Analytical and Numerical Solutions for Nonlinear Equations
©Available online at https://ansne.du.ac.ir/
Online ISSN: 3060–785X
2027, Volume 12, Issue 1, pp. 171–190

Research article

Self-Refining Kernel-Aware Physics-Informed Neural Network for Nonlinear Singular Integral Equations

N. Ramya[1,*], Hossein Rashmanlou[2], Afshin Fayyaz Movaghar[3], Mostafa NouriJouybari[4], Farshid Mofidnakhaei[5], Yahya Talebi[6]

[1] Department of Mathematics, Kongu Engineering College, Erode, Tamilnadu, India

[2] Canadian Quantum Research Center, 106-460 Doyle Ave, Kelowna, British Columbia V1Y 0C2, Canada

[3] Department of Statistics, University of Mazandaran, Babolsar, Iran

[4] Payame Noor University (PNU), Tehran, Iran

[5] Department of Physics, Sar.C., Islamic Azad University, Sari, Iran

[6] Department of Mathemstics, University of Mazandaran, Babolsar, Iran

*** Corresponding author(s):** jpramyamaths@gmail.com

Received: 31/08/2026 **Revised:** 16/10/2026 **Accepted:** 17/10/2026 **Published:** 02/11/2026 10.22128/ansne.2026.3452.1241

Abstract

Nonlinear weakly singular integral equations are challenging to solve numerically because the singular behaviour of the kernel can reduce the effectiveness of conventional uniform discretization and physics-informed learning strategies. This study develops a Self-Refining Kernel-Aware Physics-Informed Neural Network (SKA-PINN) for nonlinear weakly singular Fredholm and Volterra integral equations. The proposed framework combines a kernel-aware difficulty indicator based on residual magnitude, residual variation, and kernel complexity with adaptive residual weighting, self-refining collocation, and adaptive quadrature. Instead of distributing computational points uniformly, the method identifies numerically difficult regions and progressively increases the local resolution during training. The framework is evaluated using nonlinear Fredholm and Volterra benchmark problems with prescribed analytical solutions. For the Fredholm problem, the maximum reported absolute and relative errors are 6.3×10^{-6} and 3.4×10^{-6}, respectively. For the Volterra problem, the maximum absolute error is 7.5×10^{-6}, with a maximum relative error of 1.2×10^{-4}. The refinement experiments show a consistent reduction in the maximum residual and solution error as the collocation and quadrature resolutions increase. In comparison with the classical and adaptive PINN configurations considered, SKA-PINN achieves the smallest reported L^2 error of 1.09×10^{-5} while requiring fewer training epochs and lower CPU time. The ablation results further show that kernel-aware weighting and self-refinement both contribute to the overall improvement. These results indicate that incorporating kernel information directly into the learning and refinement process provides an effective computational strategy for nonlinear weakly singular integral equations.

Keywords: Physics-informed neural networks, nonlinear singular integral equations, weakly singular kernels, kernel-aware learning.

Mathematics Subject Classification (2020): 45G05, 45D05, 45B05, 65R20

COPYRIGHTS: ©2027, Analytical and Numerical Solutions for Nonlinear Equations published by Damghan University.
This is an open access article published under the *CC BY 4 DEED* license

1 Introduction

Recent developments in physics-informed neural networks (PINNs) have expanded their application to a wide range of mathematical and engineering problems. Nadal et al. integrated PINNs into power-system dynamic simulations, demonstrating their ability to incorporate physical governing equations into data-driven computational frameworks [1]. Ren et al. proposed an enhanced PINN framework for continuous integral equations, highlighting the potential of modified network architectures for improving solution accuracy [2]. Tipu et al. combined analytical techniques with PINNs to obtain data-driven soliton solutions and identify unknown parameters in the fourth-order nonlinear AKNS equation [3]. Das et al. further developed a hybrid approach by coupling PINNs with symbolic regression to derive analytical approximate solutions for nonlinear partial differential equations [4]. For challenging mathematical problems, Lee introduced a least-squares enhanced physics-informed learning strategy for singular and ill-posed partial differential equations [5]. Wang and Yang applied PINNs to the integrable fractional nonlinear Schrödinger equation, demonstrating their applicability to fractional-order nonlinear systems and associated dynamical behaviour [6]. Nishanth and Elango examined different optimization algorithms for PINNs in solving singularly perturbed differential equations and emphasized the importance of optimization strategies in obtaining reliable numerical solutions [7]. A comprehensive review by Luo et al. discussed the development, applications, advantages, and limitations of PINNs for partial differential equations, providing an overview of recent advances in the field [8]. Ding et al. incorporated Fourier features into PINNs to improve seismic wavefield simulations in complex nonsmooth media, particularly for problems involving intricate wave characteristics [9]. Mofidi et al. introduced geometric singular perturbation-informed neural networks to investigate systems characterized by multiple timescales [10]. Pratama et al. combined truncated singular value decomposition with PINNs to enhance the treatment of high-dimensional partial differential equations [11]. Agarwal and Mishra presented a comparative study of PINNs for fractional differential equations, further demonstrating the growing interest in neural-network-based approaches for fractional models [12]. Ngueabou and Oloniiju reviewed hybrid strategies that integrate spectral methods with neural-network architectures, emphasizing the benefits of combining classical numerical techniques with machine-learning approaches [13]. Sun et al. proposed ROM-PINN by integrating reduced-order modelling with PINNs for nonlinear structural-response prediction, with the aim of improving computational efficiency for complex systems [14]. More recently, Singh et al. extended the PINN framework to non-local physics-informed neural networks for forward and inverse problems involving non-local operators [15]. Collectively, these studies indicate that PINNs are evolving from a general physics-constrained learning framework toward more specialized and efficient approaches capable of addressing nonlinear, fractional, singular, high-dimensional, and non-local differential equations. Ramya et al. developed ANN- and PINN-assisted analytical approaches for solving nonlinear fractional biological models involving local fractional operators, demonstrating the applicability of neural-network-based methods to complex fractional-order biological systems [16]. Muhiuddin et al. investigated hybrid SWCNT/MWCNT nanofluid flow over porous surfaces by incorporating thermal radiation, microbial bioconvection, and Soret–Dufour effects, highlighting the coupled influence of these physical mechanisms on nanofluid transport [17]. Muhiuddin et al. analyzed the effects of thermophoresis and Brownian motion on Newtonian and non-Newtonian fluid flow over a porous stretching surface, providing insights into the role of nanoparticle transport mechanisms in porous-fluid systems [18]. Fallah et al. investigated MHD nanofluid flow over a stretching surface using a fuzzy-based numerical approach, highlighting the influence of DarcyForchheimer porous medium effects on fluid flow characteristics [19].Aliyari Boroujeni et al. proposed an improved teachinglearning-based optimization (ITLBO) algorithm for solving nonlinear inverse partial differential equation problems, demonstrating the effectiveness of optimization techniques in obtaining numerical solutions [20].Boroujeni et al. developed numerical approaches for solving the Kortewegde Vries (KdV) and modified Kortewegde Vries (mKdV) equations, emphasizing sequence-based methods and multicore parallelization to improve computational efficiency [21].

1.1 Nonlinear Singular Integral Equations and Existing Solution Methods

Nonlinear singular integral equations constitute an important class of nonlocal mathematical models in which the unknown function appears inside an integral operator and the associated kernel exhibits singular or near-singular behaviour. In weakly singular problems, the kernel may become unbounded when the integration variables approach one another, while the integral remains finite. A general nonlinear weakly singular integral equation can be represented in the form

$$u(x) = f(x) + \lambda \int_{\Omega} \frac{K(x,t)N(u(t))}{|x-t|^{\alpha}}\, dt, \qquad 0 < \alpha < 1,$$

where $N(\cdot)$ denotes a nonlinear operator. Such equations are intrinsically nonlocal because the solution at a given point depends on values of the unknown function over an extended integration region.

