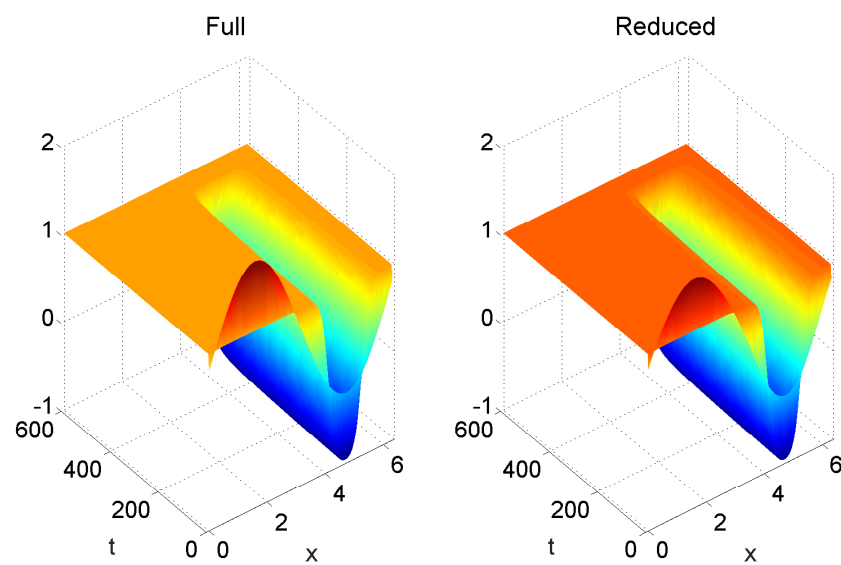


Figure 4. One-dimensional AC: Full/reduced phase trajectories for $k = 2$.

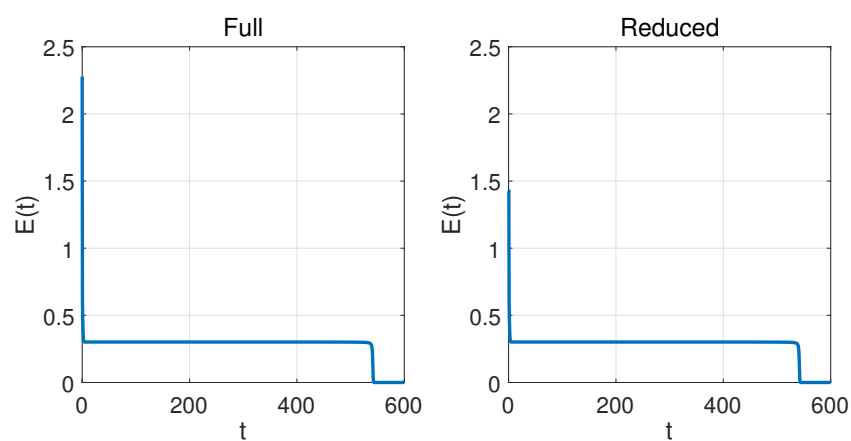


Figure 5. One-dimensional AC: Energy plots by full/reduced order solutions for $k = 2$.

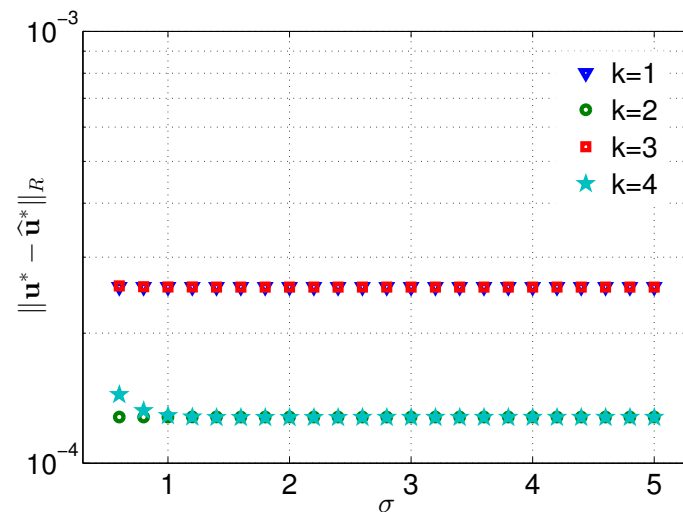


Figure 6. One-dimensional AC: Effect of scale parameter σ of Gaussian kernel for different values of k .

