# EPINN: Enhanced Physics-Informed Neural Network for Solving Continuous Integral Equations

ZHIYUAN REN*, University of Electronic Science and Technology of China, China
SHIJIE ZHOU, University of Electronic Science and Technology of China, China
DONG LIU, University of Electronic Science and Technology of China, China
QIHE LIU, University of Electronic Science and Technology of China, China

**Background:** Integral equations play a crucial role in modeling complex systems across various scientific disciplines. However, traditional numerical methods and existing physics-informed neural networks (PINNs) face substantial challenges, including the curse of dimensionality, uncontrolled error propagation, and limited generalization capabilities.
**Objectives:** This paper aims to overcome these limitations by developing a robust and scalable solver for high-dimensional and nonlinear integral equations. The primary goal is to achieve higher accuracy and efficiency compared to traditional methods and existing deep learning approaches.
**Methods:** We present the enhanced physics-informed neural network (EPINN), a novel framework that incorporates three key innovations: 1) a variable-order operator decomposition theory that transforms integral equations into well-posed differential systems, thereby mitigating error accumulation, 2) a differentiable primal function projection layer that ensures physical consistency within the Sobolev spaces, and 3) a boundary-aware multi-objective training paradigm that improves generalization.
**Results:** Experimental validation across five benchmark cases spanning two to four dimensions, including linear/nonlinear Volterra/Fredholm and hybrid Volterra-Fredholm integral equations, demonstrates the superior performance of EPINN. Compared with traditional methods, EPINN reduces relative errors by 1 to 2 orders of magnitude, while achieving over 92% accuracy with limited training data. When compared with existing deep learning solvers, EPINN provides significant improvements in computational efficiency (with a speedup factor of 3 to 6 times) and accuracy (error reduction of 23% to 85%).
**Conclusions:** These advancements establish EPINN as a robust and scalable solver for high-dimensional and nonlinear integral equations, with wide-ranging applications in computational physics and engineering. The success of EPINN suggests that integrating physical principles with neural networks can lead to substantial improvements in solving complex mathematical problems.

**JAIR Associate Editor:** Atlas Wang

**JAIR Reference Format:**
Zhiyuan Ren, Shijie Zhou, Dong Liu, and Qihe Liu. 2025. EPINN: Enhanced Physics-Informed Neural Network for Solving Continuous Integral Equations. *Journal of Artificial Intelligence Research* 84, Article 27 (December 2025), 46 pages. DOI: 10.1613/jair.1.20161

*Corresponding Author.

Authors' Contact Information: Zhiyuan Ren, ORCID: 0009-0005-5682-787X, renzhiyuan@uestc.edu.cn, University of Electronic Science and Technology of China, Chengdu, Sichuan Province, China; Shijie Zhou, ORCID: 0000-0001-8314-754X, sjzhou@uestc.edu.cn, University of Electronic Science and Technology of China, Chengdu, Sichuan Province, China; Dong Liu, ORCID: 0009-0002-4768-9039, liudong@uestc.edu.cn, University of Electronic Science and Technology of China, Chengdu, Sichuan Province, China; Qihe Liu, ORCID: 0000-0002-8195-1304, qiheliu@uestc.edu.cn, University of Electronic Science and Technology of China, Chengdu, Sichuan Province, China.