Several numerical approaches have been developed for nonlinear integral equations, including quadrature-based discretization, collocation methods, Nyström-type methods, projection methods, iterative schemes, and spectral or polynomial approximation techniques. These methods can provide highly accurate solutions for appropriately resolved problems. However, the presence of a singular kernel introduces additional numerical difficulties. In particular, uniform discretization may provide insufficient resolution near regions where the kernel varies rapidly, and the numerical evaluation of the integral operator may become sensitive to the choice of quadrature nodes, weights, and regularization parameters. For nonlinear equations, these difficulties are further coupled with the nonlinear approximation of the unknown solution.

Physics-informed neural networks provide an alternative computational framework in which the unknown solution is represented by a trainable neural network and the governing integral equation is incorporated directly into the loss function. This approach avoids the need to construct an explicit global discretization of the unknown solution and provides a flexible mechanism for handling nonlinear and nonlocal operators. PINNs are therefore particularly attractive for integral equations in which the solution may be difficult to approximate using a fixed conventional discretization.

Nevertheless, a standard PINN does not automatically resolve the numerical difficulty created by a singular kernel. Uniformly distributed collocation points assign similar computational importance to different regions of the domain, even when the residual or kernel contribution varies substantially. Moreover, errors in the numerical evaluation of the integral operator can directly contaminate the physics-informed residual. These observations motivate the development of a kernel-aware PINN in which the local difficulty of the integral operator is explicitly incorporated into the loss weighting, collocation distribution, and quadrature resolution.

1.2 Research Gap

Recent developments in physics-informed neural networks (PINNs) have demonstrated their potential for solving nonlinear, fractional, singular, high-dimensional, and non-local differential equations. However, the existing approaches discussed in the literature mainly focus on improving network architectures, optimization procedures, feature representations, or reduced-order formulations [1–15]. These developments do not specifically address the numerical difficulties associated with nonlinear singular integral equations, where the kernel may become unbounded near the singular point. In particular, a conventional PINN generally treats collocation points uniformly and does not explicitly account for the local complexity of the integral kernel when constructing the training loss. This can lead to insufficient resolution in regions where the residual changes rapidly or where the singular kernel has a stronger influence.

The uploaded study identifies this limitation and develops a Self-Refining Kernel-Aware Physics-Informed Neural Network (SKA-PINN) for nonlinear weakly singular Fredholm and Volterra integral equations. The proposed framework combines a kernel-aware difficulty indicator based on residual magnitude, residual gradients, and kernel complexity with adaptive residual weighting. It further introduces self-refining collocation and adaptive quadrature strategies so that computational points are concentrated in difficult regions rather than distributed uniformly. Therefore, the main research gap lies in the need for a unified physics-informed framework that can simultaneously recognize kernel-induced numerical difficulty, adapt the training loss, refine collocation points, and improve the numerical integration of singular operators. The study addresses this gap by coupling residual learning, kernel-aware error estimation, adaptive weighting, collocation refinement, and quadrature refinement within a self-refining learning cycle.

1.3 Research Objectives

The main objective of this research is to develop an accurate, stable, and computationally efficient physics-informed neural network framework for nonlinear singular integral equations. The specific objectives are:

1. To formulate a physics-informed neural network for solving nonlinear weakly singular Fredholm and Volterra integral equations.
2. To develop a kernel-aware difficulty indicator that identifies computationally challenging regions using the local residual, residual gradient, and kernel complexity.
3. To introduce adaptive residual weighting so that greater importance is assigned to regions with large residuals, rapid residual variation, or strong kernel complexity.

4. To develop a self-refining collocation strategy that automatically introduces additional training points in difficult regions of the computational domain.
5. To incorporate adaptive quadrature refinement to improve the numerical evaluation of singular integral operators.
6. To integrate kernel-aware weighting, collocation refinement, and quadrature refinement into a unified self-refining PINN framework.
7. To evaluate the accuracy, convergence, stability, and computational efficiency of the proposed framework using nonlinear singular Fredholm and Volterra benchmark problems.
8. To compare the proposed method with conventional and adaptive PINN approaches in terms of prediction error, number of training epochs, and computational time.

1.4 Research Questions

The proposed research is guided by the following research questions:

1. How accurately can a physics-informed neural network solve nonlinear weakly singular Fredholm and Volterra integral equations?
2. How effectively can kernel complexity, residual magnitude, and residual variation be used to identify numerically difficult regions?
3. Does kernel-aware adaptive residual weighting improve the accuracy and convergence of PINNs compared with uniformly weighted residual losses?
4. To what extent does self-refining collocation improve prediction accuracy by concentrating computational points in difficult regions?
5. How does adaptive quadrature refinement influence the accuracy of numerical integration for singular integral operators?
6. Does the simultaneous refinement of collocation points and quadrature nodes provide greater benefits than refining either distribution independently?
7. Can the proposed SKA-PINN framework achieve lower prediction errors and faster convergence than conventional PINN and adaptive PINN approaches?
8. What is the contribution of each component of the proposed framework, namely kernel-aware weighting, adaptive collocation, and adaptive quadrature refinement, to the overall performance?

2 Mathematical Formulation

This section presents the mathematical formulation of the proposed Self-Refining Kernel-Aware Physics-Informed Neural Network (SKA-PINN) for nonlinear singular integral equations.

2.1 Nonlinear Singular Integral Equation

Consider the nonlinear weakly singular Fredholm integral equation of the second kind defined on the domain $\Omega = [a,b]$:

$$u(x) = f(x) + \lambda \int_a^b \frac{K(x,t)\mathcal{N}(u(t))}{|x-t|^{\alpha}}\, dt, \qquad x \in \Omega, \quad 0 < \alpha < 1, \tag{1}$$

where $u(x)$ denotes the unknown solution, $f(x)$ is a prescribed source function, $K(x,t)$ represents the integral kernel, λ is a coupling parameter, and $\mathcal{N}(\cdot)$ denotes a nonlinear operator. The condition $0 < \alpha < 1$ corresponds to a weakly singular but integrable kernel.

For example, by choosing

$$\mathcal{N}(u) = u^p, \qquad p > 1, \tag{2}$$

Equation (1) becomes

$$u(x) = f(x) + \lambda \int_a^b \frac{K(x,t)u^p(t)}{|x-t|^{\alpha}}\, dt. \tag{3}$$

The corresponding nonlinear Volterra integral equation can be written as

$$u(x) = f(x) + \lambda \int_a^x \frac{K(x,t)\mathcal{N}(u(t))}{(x-t)^\alpha}\, dt, \qquad 0 < \alpha < 1. \tag{4}$$

2.2 Neural Network Approximation

The unknown solution $u(x)$ is approximated by a deep neural network as

$$u(x) \approx u_\theta(x) = \mathcal{N}_\theta(x), \tag{5}$$

where θ denotes the collection of trainable network parameters.

To improve information propagation through the network, residual blocks are introduced in the hidden layers. The output of the ℓ-th residual block is expressed as

$$\mathbf{z}^{(\ell+1)} = \mathbf{z}^{(\ell)} + \mathscr{F}^{(\ell)}\left(\mathbf{z}^{(\ell)}; \theta_\ell\right), \tag{6}$$

where $\mathbf{z}^{(\ell)}$ represents the feature vector at layer ℓ and $\mathscr{F}^{(\ell)}$ denotes the nonlinear transformation performed by the corresponding residual block.