4.2. Two-Dimensional Problem

The AC equation, Equation (1), will be numerically modeled over the two-dimensional domain $\Omega = [0, 2\pi]^2$ with periodic boundary conditions and with the target time $T = 5$. The inter-facial width is set as $\mu = 0.05$, and the initial condition is

$$u(x_1, x_2, 0) = 2e^{\sin(x_1) + \sin(x_2) - 2} + 2.2e^{-\sin(x_1) - \sin(x_2) - 2} - 1.$$

Data matrix U is created using discrete solutions with the mesh sizes $\Delta x_1 = \Delta x_2 = \pi/32$ and $\Delta t = 0.01$ [6]. The discretized data matrix for this setting is formed as $U = [u^1 \dots u^{500}] \in \mathbb{R}^{4096 \times 500}$. The cpu time needed for solving the FOM is recorded as 14.32 s.

Table 4 compares the PCA and KPCA methods in terms of accuracy and computational efficiency. For this comparison, the solution vector $u^* = u^{100}$ input space corresponding to the solution at time $t = 1$ is used. Results are presented for reduced dimensions $k = 1, \dots, 6$. The relative absolute errors between the exact solution u^* and the reduced approximate solution $\hat{u}^*$ are shown when PCA and KPCA are used. When using the same number of bases, more accurate results are obtained with KPCA.

Table 4. Two-dimensional AC: Relative absolute errors $\|u^* - \hat{u}^*\|_R$ for different values of k .

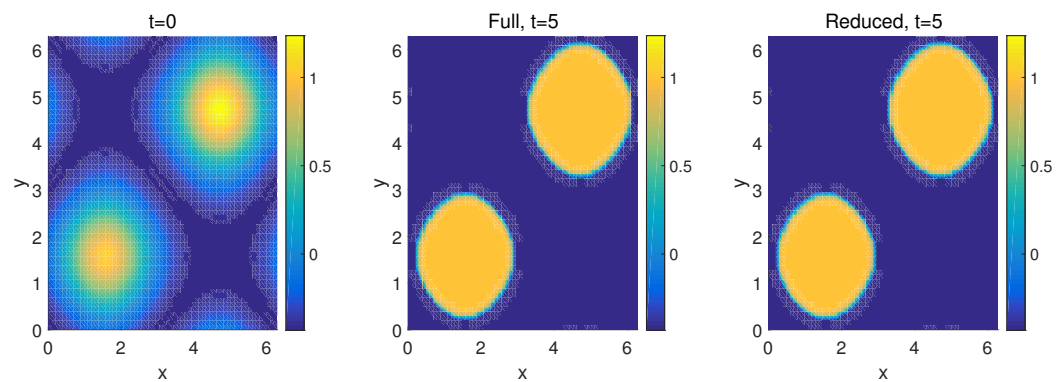
k	1	2	3	4	5	6
PCA	2.80×10^{-1}	8.22×10^{-2}	4.78×10^{-2}	2.13×10^{-2}	6.53×10^{-3}	3.54×10^{-3}
KPCA	2.50×10^{-3}	1.49×10^{-3}	3.28×10^{-3}	1.92×10^{-3}	2.45×10^{-3}	1.37×10^{-3}

In terms of computational efficiency, Table 5 shows the solution times required by the KPCA method using Formula (15) solved by fixed point iteration and the method using the algebraic Equation (18). The first two columns show the errors between the full and reduced solutions obtained with both formulas, while the last two columns show the processing time required to generate the preliminary images. It can be seen that the errors obtained with the solutions of the non-linear equation solved with iterations (three iterations) and without iterations are almost the same, which shows that the solutions obtained without iterations are acceptable in terms of precision. It can be seen in the last two columns that the solution time required by the KPCA method without iterations is considerably less than the solution time required with iteration.