DOI: 10.1613/jair.1.20161

## 1 Introduction

Integral equations are a cornerstone mathematical tool for modeling complex systems across disciplines. In quantum dynamics, Feynman path integrals (Feldbrugge and Jones 2025) encode particle trajectories as high-dimensional integrals ($d \geq 10^3$), but traditional discretization fails beyond four dimensions (Iyengar and Kais 2023). Similarly, nonlocal continuum mechanics uses integral operators to model strain gradients in fractured materials (Eringen and Wegner 2003), where singular kernels induce computational instability. These applications exemplify the need for scalable, singularity-robust solvers–addressed by EPINN's operator decomposition and adaptive training. While comprehensive partial differential equations (PDEs) benchmarks like PDEBench (Takamoto et al. 2022) provide valuable evaluation frameworks for differential equations (DEs), they fall outside our scope as we focus exclusively on integral equations–a distinct mathematical class requiring specialized solution approaches. Integral equations involve global domain dependencies through integration operators, contrasting with PDEs' local differential operators. This fundamental difference necessitates tailored architectures like EPINN's operator decomposition rather than direct application of PDE-oriented methods. Traditional numerical methods, such as Nyström discretization (Atkinson 1997; Sun et al. 2015) and Galerkin projection (H. Gao et al. 2022; Kress 1999), have proven robust for low-dimensional linear problems. However, they face three significant challenges. 1) The curse of dimensionality arises: High-dimensional integrals exhibit exponential computational complexity $O(n^d)$, where $d$ is dimension and $n$ is node count (Bellman 1958). For example, a three-dimensional Fredholm equation ($d = 3$) discretized with $n = 100$ nodes per dimension requires $100^3 = 10^6$ function evaluations. However, a 10 dimensions counterpart ($d = 10$) with the same nodal resolution demands $100^{10} = 10^{20}$ evaluations–exceeding the computational feasibility of modern supercomputers (e.g., Frontier requires $\approx$ one billion years for $10^{18}$ operations at $10^{18}$ FLOP/s). 2) Uncontrollable error propagation occurs: Nonlinear kernels induce error accumulation diverging at $O(k^m)$, with $k$ as step size and $m$ as equation order (Atkinson and Potra 1987). 3) Limited generalization stems from complex boundary conditions requiring tailored discretization, reducing versatility (Brenner and Scott 2008). These three challenges highlight the fundamental limitation of traditional methods–the discretization process disrupts the continuous dependency inherent in integral equations. The integration of deep learning offers a promising alternative–by exploiting the implicit approximation capabilities of neural networks on Sobolev spaces (Yarotsky 2017), it becomes possible to enable end-to-end modeling of continuous solution spaces, thus providing a paradigm shift in solving these problems.

PINNs have revolutionized scientific machine learning by successfully embedding physical constraints into neural network training for PDEs (Raissi et al. 2019). Their ability to solve inverse problems and handle noisy data makes them valuable for PDE-based modeling. However, when applied to integral equations, existing PINN frameworks face three key challenges that limit their effectiveness. 1) There is an operator mismatch, where the continuous dependence of integral kernel functions induces systematic errors in traditional discretization-based approximation methods (Kenney et al. 1989). 2) Uncontrolled error propagation arises: discretization-induced approximations cause integral term errors to grow exponentially as $O(e^{b \cdot k})$, where $b$ is the Lyapunov exponent quantifying system sensitivity, and $k$ is the iteration index. For typical quadrature-based PINNs, $b \propto |\nabla K_i|_{L^\infty}$ (Constantin et al. 1989). 3) A computational efficiency bottleneck emerges, with the increasing demand for computational resources during the neural network training process sharply elevating training cost (Niederreiter 1992).

To address the aforementioned challenges, this paper proposes EPINN, which offers innovative contributions across three key areas. 1) In terms of theoretical advancements, we build upon the existence theorem for integral equations presented in (Atkinson 1997) and introduce a theory based on operator decomposition with variable order. This theory transforms continuous integral equations into equivalent systems of DEs, providing rigorous proof of the well-posedness of the transformed system and mitigating the error accumulation typical of traditional finite approximation methods. 2) In terms of architectural innovation, EPINN incorporates a differentiable primal

function projection layer that dynamically balances physical constraints and data-driven features through adaptive weighting, ensuring the physical consistency of the solution within the Sobolev space $H^2(\Omega)$. 3) In the realm of training paradigms, the approach integrates a boundary-specific sample injection strategy with a multi-objective optimization mechanism, enhancing the generalization performance of the network.

Experimental validation demonstrates that EPINN achieves state-of-the-art performance across diverse integral equation types, including linear and nonlinear, Volterra and Fredholm hybrid, as well as low- and high-dimensional cases. Compared with traditional numerical methods (e.g., Nyström, Galerkin), EPINN reduces relative errors by 1 to 2 orders of magnitude (e.g., achieving $1.37 \times 10^{-8}$ mean squared error (MSE) for a four-dimensional Fredholm equation vs. $7.25 \times 10^{-3}$ for Nyström). Remarkably, EPINN maintains over $> 92\%$ accuracy stability with limited training data, demonstrating superior sample efficiency. Against existing deep learning solvers denoted as ODLM (other deep learning methods), EPINN exhibits 3 to 6 times computational speedup and 23%–93% lower errors (e.g., nonlinear Volterra equations: $2.05 \times 10^{-4}$ L2RE vs. ODLM's $2.76 \times 10^{-4}$), facilitated by its parallelizable architecture and adaptive gradient balancing.