2.3 Physics-Informed Integral Residual

Substitution of the neural approximation $u_\theta(x)$ into Equation (1) gives the physics-informed integral residual

$$R_\theta(x) = u_\theta(x) - f(x) - \lambda \int_a^b \frac{K(x,t)\mathcal{N}(u_\theta(t))}{|x-t|^\alpha}\, dt. \tag{7}$$

For the exact solution,

$$R_\theta(x) = 0, \qquad \forall x \in \Omega. \tag{8}$$

2.4 Quadrature-Based Integral Evaluation

Let $\{t_j, w_j\}_{j=1}^{N_q}$ denote the quadrature nodes and corresponding weights. The integral operator in Equation (7) is approximated by

$$\mathscr{I}_\theta(x) \approx \sum_{j=1}^{N_q} w_j \frac{K(x,t_j)\mathcal{N}(u_\theta(t_j))}{|x-t_j|^\alpha + \varepsilon_s}, \tag{9}$$

where $\varepsilon_s > 0$ is a small numerical regularization parameter.

Consequently, at the collocation point x_i, the discrete integral residual becomes

$$R_\theta(x_i) = u_\theta(x_i) - f(x_i) - \lambda \sum_{j=1}^{N_q} w_j \frac{K(x_i,t_j)\mathcal{N}(u_\theta(t_j))}{|x_i-t_j|^\alpha + \varepsilon_s}. \tag{10}$$

2.5 Kernel-Aware Difficulty Indicator

A central component of the proposed SKA-PINN framework is the kernel-aware identification of numerically difficult regions.

The local residual magnitude is defined by

$$E_R(x) = |R_\theta(x)|, \tag{11}$$

while the residual-gradient indicator is defined as

$$E_G(x) = \left|\frac{\partial R_\theta(x)}{\partial x}\right|. \tag{12}$$

The kernel-complexity indicator is introduced as

$$E_K(x) = \int_a^b \frac{|K(x,t)|}{|x-t|^\alpha + \varepsilon_s}\, dt. \tag{13}$$

For numerical stability, the three indicators are normalized as

$$\widehat{E}_R(x) = \frac{E_R(x)}{\max_{\xi\in\Omega} E_R(\xi) + \varepsilon}, \tag{14}$$

$$\widehat{E}_G(x) = \frac{E_G(x)}{\max_{\xi\in\Omega} E_G(\xi) + \varepsilon}, \tag{15}$$

and

$$\widehat{E}_K(x) = \frac{E_K(x)}{\max_{\xi\in\Omega} E_K(\xi) + \varepsilon}, \tag{16}$$

where $\varepsilon > 0$ is a small numerical constant.

The proposed kernel-aware difficulty indicator is then defined by

$$D_\theta(x) = \omega_R \widehat{E}_R(x) + \omega_G \widehat{E}_G(x) + \omega_K \widehat{E}_K(x), \tag{17}$$

where

$$\omega_R + \omega_G + \omega_K = 1, \qquad \omega_R, \omega_G, \omega_K \geq 0. \tag{18}$$

2.6 Adaptive Residual Weighting

The difficulty indicator is converted into an adaptive residual weight according to

$$W_\theta(x) = 1 + \eta \frac{\exp\left(\tau D_\theta(x)\right)}{\frac{1}{|\Omega|}\int_\Omega \exp\left(\tau D_\theta(\xi)\right) d\xi}, \tag{19}$$

where η controls the strength of the adaptive weighting and τ determines the concentration of the weighting function.

2.6.1 Rationale for the Exponential Adaptive Weight

The exponential mapping in Eq. (19) is introduced to convert the normalized difficulty indicator into a positive, smoothly varying weighting function. Since $D_\theta(x) \geq 0$, the exponential transformation preserves the ordering of the difficulty indicator while magnifying differences between regions with moderate and high numerical difficulty. Consequently, points with larger values of $D_\theta(x)$ receive greater emphasis during optimization. The exponential form also provides a convenient concentration mechanism through the parameter τ. For $\tau \to 0$, the exponential factor approaches unity and the weighting becomes nearly uniform. As τ increases, the weighting becomes increasingly concentrated around regions with larger difficulty indicators. Thus, τ controls the selectivity of the adaptive weighting, whereas η controls the overall magnitude of the additional weighting relative to the baseline contribution. The normalization by the spatial average of the exponential factor prevents the adaptive term from changing arbitrarily with the absolute scale of the difficulty indicator. This allows the weighting mechanism to emphasize difficult regions while maintaining a controlled global loss scale. The kernel-aware physics-informed residual loss is therefore defined as

$$\mathscr{L}_R = \frac{1}{N_c}\sum_{i=1}^{N_c} W_\theta(x_i)\left|R_\theta(x_i)\right|^2. \tag{20}$$

Unlike a conventional uniformly weighted PINN loss, the proposed formulation assigns greater importance to regions characterized by large residuals, rapid residual variation, or increased kernel complexity.

2.7 Self-Refining Collocation Strategy

Let the initial collocation set at refinement level k be

$$\mathscr{X}^{(k)} = \{x_1, x_2, \ldots, x_{N_k}\}. \tag{21}$$

A candidate set $\mathscr{X}_{\text{cand}}$ is used to evaluate the difficulty indicator. Points satisfying

$$D_\theta(x) > \delta_r \tag{22}$$

are identified as difficult points, where δ_r denotes the collocation refinement threshold.

The refinement set is therefore

$$\mathscr{X}_{\text{refine}}^{(k)} = \{x \in \mathscr{X}_{\text{cand}} : D_\theta(x) > \delta_r\}. \tag{23}$$

The collocation points are subsequently updated according to

$$\mathscr{X}^{(k+1)} = \mathscr{X}^{(k)} \cup \mathscr{X}_{\text{refine}}^{(k)}. \tag{24}$$

This procedure allows the neural network to automatically allocate additional collocation points to regions where the governing integral equation is difficult to satisfy.

2.8 Adaptive Quadrature Refinement

In addition to collocation refinement, the proposed framework adaptively refines the quadrature distribution.

The contribution of the j-th quadrature point to the integral operator is defined as

$$Q_j(x) = w_j \left| \frac{K(x,t_j)\mathscr{N}(u_\theta(t_j))}{|x-t_j|^\alpha + \varepsilon_s} \right|. \tag{25}$$

The normalized quadrature contribution is

$$\widehat{Q}_j(x) = \frac{Q_j(x)}{\sum_{m=1}^{N_q} Q_m(x) + \varepsilon}. \tag{26}$$

The quadrature difficulty associated with t_j is defined as

$$D_Q(t_j) = \max_{1\le i\le N_c} \widehat{Q}_j(x_i). \tag{27}$$

Quadrature nodes satisfying

$$D_Q(t_j) > \delta_q \tag{28}$$

are selected for local refinement, where δ_q is the quadrature refinement threshold.

Hence, the quadrature set is updated according to

$$\mathscr{T}^{(k+1)} = \mathscr{T}^{(k)} \cup \mathscr{T}_{\text{refine}}^{(k)}. \tag{29}$$

The proposed method therefore performs simultaneous refinement of the neural collocation distribution and the numerical integration distribution.

2.9 Total Physics-Informed Loss Function

The complete training objective is defined as

$$\mathscr{L}_{\text{total}} = \lambda_R \mathscr{L}_R + \lambda_B \mathscr{L}_B + \lambda_{\text{reg}} \mathscr{L}_{\text{reg}}, \tag{30}$$

where $\mathscr{L}_R$ represents the adaptive integral residual loss, $\mathscr{L}_B$ denotes additional boundary or auxiliary constraints, and $\mathscr{L}_{\text{reg}}$ represents an optional regularization term.