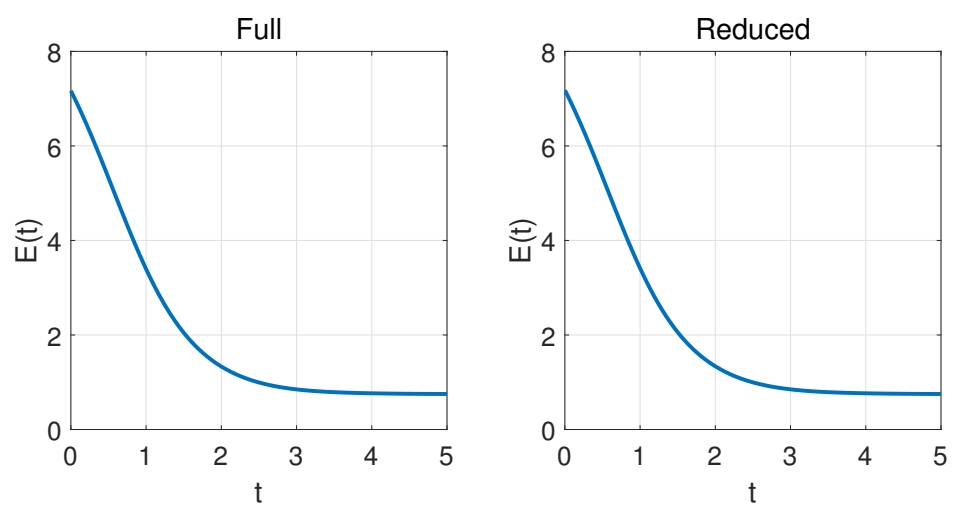
Table 5. Two-dimensional AC: Relative absolute errors and CPU times for different values of k .

k	$\ u^* - \hat{u}^*\ _R$		Cpu Time (s)	
	With Iteration	Without Iteration	With Iteration	Without Iteration
1	2.50×10^{-3}	3.80×10^{-3}	12.2090	1.4310
2	1.49×10^{-3}	1.89×10^{-3}	11.7000	1.6470
3	3.28×10^{-3}	3.77×10^{-3}	12.2340	1.5070
4	1.92×10^{-3}	1.82×10^{-3}	12.6700	1.7160
5	2.45×10^{-3}	3.66×10^{-3}	13.2030	1.8110
6	1.37×10^{-3}	1.68×10^{-3}	13.5700	1.8230

Figure 7 shows the initial profile and profiles of the reduced solution together with the full solution at the final time $t = 5$. For the reduced dimension $k = 2$, the full and reduced solutions are similar. This means that the reduced solutions have the same behavior as the full one.

**Figure 7.** Two-dimensional AC: Full/reduced solution profiles for $k = 2$.

On the other hand, the energy plots obtained with full solutions and preliminary images are given in Figure 8. Here, it is clearly seen that the same energy reduction behavior occurs in both plots, similar to the one-dimensional AC equation.

**Figure 8.** Two-dimensional AC: Energy plots by full/reduced order solutions for $k = 2$.

4.3. Three-Dimensional Problem

In this last example, the scaled AC equation, Equation [6,30,31],

$$u_t = \Delta u - \frac{1}{\mu^2} f(u),$$

will be numerically modeled over the three-dimensional domain $\Omega = [0, 1]^3$ with homogeneous Neumann boundary conditions and with the target time $T = 0.035$. The inter-facial width is set as $\mu = 0.1$, and the initial condition is

$$u(x_1, x_2, x_3, 0) = \tanh\left(\frac{R_0 - \sqrt{(x_1 - 0.5)^2 + (x_2 - 0.5)^2 + (x_3 - 0.5)^2}}{\sqrt{2}\mu}\right).$$

Here, the given parameter R_0 represents the initial radius of the spherical zero isosurface, and the exact radius R of the sphere at time t follows from the formula $R(t) = \sqrt{R_0^2 - 4t}$ [30]. This indicates a decrease in the radius of the spherical zero isosurface over time. In the numerical simulation, we take the initial radius as $R_0 = 0.4$.