The remainder of this paper is organized as follows. Section 2 provides an overview of the relevant advancements in the field. Section 3 introduces the variable-order operator decomposition theory and problem formulation. Section 4 details the EPINN architecture, loss function design, EPINN's well-posedness, equivalence, convergence analysis, and training algorithm. Section 5 presents multidimensional experiments and ablation studies. Section 6 concludes with directions for future research.

## 2 Related Works

This section reviews and analyzes related work from three aspects: numerical methods for integral equations, advances and challenges of PINN, deep learning solvers for integral equations.

### 2.1 Numerical Methods for Integral Equations

Integral equations are fundamental tools for modeling physical systems and have been the subject of extensive development for over a century, resulting in a well-established framework of numerical methods. For linear equations of the Fredholm and Volterra types, the Nyström method offers efficient solutions by discretizing the integral operator (Atkinson 1997), while the Galerkin method ensures convergence through the use of orthogonal basis function expansions (Kress 1999). For nonlinear equations, iterative methods (e.g., Newton-Kantorovich) (Isewid 2024) and collocation methods significantly improve stability (Brunner 2004; Jiang and X. Gao 2024). When dealing with singular kernels, adaptive mesh refinement and analytical kernel decomposition methods effectively mitigate the impact of singularities (Monegato 1994). In the field of high-frequency electromagnetic scattering, these advancements have facilitated the development of fast multipole algorithms (Greengard and Rokhlin 1987; Kalfa et al. 2023) and preconditioned iterative methods (Christiansen and Nédélec 2002; Liu et al. 2024; Tang and Huang 2024), which reduce the computational complexity to $O(N \log N)$.

Classical methods face inherent limitations when addressing high-dimensional problems. The curse of dimensionality leads to an exponential increase in computational costs as the number of dimensions grows. For instance, the storage requirements for tensor product grids in three-dimensional integral equations can exceed $10^{12}$ units (Trefethen 2019). In complex geometric domains, the computation of singular integrals necessitates fine grid discretization, significantly restricting the applicability of these methods. The underlying issue is that the discretization process disrupts the continuous dependencies inherent in integral equations, highlighting the need for novel, grid-free, and robust solution approaches capable of handling high-dimensional problems effectively.

## 2.2 Advances and Challenges of PINNs

PINNs have introduced a novel paradigm in data-physics-integrated modeling by incorporating the residuals of PDEs into the loss function (Raissi et al. 2019). Key advantages of PINNs include automatic differentiation (AD) to precisely compute differential operators and the implicit approximation of solution continuity via Sobolev spaces. Subsequent research has focused on enhancing training stability and improving the generalization capabilities for complex problems. For instance, VPINN reduces the need for higher-order derivatives by employing weak formulations (Anandh et al. 2025; Kharazmi et al. 2021), adaptive weight distribution mitigates gradient imbalance (Wang et al. 2021), domain decomposition methods effectively address multiscale problems (Jagtap, Kawaguchi, et al. 2020; S. Li et al. 2023), symbolic regularization constrains the solution space (Lu, Pestourie, et al. 2021; D. Yu et al. 2023), and reinforcement learning techniques are leveraged to optimize hyperparameters (Bischof and Kraus 2022; Shakya et al. 2023).

The standard PINN framework, however, exhibits inherent limitations when applied to solving integral equations. Discretization and sampling result in the loss of the continuous dependencies within the integral kernel, leading to mismatches in the representation of operators. The approximation introduced by finite sampling points causes errors in the integral term to grow exponentially during training iterations, leading to uncontrolled error propagation. Furthermore, in high-dimensional scenarios involving singular kernels, the computational demands of neural networks increase significantly, creating a substantial bottleneck in computational efficiency. While existing research has predominantly focused on DEs, the application of PINNs to integral equations remains largely in the exploratory stage.