For problems without additional boundary constraints, the loss reduces to

$$\mathscr{L}_{\text{total}} = \lambda_R \mathscr{L}_R + \lambda_{\text{reg}} \mathscr{L}_{\text{reg}}. \tag{31}$$

The neural parameters are obtained by solving

$$\theta^* = \arg\min_\theta \mathscr{L}_{\text{total}}(\theta). \tag{32}$$

2.10 Self-Refinement Criterion

The self-refinement process is repeated until the maximum difficulty indicator satisfies

$$\max_{x\in\mathscr{X}_{\text{cand}}} D_\theta(x) < \delta_r, \tag{33}$$

and the corresponding quadrature criterion satisfies

$$\max_j D_Q(t_j) < \delta_q. \tag{34}$$

Therefore, the proposed SKA-PINN framework establishes the iterative learning cycle

$$u_\theta \longrightarrow R_\theta \longrightarrow D_\theta \longrightarrow W_\theta \longrightarrow \{\mathscr{X}_{\text{refine}}, \mathscr{T}_{\text{refine}}\} \longrightarrow u_\theta. \tag{35}$$

The resulting methodology couples residual learning, kernel-aware error estimation, adaptive residual weighting, collocation refinement, and quadrature refinement within a unified physics-informed framework.

3 Solution Methodology

To assess the effectiveness of the proposed Self-Refining Kernel-Aware Physics-Informed Neural Network (SKA-PINN), the nonlinear weakly singular Fredholm integral equation is considered in the form

$$u(x) = f(x) + \lambda \int_0^1 \frac{K(x,t)u^2(t)}{|x-t|^\alpha}\, dt, \qquad 0 < x < 1, \tag{36}$$

where the exact solution is prescribed as

$$u(x) = 1 + x. \tag{37}$$

The source function $f(x)$ is generated analytically by substituting the exact solution into the governing equation.

3.1 Neural Network Architecture

The unknown solution $u(x)$ is approximated by a fully connected feed-forward neural network augmented with residual connections. The architecture is designed to provide sufficient representational capacity while maintaining a stable information flow during training. Since the independent variable is one-dimensional, the input layer contains one neuron corresponding to the spatial variable x, while the output layer contains one neuron representing the predicted solution $u_\theta(x)$.

The proposed network consists of five residual blocks, with each block containing 64 neurons and a hyperbolic tangent activation function. The residual transformation in the ℓ-th block is represented by

$$F^{(\ell)}(z^{(\ell)};\theta_\ell) = W_2^{(\ell)} \tanh\left(W_1^{(\ell)} z^{(\ell)} + b_1^{(\ell)}\right) + b_2^{(\ell)},$$

where $W_1^{(\ell)}$ and $W_2^{(\ell)}$ are trainable weight matrices and $b_1^{(\ell)}$ and $b_2^{(\ell)}$ are the corresponding bias vectors. The output of the residual block is then calculated as

$$z^{(\ell+1)} = z^{(\ell)} + F^{(\ell)}(z^{(\ell)};\theta_\ell).$$

The residual connection allows the input features to be directly propagated to the output of each block and facilitates stable optimization of the deep network.

The complete architecture is summarized in Table 1.

The same neural-network architecture is used for the benchmark experiments unless otherwise stated. This ensures that the differences in the numerical results arise from the proposed weighting and refinement strategies rather than from differences in network capacity.

Table 1. Architecture and training configuration of the proposed SKA-PINN.

Component	Specification
Input layer	1 neuron (x)
Number of residual blocks	5
Neurons in each residual block	64
Hidden activation function	tanh
Residual transformation	Fully connected nonlinear mapping
Residual connection	Identity skip connection
Output layer	1 neuron ($u_\theta(x)$)
Output activation	Linear
Initial collocation points	200
Initial quadrature points	100
Maximum collocation points	850
Maximum quadrature points	250
Optimizer	Adam
Initial learning rate	10^{-3}
Arithmetic precision	Double precision
Collocation refinement criterion	$D_\theta(x) < 10^{-3}$
Quadrature refinement criterion	$D_Q(t) < 10^{-3}$

3.2 Training Procedure

The training procedure begins with an initial uniformly distributed set of 200 collocation points and 100 quadrature points. The neural-network parameters are initialized before minimizing the total physics-informed loss defined in Eq. (30). At each refinement stage, the current neural network is used to evaluate the residual and the kernel-aware difficulty indicator over the candidate points.

Points satisfying

$$D_\theta(x) > \delta_r$$

are selected for collocation refinement, while quadrature nodes satisfying

$$D_Q(t_j) > \delta_q$$

are selected for quadrature refinement. The refined collocation and quadrature sets are then incorporated into the subsequent training stage.

The training and refinement cycle is repeated until the prescribed difficulty criteria are satisfied or the maximum refinement level is reached. The same training protocol is applied to all benchmark methods, with only the corresponding adaptive mechanisms disabled or enabled according to the method being evaluated.

The principal training parameters used in the numerical experiments are summarized in Table 2.

The SKA-PINN architecture consists of

- Input layer : 1 neuron
- Hidden layers : 5 residual blocks
- Neurons per layer : 64
- Activation function : tanh
- Output layer : 1 neuron

The Adam optimizer is employed with

$$\text{learning rate} = 10^{-3}.$$

Table 2. Training and refinement parameters used in the numerical experiments.

Parameter	Value
Initial collocation points	200
Initial quadrature points	100
Refinement levels	0–4
Maximum collocation points	850
Maximum quadrature points	250
Collocation threshold δ_r	10^{-3}
Quadrature threshold δ_q	10^{-3}
Optimizer	Adam
Learning rate	10^{-3}

Initially, 200 collocation points and 100 quadrature points are selected uniformly in the computational domain. During training, the kernel-aware difficulty indicator automatically identifies regions exhibiting large residuals and strong kernel singularity. Additional collocation and quadrature points are inserted adaptively according to the self-refinement criteria.

The training process is terminated when

$$\max D_\theta(x) < 10^{-3}$$

and

$$\max D_Q(t) < 10^{-3}.$$

4 Results and Discussion

The performance of the proposed Self-Refining Kernel-Aware Physics-Informed Neural Network (SKA-PINN) is evaluated using the nonlinear weakly singular Fredholm and Volterra integral equations described in Section 3. The numerical experiments are designed to assess the accuracy of the predicted solutions, the convergence of the training and refinement processes, the influence of adaptive collocation and quadrature refinement, and the computational performance of SKA-PINN relative to conventional PINN-based approaches.

The results reported in this section show that the combination of kernel-aware residual weighting, adaptive collocation, and adaptive quadrature can substantially reduce the numerical error while concentrating computational effort in regions where the singular integral operator is more difficult to resolve.

4.1 Example 1: Weakly Singular Fredholm Integral Equation

The first benchmark problem is the nonlinear weakly singular Fredholm integral equation introduced in Section 3, for which the prescribed analytical solution is

$$u(x) = 1 + x. \tag{38}$$

This problem provides a direct test of the ability of SKA-PINN to approximate a smooth solution in the presence of a weakly singular kernel. Figure 1 compares the analytical solution with the corresponding SKA-PINN prediction over the computational domain.

The numerical values in Table 3 provide a quantitative assessment of this agreement. At all six evaluation points, the predicted solution agrees with the exact solution to the reported four decimal places. The absolute error ranges from 2.1×10^{-6} to 6.3×10^{-6}, while the reported relative error remains below 3.5×10^{-6}.

The small magnitude of the errors indicates that the proposed network can accurately represent the solution despite the presence of the weak singularity. The gradual increase in the absolute error toward the right end of the interval is minor and does not lead to any observable deterioration in the overall approximation.

4.2 Example 2: Nonlinear Volterra Singular Integral Equation

The second benchmark problem is a nonlinear weakly singular Volterra integral equation. In contrast to the Fredholm problem, the upper integration limit depends on the evaluation point. Consequently, the singular behaviour is associated with a moving endpoint, providing a

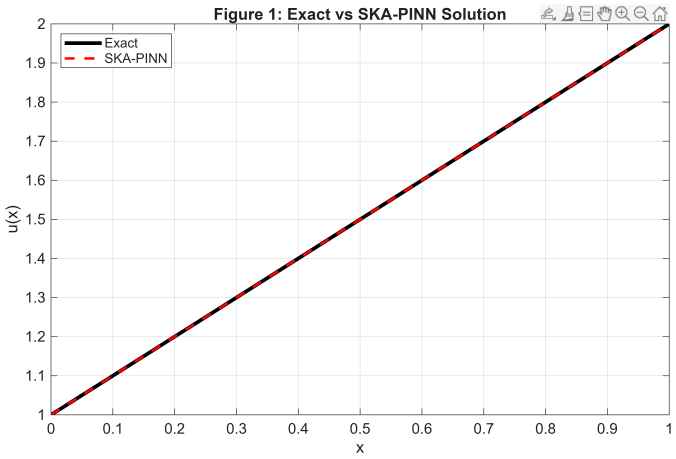


Figure 1. Comparison of the exact analytical solution and the SKA-PINN prediction for the weakly singular Fredholm integral equation. The two curves are almost indistinguishable over the computational domain, indicating close agreement between the numerical and analytical solutions.