For the discrete solution matrix U , we apply the mesh sizes $\Delta x_1 = \Delta x_2 = \Delta x_3 = 0.05$ and $\Delta t = 0.0001$; i.e., the discretized data matrix is $U = [u^1 \dots u^{350}] \in \mathbb{R}^{9261 \times 350}$. The cpu time needed for solving the FOM is recorded as 32.27 s. Table 6 compares the PCA and KPCA methods in terms of accuracy. For this comparison, the solution vector $u^* = u^{100}$ input space corresponding to the solution at time $t = 0.01$ is used. The results are presented for reduced dimensions $k = 1, \dots, 6$. The relative absolute errors between the full solution u^* and the reduced approximate solution $\hat{u}^*$ are shown when PCA and KPCA are used. When using the same number of bases, more accurate results are obtained with KPCA.

Table 6. Three-Dimensional AC: Relative absolute errors $\|u^* - \hat{u}^*\|_R$ for different values of k .

k	1	2	3	4	5	6
PCA	7.06×10^{-2}	8.55×10^{-2}	1.38×10^{-2}	1.16×10^{-2}	2.78×10^{-3}	4.85×10^{-4}
KPCA	4.57×10^{-3}	2.25×10^{-3}	4.83×10^{-3}	2.03×10^{-3}	6.68×10^{-3}	5.37×10^{-3}

In terms of computational efficiency, Table 7 shows the solution times required by the KPCA method with and without iteration. It can be seen that the errors obtained with the solutions of the non-linear equation solved with iterations (two iterations) and without iteration are almost the same, similar to the one- and two-dimensional cases, and that the KPCA method without iteration is fastest.

Table 7. Three-dimensional AC: Relative absolute errors and CPU times for different values of k .

k	$\ u^* - \hat{u}^*\ _R$		Cpu Time (s)	
	With Iteration	Without Iteration	With Iteration	Without Iteration
1	4.57×10^{-3}	3.69×10^{-3}	12.2990	2.1780
2	2.25×10^{-3}	2.47×10^{-3}	13.4760	2.1510
3	4.83×10^{-3}	4.74×10^{-3}	14.1110	2.0270
4	2.03×10^{-3}	2.17×10^{-3}	14.1600	2.7510
5	6.68×10^{-3}	6.55×10^{-3}	15.2780	2.5950
6	5.37×10^{-3}	5.30×10^{-3}	15.1340	2.9640

In Figure 9, left, the energy plots obtained with full and reduced solutions are given. Again, it is clearly seen that the same energy reduction behavior occurs by both full and reduced solutions. Figure 10 shows the profiles of the zero isosurface by full and reduced solutions at the times $t = 0.02, 0.03, 0.035$. For the reduced dimension $k = 2$, the full and

reduced profiles are highly similar, and the radius of the spherical zero isosurface decreases as time progresses. The decrease in the exact and numerical radii is also demonstrated in Figure 9, right.

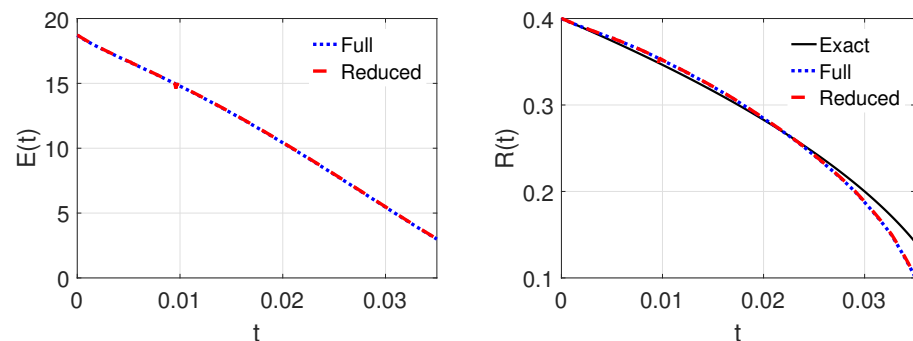


Figure 9. Three-Dimensional AC: Energy plots (left) and radii of the spherical zero isosurface (right) by full/reduced solutions for $k = 2$.