## 2.3 Deep Learning Solvers for Integral Equations

Deep learning offers two innovative approaches to solving integral equations. Data-driven methods leverage deep neural networks to establish nonlinear mappings from the input function to the solution space. Key contributions include DeepONet's branch-trunk architecture (Lu, Jin, et al. 2021), which suffers from kernel continuity loss under quadrature discretization (Taassob et al. 2024), and Transformer-Operator's self-attention mechanisms (Cao 2021), which exhibit $O(N^2)$ memory overhead in high dimensions (X. Li et al. 2025; Ovadia et al. 2024). Fourier neural operators (FNO) (Z. Li et al. 2021) provide efficient spectral processing but struggle with non-periodic boundaries common in integral equations. Unlike these data-driven approaches, EPINN's operator-theoretic framework preserves continuous dependencies via structured differential decomposition (Section 3.2), avoiding discretization-induced errors while maintaining $O(N)$ complexity. Further, EPINN's boundary-aware training (Section 4.3) mitigates spectral bias in high-frequency kernels–a limitation acknowledged in Transformer-Operator's adaptive Fourier layers (X. Li et al. 2025). Our experimental comparisons (Section 5) demonstrate EPINN's advantages over these SOTA operator learning methods in accuracy, efficiency, and robustness across diverse integral equation types. While these methods can alleviate the curse of dimensionality, they are heavily reliant on the specific characteristics of the training data, which compromises their physical interpretability. Moreover, their generalization capabilities are constrained by the distribution of the training data.

Physics-informed methods guide neural networks toward physically consistent solutions by integrating constraints derived from integral equations. For example, the work in (Lin et al. 2025) introduces Monte Carlo sampling to approximate the integral terms, combined with variance reduction techniques to accelerate convergence. The CPINN framework utilizes a domain decomposition strategy to enhance adaptability to complex geometries (Jagtap, Kharazmi, et al. 2020), while gPINN improves solution smoothness through gradient-based constraints (Eshkofti and Hosseini 2023; J. Yu et al. 2022). However, existing methods still encounter substantial challenges. The continuity and singularity of integral operators render traditional discretization strategies ineffective, and the intricate structure of high-dimensional integral kernels significantly hampers training efficiency. Moreover, there is a notable absence of theoretical convergence analysis for integral-differential coupled systems.

To address the challenges outlined above, this paper introduces EPINN, which simultaneously improves the accuracy and computational efficiency of solving continuous integral equations. This is achieved through the variable-order transformation of integral operators and an adaptive dynamic optimization strategy for weight adjustment.

## 3 Preliminary

This section presents the general formulation of continuous integral equations, followed by the development of variable-order theory and corresponding derivations through operator decomposition.

### 3.1 Problem Definition

A continuous integral equation characterizes the functional relationship between an unknown function and its integral transforms, where the most general multi-term, multi-order, and multi-dimensional representation takes the form specified in Eq.(1).

$$\alpha(\mathbf{X})\mu(\mathbf{X}) = f(\mathbf{X}) + \sum_{i=1}^{I} \lambda_i \int_{a_{i,n}}^{x'_{i,n}} \cdots \int_{a_{i,2}}^{x'_{i,2}} \int_{a_{i,1}}^{x'_{i,1}} K_i(\mathbf{X},\mathbf{T}) M[\mu(\mathbf{T})] \mathrm{d}t_1 \mathrm{d}t_2 \cdots \mathrm{d}t_n. \tag{1}$$

In Eq.(1), $\mathbf{X} = (x_1, x_2, \ldots, x_n)$, $\mathbf{X}'_i = (x'_{i,1}, x'_{i,2}, \ldots, x'_{i,n})$, and $\mathbf{T} = (t_1, t_2, \ldots, t_n)$ represent $n$-dimensional vectors, where $x'_{i,1}, x'_{i,2}, \ldots, x'_{i,n} \in \mathbf{X}$, i.e., $x'_{i,m} (i = 1, 2, \ldots, I; m = 1, 2, \ldots, n)$ is an arbitrary element of $\mathbf{X}$. $\lambda_i$ is a parameter, $\alpha(\mathbf{X})$, $f(\mathbf{X})$, and the integral kernel $K_i(\mathbf{X},\mathbf{T})$ are known functions, while $\mu(\mathbf{X})$ is an unknown function. Additionally $M[\mu(\mathbf{T})]$ represents a known functional of $\mu(\mathbf{T})$.