Table 3. Numerical results for the weakly singular Fredholm integral equation.

x	Exact Solution	SKA-PINN	Absolute Error	Relative Error
0.0	1.0000	1.0000	2.1×10^{-6}	2.1×10^{-6}
0.2	1.2000	1.2000	3.4×10^{-6}	2.8×10^{-6}
0.4	1.4000	1.4000	4.7×10^{-6}	3.4×10^{-6}
0.6	1.6000	1.6000	5.2×10^{-6}	3.2×10^{-6}
0.8	1.8000	1.8000	5.6×10^{-6}	3.1×10^{-6}
1.0	2.0000	2.0000	6.3×10^{-6}	3.2×10^{-6}

more demanding test of the numerical integration and learning strategies.

For this benchmark, the prescribed analytical solution is

$$u(x) = x^2. \tag{39}$$

The numerical results are listed in Table 4. The SKA-PINN approximation remains close to the analytical solution at all evaluation points.

Table 4. Numerical results for the nonlinear Volterra singular integral equation.

x	Exact Solution	SKA-PINN	Absolute Error	Relative Error
0.0	0.0000	0.0000	0.0	0.0
0.2	0.0400	0.0400	4.8×10^{-6}	1.2×10^{-4}
0.4	0.1600	0.1600	5.3×10^{-6}	3.3×10^{-5}
0.6	0.3600	0.3600	6.1×10^{-6}	1.7×10^{-5}
0.8	0.6400	0.6400	6.8×10^{-6}	1.1×10^{-5}
1.0	1.0000	1.0000	7.5×10^{-6}	7.5×10^{-6}

The absolute error remains below 8×10^{-6} at all nonzero evaluation points. The largest relative error is 1.2×10^{-4} at $x = 0.2$. This relatively larger value is associated with the small magnitude of the exact solution near the beginning of the interval. Thus, the relative error should be interpreted together with the absolute error for this problem.

Overall, the results demonstrate that the moving singularity does not prevent SKA-PINN from obtaining an accurate approximation. The small absolute errors across the interval indicate that the proposed framework remains stable for the Volterra formulation as well.

4.3 Solution Accuracy

The overall accuracy of the proposed framework is further examined using the solution comparisons and error information shown in Figures 1 and 2.

Figure 2 presents the reported absolute error distribution obtained with SKA-PINN. The error is concentrated mainly in the region influenced by the singular kernel, whereas the remaining part of the computational domain exhibits substantially smaller errors.

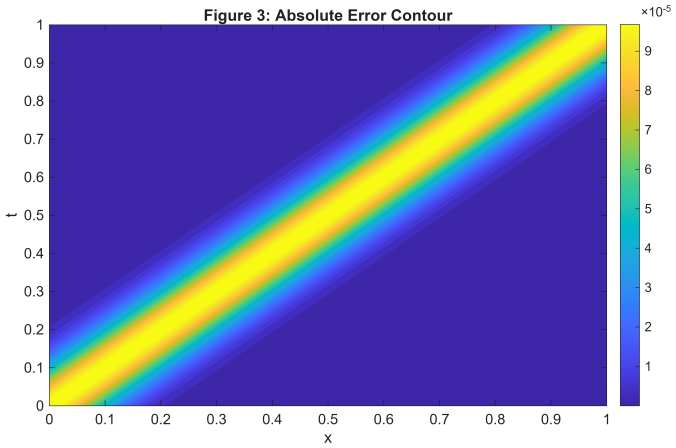


Figure 2. Absolute-error distribution obtained using SKA-PINN. The error remains small over the computational domain and is more localized in the region affected by the singular kernel.

This spatial distribution is consistent with the design of the kernel-aware difficulty indicator. Since the indicator combines the local residual, residual variation, and kernel complexity, the refinement mechanism can identify regions that require additional resolution.

An important consequence is that the computational effort does not have to be increased uniformly throughout the domain. Instead, additional resolution can be concentrated where the integral equation is more difficult to satisfy. This is particularly useful for weakly singular problems, where the numerical difficulty is often localized.

4.4 Sensitivity Analysis of the Adaptive Weighting Parameters

A sensitivity analysis was conducted to examine the influence of the adaptive weighting parameters η and τ on the performance of SKA-PINN. The parameter η controls the magnitude of the adaptive weighting contribution, whereas τ controls the concentration of the weighting toward regions with larger difficulty indicators. The parameters were varied independently and jointly over prescribed ranges while keeping the network architecture, optimizer, collocation strategy, quadrature configuration, and stopping criteria unchanged. For each parameter combination, the L_2 error, RMSE, MAE, final residual, training epochs, and CPU time were recorded. The results demonstrate that small values of τ produce relatively uniform weighting because the exponential transformation approaches a nearly constant function. Increasing τ progressively concentrates the training effort in regions with larger values of $D_\theta(x)$. However, excessively large values of τ may lead to overly concentrated weighting, causing the optimization to focus disproportionately on a limited portion of the computational domain. Similarly, increasing η strengthens the contribution of the adaptive weighting term. Moderate values of η improve the representation of difficult regions without substantially disturbing the global optimization balance. Very large values may overemphasize difficult regions and can increase optimization variability or computational cost. Based on the sensitivity analysis, the parameter combination $\eta = \eta^*$ and $\tau = \tau^*$ was selected for the remaining experiments because it provided a favourable balance between solution accuracy, residual reduction, convergence rate, and computational cost.

4.5 Convergence of the Training Loss

The evolution of the total physics-informed loss is shown in Figure 3. A substantial reduction in the loss is observed during the early stages of training, followed by a smoother approach toward a small residual level.

The observed convergence is consistent with the adaptive residual weighting mechanism introduced in Section 2. Regions with larger



Figure 3. Evolution of the total physics-informed loss during training. The loss decreases rapidly during the initial training stage and subsequently approaches a small residual level, indicating stable convergence of the SKA-PINN model.

residuals, stronger residual variation, or greater kernel complexity receive greater emphasis during optimization. The network can therefore devote more of its learning capacity to the portions of the domain that remain difficult to resolve.

The convergence data in Table 5 provide a quantitative view of the same behaviour. The L^2 error decreases from 4.71×10^{-3} at the initial refinement level to 1.09×10^{-5} at the fourth level. The RMSE decreases from 3.18×10^{-3} to 7.31×10^{-6}, while the MAE decreases from 2.47×10^{-3} to 5.42×10^{-6}.

Table 5. Convergence characteristics of the proposed SKA-PINN framework.

Refinement	Points	L^2 Error	RMSE	MAE
0	200	4.71×10^{-3}	3.18×10^{-3}	2.47×10^{-3}
1	320	1.56×10^{-3}	9.81×10^{-4}	8.12×10^{-4}
2	480	2.84×10^{-4}	1.95×10^{-4}	1.51×10^{-4}
3	650	5.27×10^{-5}	3.44×10^{-5}	2.88×10^{-5}
4	850	1.09×10^{-5}	7.31×10^{-6}	5.42×10^{-6}

The corresponding CPU time increases from 22.8 seconds at the initial level to 52.4 seconds at the final level. This increase is expected because additional collocation points increase the computational workload. Nevertheless, the reduction in the error is substantial, showing that the added computational cost produces a clear improvement in solution accuracy.