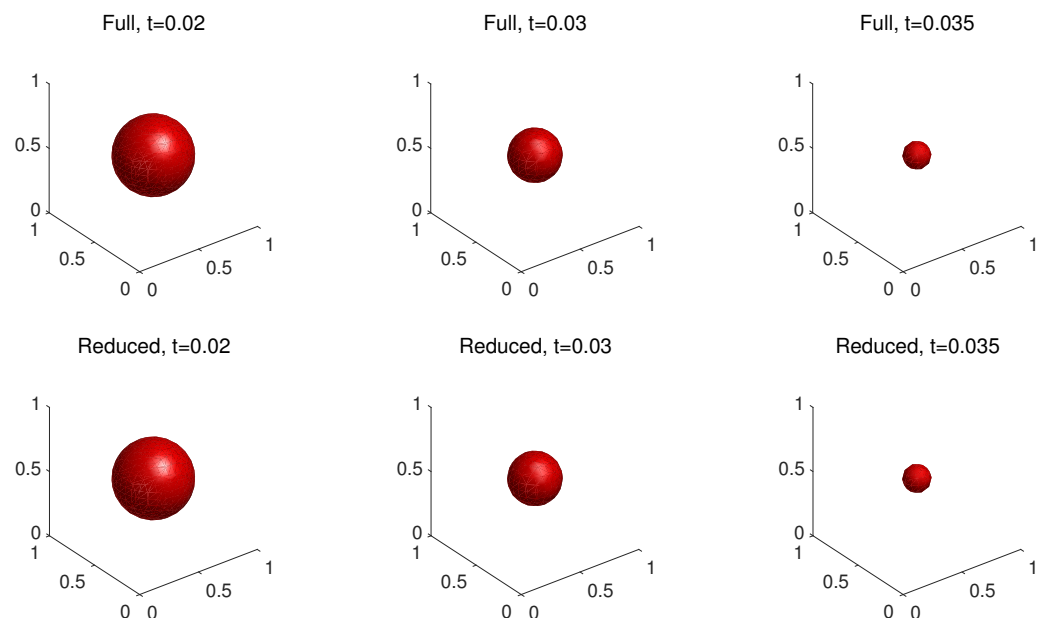


Figure 10. Three-Dimensional AC: Snapshots of the zero isosurface of full (top) and reduced (bottom) solutions for $k = 2$.

5. Discussion

In this article, we suggest a nonlinear dimensionality reduction method that upholds precision and conserves the energy decay property of the one-, two-, and three-dimensional AC equations. The method for reducing dimensionality is non-intrusive and based on the KPCA. The KPCA utilizes the MDS method and considers the input space and feature space distance metrics to develop a non-iterative algorithm for reconstructing pre-images. The numerical test examples show the accuracy of solutions and the preservation of energy dissipation structure. A comparison is made with traditional PCA and KPCA, as well as the iterative scheme with the non-iterative one. In our future research, we aim to consider stochastic AC equation [32], and explore various types of kernel functions that may offer more effective performance compared to the Gaussian kernel, for both deterministic and stochastic AC equations.

Author Contributions: Conceptualization, Y.Ç. and M.U.; methodology, Y.Ç. and M.U.; software, M.U.; validation, Y.Ç. and M.U.; formal analysis, Y.Ç. and M.U.; investigation, Y.Ç. and M.U.; resources, Y.Ç. and M.U.; data curation, M.U.; writing—original draft preparation, Y.Ç.; writing—review and editing, M.U.; visualization, Y.Ç. and M.U.; supervision, M.U.; project administration, Y.Ç. and M.U.; funding acquisition, Y.Ç. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Data Availability Statement: Dataset available on request from the authors.