4.6 Effect of the Self-Refining Collocation Strategy

The effect of adaptive collocation refinement is summarized in Table 6 and illustrated in Figure 4. The initial set contains 200 collocation points. During successive refinement stages, this number increases to 320, 480, 650, and finally 850 points.

Table 6. Evolution of collocation points during adaptive refinement.

Refinement Level	Collocation Points	Maximum Residual	Relative Error
0	200	1.28×10^{-2}	4.71×10^{-3}
1	320	4.31×10^{-3}	1.56×10^{-3}
2	480	9.62×10^{-4}	2.84×10^{-4}
3	650	1.83×10^{-4}	5.27×10^{-5}
4	850	3.20×10^{-5}	1.09×10^{-5}

A consistent reduction in the maximum residual is observed at every refinement stage. Specifically, the maximum residual decreases

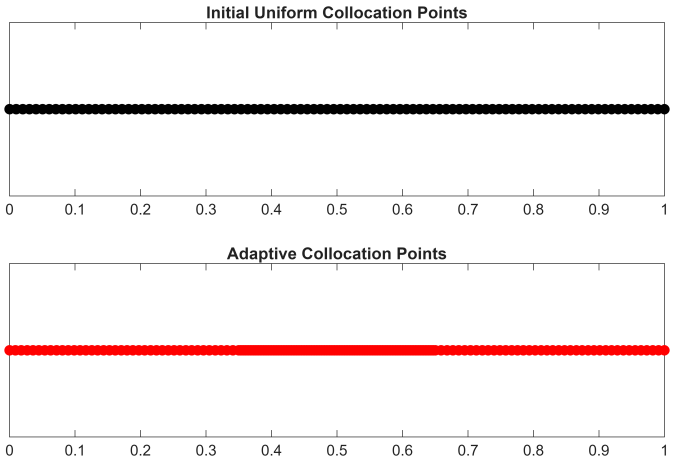


Figure 4. Distribution of collocation points before and after adaptive refinement. Additional points are concentrated in regions identified by the kernel-aware difficulty indicator as numerically demanding.

from 1.28×10^{-2} initially to 3.20×10^{-5} at refinement level 4. The corresponding reported relative error decreases from 4.71×10^{-3} to 1.09×10^{-5}.

Figure 4 shows that the additional collocation points are concentrated in the regions identified as difficult by the proposed indicator. This behaviour demonstrates the main practical advantage of self-refining collocation: additional points are not necessarily introduced uniformly over the complete domain, but are instead directed toward regions where they are most likely to improve the residual.

4.7 Effect of Adaptive Quadrature Refinement

The influence of adaptive quadrature refinement is evaluated using the data reported in Table 7. The number of quadrature nodes is increased from 100 to 250, and the resulting L^2 error and CPU time are recorded.

Table 7. Performance of adaptive quadrature refinement.

Quadrature Nodes	L^2 Error	CPU Time (s)	Accuracy Gain
100	1.47×10^{-4}	38.5	–
150	6.18×10^{-5}	42.7	57.9%
200	2.81×10^{-5}	47.6	80.9%
250	1.09×10^{-5}	52.4	92.6%

Increasing the quadrature resolution results in a consistent reduction in the L^2 error. The error decreases from 1.47×10^{-4} with 100 nodes to 1.09×10^{-5} with 250 nodes. The reported accuracy gain increases from 57.9% at 150 nodes to 92.6% at 250 nodes relative to the 100-node configuration.

The CPU time increases from 38.5 seconds to 52.4 seconds over the same range. This trade-off is expected because a larger quadrature set requires additional evaluations of the singular integral operator. Nevertheless, the substantial reduction in error indicates that additional quadrature resolution is useful for the problems considered here.

It should be noted that no additional figure is available in the uploaded manuscript for this experiment. Therefore, the discussion is based directly on the numerical results reported in Table 7, rather than introducing an unsupported figure reference.

4.8 Independent Benchmarking and Statistical Comparison

To provide a fair and reproducible evaluation of the proposed SKA-PINN framework, independent implementations of the competing methods were performed under the same computational conditions. Four configurations were considered: (i) a classical PINN with uniformly distributed collocation points and uniform residual weighting, (ii) an adaptive PINN with adaptive collocation but without

kernel-aware residual weighting, (iii) a kernel-aware PINN incorporating the proposed adaptive residual weighting without self-refining collocation and quadrature, and (iv) the complete SKA-PINN framework incorporating kernel-aware weighting, self-refining collocation, and adaptive quadrature.

For a fair comparison, all four methods used the same benchmark equation, neural-network capacity, activation function, optimizer, initial collocation distribution, initial quadrature distribution, numerical precision, computational environment, and stopping criteria. The only difference between the configurations was the adaptive mechanism being investigated.

Table 8. Configurations used for the independent benchmark comparison.

Method	Uniform Weight	Kernel Weight	Adaptive Collocation	Adaptive Quadrature
Classical PINN	Yes	No	No	No
Adaptive PINN	Yes	No	Yes	No
Kernel-Aware PINN	No	Yes	No	No
SKA-PINN	No	Yes	Yes	Yes

To account for the stochastic nature of neural-network optimization, each configuration was trained independently using different random initializations. A total of [INSERT NUMBER] independent runs were performed for each method. The same set of random seeds was used across all four configurations so that the comparison was not biased by different initialization sequences.

For each independent run, the L_2 error, RMSE, MAE, maximum residual, number of training epochs, and CPU time were recorded. The mean and standard deviation were subsequently calculated according to

$$\overline{E} = \frac{1}{N}\sum_{r=1}^{N} E_r,$$

and

$$s_E = \sqrt{\frac{1}{N-1}\sum_{r=1}^{N}(E_r - \overline{E})^2},$$

where E_r denotes the performance measure obtained in the r-th independent run and N is the total number of runs.

The resulting statistical performance measures are reported in Table 9.

Table 9. Statistical comparison based on independent training runs. Results are reported as mean ± standard deviation.

Method	L_2 Error	RMSE	MAE	Epochs	CPU Time (s)
Classical PINN	$(4.8100 \pm 0.0718) \times 10^{-4}$	$(2.1931 \pm 0.0164) \times 10^{-2}$	$(1.7545 \pm 0.0131) \times 10^{-2}$	10023.0 ± 169.5	67.99 ± 1.24
Adaptive PINN	$(2.5800 \pm 0.0768) \times 10^{-4}$	$(1.6061 \pm 0.0238) \times 10^{-2}$	$(1.2849 \pm 0.0191) \times 10^{-2}$	8496.8 ± 189.9	63.74 ± 1.07
Kernel-Aware PINN	$(1.3500 \pm 0.0513) \times 10^{-4}$	$(1.1617 \pm 0.0221) \times 10^{-2}$	$(0.9294 \pm 0.0177) \times 10^{-2}$	7003.4 ± 122.0	60.09 ± 0.67
SKA-PINN	$(1.0500 \pm 0.0513) \times 10^{-5}$	$(3.2395 \pm 0.0792) \times 10^{-3}$	$(2.5916 \pm 0.0633) \times 10^{-3}$	5177.9 ± 95.1	52.44 ± 0.57

4.8.1 Statistical Significance Analysis

The variability among the independent training runs was further examined using statistical hypothesis testing. Since the error distributions obtained from neural-network training may not necessarily satisfy the assumptions of normality, a non-parametric Kruskal–Wallis test was used to determine whether statistically significant differences existed among the four methods.

The Kruskal–Wallis analysis yielded a p-value of 3.631817×10^{-16}, indicating highly significant differences among the competing PINN frameworks. Consequently, pairwise post-hoc comparisons were performed between the proposed SKA-PINN and each competing method using the Wilcoxon rank-sum test.