Conflicts of Interest: The authors declare no conflicts of interest.

References

- Allen, S.M.; Cahn, J.W. A microscopic theory for antiphase boundary motion and its application to antiphase domain coarsening. *Acta Metall.* **1979**, *27*, 1085–1095. [\[CrossRef\]](#)
- Chen, L.Q. Phase-field models for microstructure evolution. *Annu. Rev. Mater. Res.* **2002**, *32*, 113–140. [\[CrossRef\]](#)
- Beneš, M.; Chalupecký, V.; Mikula, K. Geometrical image segmentation by the Allen-Cahn equation. *Appl. Numer. Math.* **2004**, *51*, 187–205. [\[CrossRef\]](#)
- Yang, X.; Feng, J.J.; Liu, C.; Shen, J. Numerical simulations of jet pinching-off and drop formation using an energetic variational phase-field method. *J. Comput. Phys.* **2006**, *218*, 417–428. [\[CrossRef\]](#)
- Feng, X.; Prohl, A. Numerical analysis of the Allen-Cahn equation and approximation for mean curvature flows. *Numer. Math.* **2003**, *94*, 33–65. [\[CrossRef\]](#)
- Uzunca, M.; Karasözen, B. Linearly implicit methods for Allen-Cahn equation. *Appl. Math. Comput.* **2023**, *450*, 127984. [\[CrossRef\]](#)
- Wang, J.; Shi, Z. Multi-Reconstruction from Points Cloud by Using a Modified Vector-Valued Allen-Cahn Equation. *Mathematics* **2021**, *9*, 1326. [\[CrossRef\]](#)
- Haq, M.U.; Haq, S.; Ali, I.; Ebadi, M.J. Approximate Solution of PHI-Four and Allen-Cahn Equations Using Non-Polynomial Spline Technique. *Mathematics* **2024**, *12*, 798. [\[CrossRef\]](#)
- Jackson, B.B.; Bund, B. *Multivariate Data Analysis: An Introduction*; Irwin, R.D., Ed.; ThriftBooks-Atlanta: Austell, GA, USA, 1983.
- Johnson, R.; Wichern, D. *Applied Multivariate Statistical Analysis*; Michael Bell: Englewood Cliffs, NJ, USA, 2007.
- Hached, M.; Jbilou, K.; Koukouvinos, C.; Mitrouli, M. A Multidimensional Principal Component Analysis via the C-Product Golub-Kahan-SVD for Classification and Face Recognition. *Mathematics* **2021**, *9*, 1249. [\[CrossRef\]](#)
- González, A.G.; Huerta, A.; Zlotnik, S.; Diez, P. A kernel Principal Component Analysis (kPCA) digest with a new backward mapping (pre-image reconstruction) strategy. *arXiv* **2021**. [\[CrossRef\]](#)
- Mika, S.; Schölkopf, B.; Smola, A.; Müller, K.R.; Scholz, M.; Rätsch, G. Kernel PCA and De-Noising in Feature Spaces. In *Advances in Neural Information Processing Systems*; Kearns, M., Solla, S., Cohn, D., Eds.; MIT Press: Cambridge, MA, USA, 1998; Volume 11.
- Rathi, Y.; Dambreville, S.; Tannenbaum, A. Statistical Shape Analysis using Kernel PCA. *Sch. Electr. Comput. Eng. Ga. Inst. Technol.* **2006**, *6064*, 641417. [\[CrossRef\]](#)
- Schölkopf, B.; Smola, A. *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond*; Adaptive Computation and Machine Learning; MIT Press: Cambridge, MA, USA, 2002.
- Schölkopf, B.; Smola, A.; Müller, K.R. Nonlinear Component Analysis as a Kernel Eigenvalue Problem. *Neural Comput.* **1998**, *10*, 1299–1319. [\[CrossRef\]](#)
- Shao, Z.; Xiang, P.; Zhao, H.; Zhang, P.; Xie, X.; Gan, L.; Li, W.; Yin, B.; Liew, K.M. A novel train-bridge interaction computational framework based on a meshless box girder model. *Adv. Eng. Softw.* **2024**, *192*, 103628. [\[CrossRef\]](#)
- Wang, Q. Kernel Principal Component Analysis and its Applications in Face Recognition and Active Shape Models. *arXiv* **2012**. [\[CrossRef\]](#)
- Zhang, Q.; Ying, Z.; Zhou, J.; Sun, J.; Zhang, B. Broad Learning Model with a Dual Feature Extraction Strategy for Classification. *Mathematics* **2023**, *11*, 4087. [\[CrossRef\]](#)
- Cox, T.F.; Cox, M.A.A. *Multidimensional Scaling*; Monographs on Statistics and Applied Probability; Chapman & Hall: London, UK, 2001.
- Williams, C.K.I. On a connection between kernel PCA and metric multidimensional scaling. In *Advances in Neural Information Processing Systems 13*; Leen, T., Dietterich, T., Tresp, V., Eds.; MIT Press: Cambridge, MA, USA, 2000.
- Song, H.; Jiang, L.; Li, Q. A reduced order method for Allen-Cahn equations. *J. Comput. Appl. Math.* **2016**, *292*, 213–229. [\[CrossRef\]](#)
- Kalashnikova, I.; Barone, M.F. Efficient non-linear proper orthogonal decomposition/Galerkin reduced order models with stable penalty enforcement of boundary conditions. *Int. J. Numer. Methods Eng.* **2012**, *90*, 1337–1362. [\[CrossRef\]](#)
- Uzunca, M.; Karasözen, B. Energy Stable Model Order Reduction for the Allen-Cahn Equation. In *Model Reduction of Parametrized Systems*; Benner, P., Ohlberger, M., Patera, A., Rozza, G., Urban, K., Eds.; Springer International Publishing: Cham, Switzerland, 2017; pp. 403–419. [\[CrossRef\]](#)