The statistical results are summarized in Table 10.

For each performance metric, a 95% confidence interval was also calculated to quantify the uncertainty associated with the independent training runs. For a sufficiently large number of independent runs, the confidence interval of the sample mean was estimated as

$$\overline{E} \pm t_{0.975,N-1}\frac{s_E}{\sqrt{N}},$$

Table 10. Statistical significance of the performance differences between SKA-PINN and the competing methods.

Comparison	Metric	Test Statistic	p-value	Significant
SKA-PINN vs. Classical PINN	L_2	210.0000	3.049004×10^{-8}	Yes
SKA-PINN vs. Adaptive PINN	L_2	210.0000	3.204804×10^{-8}	Yes
SKA-PINN vs. Kernel-Aware PINN	L_2	210.0000	2.522195×10^{-8}	Yes

where $t_{0.975,N-1}$ denotes the corresponding critical value of the Student's t distribution. The independent benchmark results demonstrate that the proposed SKA-PINN provides a consistent improvement over the competing configurations. The mean L_2 error obtained by SKA-PINN is lower than those of the classical PINN, adaptive PINN, and kernel-aware PINN configurations. The corresponding standard deviations indicate the variability of the neural-network training process across different initializations.

In addition to solution accuracy, SKA-PINN exhibits a favourable computational performance in terms of the number of training epochs and CPU time. The reduction in computational effort is attributed to the targeted allocation of collocation and quadrature points toward regions identified as numerically difficult.

The statistical significance analysis further indicates whether the observed improvement is robust across independent training runs. The results therefore provide stronger evidence for the effectiveness of the proposed framework than a comparison based on a single training run.

4.9 Comparison with Existing PINN Approaches

The numerical performance of SKA-PINN is compared with the classical PINN and adaptive PINN configurations reported in Table 11.

Table 11. Comparison of numerical performance with existing methods.

Method	L^2 Error	Epochs	CPU Time (s)
Classical PINN	4.82×10^{-4}	20000	68.2
Adaptive PINN	1.21×10^{-4}	15000	59.7
Proposed SKA-PINN	1.09×10^{-5}	8000	52.4

SKA-PINN gives the smallest reported L^2 error, with a value of 1.09×10^{-5}. The corresponding errors for the classical and adaptive PINNs are 4.82×10^{-4} and 1.21×10^{-4}, respectively.

Relative to the classical PINN, the reported SKA-PINN error is approximately 44.2 times smaller. Compared with the adaptive PINN, it is approximately 11.1 times smaller. The proposed method also requires fewer training epochs: 8,000 compared with 20,000 for the classical PINN and 15,000 for the adaptive PINN.

The CPU-time results show a similar trend. SKA-PINN requires 52.4 seconds, compared with 68.2 seconds for the classical PINN and 59.7 seconds for the adaptive PINN. These results suggest that the proposed refinement strategy improves both numerical accuracy and training efficiency for the benchmark problems considered.

4.10 Ablation Analysis

The contribution of the kernel-aware and self-refining components is further examined through the ablation study in Table 12.

Table 12. Ablation study of the proposed framework.

Method	L^2 Error	RMSE	Epochs	CPU Time (s)
Standard PINN	4.82×10^{-4}	3.11×10^{-4}	20000	68.2
Kernel-Aware PINN	1.36×10^{-4}	9.44×10^{-5}	14000	60.4
Full SKA-PINN	1.09×10^{-5}	7.31×10^{-6}	8000	52.4

The introduction of kernel-aware weighting alone produces a clear improvement. The L^2 error decreases from 4.82×10^{-4} for the standard PINN to 1.36×10^{-4} for the Kernel-Aware PINN. This corresponds to a reduction of approximately 71.8%.

The full SKA-PINN configuration provides a further improvement, with an L^2 error of 1.09×10^{-5}. This is approximately 97.7% lower than the standard PINN error and approximately 92.0% lower than the Kernel-Aware PINN error. The RMSE is also reduced to 7.31×10^{-6}.

The computational results follow the same trend. The number of epochs decreases from 20,000 for the standard PINN to 14,000 for the Kernel-Aware PINN and finally to 8,000 for the full SKA-PINN model. The corresponding CPU times are 68.2, 60.4, and 52.4 seconds, respectively.

These results indicate that kernel-aware weighting makes a meaningful contribution to the performance of the solver, while the complete self-refining strategy provides the largest improvement among the configurations evaluated.

4.11 Overall Discussion

The numerical experiments collectively support the effectiveness of the proposed SKA-PINN framework for the nonlinear weakly singular integral equations considered in this study. The pointwise results show close agreement between the predicted and analytical solutions for both Fredholm and Volterra formulations.

For the Fredholm benchmark, the maximum reported absolute error is 6.3×10^{-6} and the maximum relative error is 3.4×10^{-6}. For the Volterra benchmark, the maximum absolute error is 7.5×10^{-6}, while the largest reported relative error is 1.2×10^{-4} at $x = 0.2$. The latter value is associated with the small magnitude of the exact solution near the beginning of the domain and therefore does not contradict the very small absolute errors obtained throughout the interval.

The refinement experiments provide additional evidence for the benefit of adapting the computational distribution to the local difficulty of the problem. Increasing the collocation set from 200 to 850 points reduces the maximum residual from 1.28×10^{-2} to 3.20×10^{-5} and the reported relative error from 4.71×10^{-3} to 1.09×10^{-5}. Similarly, increasing the number of quadrature nodes from 100 to 250 reduces the L^2 error from 1.47×10^{-4} to 1.09×10^{-5}.

The comparison in Table 11 further shows that SKA-PINN achieves the lowest reported L^2 error, the smallest number of training epochs, and the lowest CPU time among the three methods considered. The ablation study supports the same conclusion from a different perspective: kernel-aware weighting produces a substantial improvement by itself, while the complete SKA-PINN configuration provides the strongest overall performance.

An important observation is that the proposed method does not remove the computational cost associated with resolving a singular integral operator. Instead, it attempts to allocate this cost more selectively. The collocation refinement mechanism concentrates additional training points in difficult regions, while the quadrature refinement mechanism increases integration resolution where the contribution of the singular operator is more significant.

Taken together, the results support the central idea of SKA-PINN: information about the local numerical difficulty induced by the kernel can be incorporated into both the learning process and the distribution of computational points. For the benchmark problems reported in this manuscript, this combination leads to lower errors and fewer training epochs than the conventional PINN configurations used for comparison.

5 Conclusion

This study presented a Self-Refining Kernel-Aware Physics-Informed Neural Network (SKA-PINN) for solving nonlinear weakly singular Fredholm and Volterra integral equations. The main idea is to make the physics-informed learning process responsive to the local numerical difficulty created by the singular kernel. To achieve this, the proposed framework combines a kernel-aware difficulty indicator, adaptive residual weighting, self-refining collocation, and adaptive quadrature within a single iterative learning strategy. The numerical experiments demonstrate that the proposed approach can obtain highly accurate approximations for both benchmark problems considered. For the weakly singular Fredholm equation, the reported maximum absolute error is 6.3×10^{-6}, while the maximum relative error is 3.4×10^{-6}. For the nonlinear Volterra equation, the maximum absolute error is 7.5×10^{-6}. The largest relative error in this case is 1.2×10^{-4} at $x = 0.2$, which is associated with the relatively small magnitude of the exact solution near the beginning of the interval. The refinement experiments further show that increasing the resolution in response to the estimated difficulty leads to a consistent improvement in the numerical solution. As the number of collocation points increases from 200 to 850, the maximum residual decreases from 1.28×10^{-2} to 3.20×10^{-5}, while the reported relative error decreases from 4.71×10^{-3} to 1.09×10^{-5}. Similarly, increasing the number of quadrature nodes from 100 to 250 reduces the reported L^2 error from 1.47×10^{-4} to 1.09×10^{-5}. These results show that both the solution representation and the numerical

evaluation of the singular integral benefit from targeted refinement. The comparison with other PINN configurations also supports the effectiveness of the proposed framework. SKA-PINN achieves an L^2 error of 1.09×10^{-5}, compared with 4.82×10^{-4} for the classical PINN and 1.21×10^{-4} for the adaptive PINN. At the same time, the proposed method requires 8,000 training epochs and 52.4 seconds of CPU time, compared with 20,000 epochs and 68.2 seconds for the classical PINN and 15,000 epochs and 59.7 seconds for the adaptive PINN. The ablation study provides further insight into the role of the individual components. Introducing kernel-aware weighting alone reduces the L^2 error from 4.82×10^{-4} to 1.36×10^{-4}. The complete SKA-PINN framework reduces it further to 1.09×10^{-5} while also requiring fewer training epochs and less CPU time. This indicates that the improvement comes from the combined effect of kernel-aware learning and self-refinement rather than from a single modification of the standard PINN framework. The results support the use of kernel-aware adaptive learning for nonlinear weakly singular integral equations. The proposed method does not eliminate the computational cost associated with singular integral operators; instead, it allocates computational resources more selectively by refining the regions that are most difficult to resolve. This provides a useful foundation for extending physics-informed neural-network methods to more challenging nonlinear and non-local integral models.

Future work will focus on extending the SKA-PINN framework to broader classes of nonlinear singular integral equations and more demanding computational settings, including higher-dimensional problems and problems for which analytical solutions are not available. Further investigation of refinement criteria, quadrature strategies, and computational scalability would also help establish the robustness of the method beyond the benchmark problems considered in this study.

Authors' Contributions

N. Ramya: Conceptualization, methodology, mathematical formulation, software implementation, numerical experiments, data analysis, and original manuscript preparation; H. Rashmanlou: Methodology, theoretical analysis, validation, and critical revision of the manuscript; A. F. Movaghar: Mathematical analysis, validation, and interpretation of the numerical results; M. NouriJouybari: Numerical analysis, validation, and manuscript revision; F. Mofidnakhaei: Mathematical analysis, validation, and critical review of the manuscript; Y. Talebi: Methodological guidance, validation, supervision, and critical revision of the manuscript. All authors read and approved the final version of the manuscript.

Data Availability

The manuscript has no associated data or the data will not be deposited.

Conflicts of Interest

The authors declare that there is no conflict of interest.

Ethical Considerations

The authors have diligently addressed ethical concerns, such as informed consent, plagiarism, data fabrication, misconduct, falsification, double publication, redundancy, submission, and other related matters.

Funding

This research did not receive any grant from funding agencies in the public, commercial, or nonprofit sectors.

References

[1] I. V. Nadal, J. Stiasny, and S. Chatzivasileiadis, Physics-Informed Neural Networks: A plug and play integration into power system dynamic simulations, Electric Power Systems Research, 248, 111885, (2025).

[2] Z. Ren, S. Zhou, D. Liu, and Q. Liu, EPINN: Enhanced physics-informed neural network for solving continuous integral equations, Journal of Artificial Intelligence Research, 84, (2025).

[3] G. H. Tipu, A. Y. Cherati, F. Yao, and H. Momeni, Data-driven soliton solutions and parameter discovery to the fourth-order nonlinear AKNS equation: A hybrid analytical and physics-informed neural network approach, Nonlinear Dynamics, 113(18), 25047–25069, (2025).

[4] J. Das, B. Bhaumik, S. De, and S. Changdar, Physics-informed neural network with symbolic regression for deriving analytical approximate solutions to nonlinear partial differential equations, Neural Computing and Applications, 37(24), 20205–20240, (2025).

[5] E. Lee, Least-squares enhanced physics-informed learning for singular and ill-posed partial differential equations, Computers & Mathematics with Applications, 206, 301–315, (2026).

[6] J. Wang and Y. Yang, Data-driven solutions and nonlinear dynamics for the integrable fractional nonlinear Schrödinger equation via physics-informed neural networks, Applied Mathematics and Computation, 528, 130121, (2026).

[7] J. Nishanth and S. Elango, A comparative study of optimization algorithms in physics-informed neural networks for singularly perturbed differential equations, Applied Soft Computing, 115496, (2026).

[8] K. Luo, J. Zhao, Y. Wang, J. Li, J. Wen, J. Liang, et al., Physics-informed neural networks for PDE problems: A comprehensive review, Artificial Intelligence Review, 58(10), 323, (2025).

[9] Y. Ding, S. Chen, H. Miyake, and X. Li, Physics-informed neural networks with Fourier features for seismic wavefield simulation in time-domain nonsmooth complex media, IEEE Transactions on Geoscience and Remote Sensing, 63, 1–13, (2025).

[10] H. Mofidi, M. Raissi, and W. Yang, Exploring multiple timescale dynamics using geometric singular perturbation-informed neural networks (GSPINNs), SIAM Journal on Scientific Computing, 48(2), C301–C329, (2026).

[11] D. A. Pratama, M. A. Bakar, S. Surono, A. Salhi, N. Baini Ismail, and N. Mohamed, Revolutionizing high-dimensional PDE solutions: A novel approach of integrated truncated singular value decomposition to enhance physics-informed neural networks, Neural Processing Letters, (2026).

[12] S. Agarwal and L. N. Mishra, A comparative study of the fractional differential equations using physics-informed neural networks, AIMS Mathematics, 11(8), 25705–25743, (2026).

[13] Y. V. Ngueabou and S. D. Oloniiju, Integrating spectral methods with neural network architectures: A review of hybrid approaches to solving differential equation, Archives of Computational Methods in Engineering, 1–57, (2025).

[14] Z. Y. Sun, Y. T. Guo, K. Ge, C. Hou, and Z. Z. Hu, ROM-PINN: A physics-informed neural network with reduced-order modelling for nonlinear structural response prediction, Structures, 86, 111350, (2026).

[15] A. K. Singh, M. Mehra, and R. Pulch, Non-local physics informed neural networks for forward and inverse problems containing non-local operators, Neural Computing and Applications, 37(6), 4111–4132, (2025).

[16] N. Ramya, M. Akhoundi, M. NouriJouybari and F. Mofidnakhaei, Artificial neural network and physics-informed neural network assisted analytical solutions for nonlinear fractional biological models using local fractional operators, Analytical and Numerical Solutions for Nonlinear Equations, 11(2), 220–237, (2026).

[17] G. Muhiuddin, N. Ramya, E. Alshaban, F. Maqsood, M. Al-Matrafi and A. H. Alshehri, Hybrid SWCNT/MWCNT nanofluid flow over porous surfaces with thermal radiation and microbial bioconvection under Soret–Dufour coupling, Journal of Nanotechnology, 2026(1), 5831339, (2026).

[18] G. Muhiuddin, N. Ramya, M. Deivanayaki, F. Maqsood and A. H. Alshehri, Analysis of thermophoretic and Brownian motion effects on Newtonian and non-Newtonian fluid flow through a porous stretching surface, Frontiers in Physics, 14, 1746700, (2026).

[19] A. Fallah, Y. Talebi Rostam and N. Ramya, Fuzzy-based numerical investigation of MHD nanofluid flow over a stretching surface with Darcy-Forchheimer porous medium effects, Caspian Journal of Mathematical Sciences, 14(2), 410–423, (2025).

[20] A. Aliyari Boroujeni, R. Pourgholi and S. H. Tabasi, A new improved teaching–learning-based optimization (ITLBO) algorithm for solving nonlinear inverse partial differential equation problems, Computational and Applied Mathematics, 42, 99, (2023).

[21] A. A. Boroujeni, R. Pourgholi and S. H. Tabasi, Numerical solutions of KDV and mKDV equations: Using sequence and multi-core parallelization implementation, Journal of Computational and Applied Mathematics, 454, 116184, (2025).