25. Kahan, W.; Li, R.C. Unconventional Schemes for a Class of Ordinary Differential Equations. *J. Comput. Phys.* **1997**, *134*, 316–331. [[CrossRef](#)]
26. Çakır, Y.; Uzunca, M. Nonlinear Reduced Order Modelling for Korteweg-de Vries Equation. *Int. J. Inform. Appl. Math.* **2024**, *7*, 57–72. [[CrossRef](#)]
27. Aydin, D. Alternative robust estimation methods for parameters of Gumbel distribution: An application to wind speed data with outliers. *Wind Struct.* **2018**, *26*, 383–395. [[CrossRef](#)]
28. Aydin, D.; Senoglu, B. Monte Carlo comparison of the parameter estimation methods for the two-parameter Gumbel distribution. *J. Mod. Appl. Stat. Method.* **2015**, *14*, 123–140. [[CrossRef](#)]
29. Tiku, M.L.; Akkaya, A.D. *Robust Estimation and Hypothesis Testing*; New Age International: New Delhi, India, 2004.
30. Li, Y.; Lee, H.G.; Jeong, D.; Kim, J. An unconditionally stable hybrid numerical method for solving the Allen-Cahn equation. *Comput. Math. Appl.* **2010**, *60*, 1591–1606. [[CrossRef](#)]
31. Karasözen, B.; Uzunca, M.; Sariaydin-Filibelioglu, A.; Yücel, H. Energy Stable Discontinuous Galerkin Finite Element Method for the Allen-Cahn Equation. *Int. J. Comput. Methods* **2018**, *15*, 1850013. [[CrossRef](#)]
32. Breit, D.; Prohl, A. Weak error analysis for the stochastic Allen–Cahn equation. *Stoch. Pde. Anal. Comp.* **2024**, *12*, 2181–2245. [[CrossRef](#)]

Disclaimer/Publisher’s Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.