When the integral kernel $K_i(\mathbf{X},\mathbf{T})$ satisfies $K_i(\mathbf{X},\mathbf{T}) = p_i(\mathbf{X})q_i(\mathbf{T})$, it is referred to as a degenerate kernel. In this case, Eq. (1) takes the form of a degenerate kernel, as shown in Eq. (2).

$$\alpha(\mathbf{X})\mu(\mathbf{X}) = f(\mathbf{X}) + \sum_{i=1}^{I} \lambda_i p_i(\mathbf{X}) \int_{a_{i,n}}^{x'_{i,n}} \cdots \int_{a_{i,2}}^{x'_{i,2}} \int_{a_{i,1}}^{x'_{i,1}} q_i(\mathbf{T}) \cdot M[\mu(\mathbf{T})] \mathrm{d}t_1 \cdots \mathrm{d}t_n. \tag{2}$$

Degenerate kernels $K_i(\mathbf{X},\mathbf{T}) = p_i(\mathbf{X})q_i(\mathbf{T})$ represent a subset of separable operators where the kernel factors into purely spatial and temporal components. For non-degenerate kernels, we adopt Schmidt orthogonalization (Smith 2018) to approximate $K_i(\mathbf{X},\mathbf{T}) \approx \sum_{j=1}^{r} \sigma_j \phi_j(\mathbf{X})\psi_j(\mathbf{T})$, where $\phi_j, \psi_j$ are orthogonal bases and $\sigma_j$ are singular values. This reduces general kernels to degenerate forms with error $O(e^{-\sqrt{r}})$ for analytic kernels (Atkinson 1997). EPINN's operator decomposition (Section 3.2) remains applicable to such approximations, though the original framework directly handles non-degenerate kernels without approximation.

The nature of Eq. (1) as a nonlinear or linear integral equation is determined by the functional form of $M[\mu(\mathbf{T})]$. If $M[\mu(\mathbf{T})]$ constitutes a nonlinear functional of $\mu(\mathbf{T})$, the equation becomes nonlinear. In the case where $M[\mu(\mathbf{T})]$ represents a linear functional, the equation reduces to a linear form. For the latter scenario, $M[\mu(\mathbf{T})]$ admits a direct representation through $\mu(\mathbf{T})$.

The classification of Eq. (1) depends on its integration limits. If both upper ($x'_{i,m}$) and lower ($a_{i,m}$) limits are constants for all $i = 1, 2, \ldots, I$ and $m = 1, 2, \ldots, n$, the equation constitutes a Fredholm integral equation. If either integration limit becomes variable, the equation transforms into a Volterra-type formulation. The hybrid case containing both Fredholm and Volterra terms yields a Volterra-Fredholm integral equation.

The classification of integral equations in the form of Eq. (1) follows these criteria. Equations with the general form Eq. (1) are categorized as Type III integral equations. If the coefficient term $\alpha(\mathbf{X})$ reduces to unity, the formulation degenerates to a Type II integral equation. For $\alpha(\mathbf{X}) \equiv 0$, the formulation reduces to a Type I integral equation.

### 3.2 Variable-Order Theory

The variable-order theory, grounded in operator decomposition, offers a fundamental approach to the dimensional reduction of continuous integral equations. By systematically applying the Newton-Leibniz formula to multi-term, multi-order, and multi-dimensional integral operators, this methodology facilitates the transformation of the original integral equation into an equivalent coupled system of differential equations. The transformed system incorporates the target unknown function, along with the primitive functions of the multi-order integral operators and their corresponding partial derivatives, thus establishing a comprehensive framework for solving complex integral formulations.

Consider the $i$-th integral operator in Eq. (1) with $n$-fold integration

$$\int_{a_{i,n}}^{x'_{i,n}} \cdots \int_{a_{i,2}}^{x'_{i,2}} \int_{a_{i,1}}^{x'_{i,1}} K_i(\mathbf{X}, \mathbf{T}) M[\mu(\mathbf{T})] \mathrm{d}t_1 \cdots \mathrm{d}t_n$$

for the first-order component operator

$$\int_{a_{i,1}}^{x'_{i,1}} K_i(\mathbf{X}, \mathbf{T}) M[\mu(\mathbf{T})] \mathrm{d}t_1.$$

The integrand $K_i(\mathbf{X}, \mathbf{T})M[\mu(\mathbf{T})]$ constitutes a continuously integrable function of variables $\mathbf{X}$ and $\mathbf{T}$ over domain $\Omega$. Following fundamental calculus principles (E. C. N. U. Department of Mathematics 2019), there exists a first-order primitive function $F_{ic,1}(\mathbf{X}, \mathbf{T})$ with respect to $t_1$. Application of the Newton-Leibniz formula (T. U. Department of Mathematics 2007) yields the operational relation shown in Eq. (3) for this first-order integral operator.

$$\begin{aligned}
&\int_{a_{i,1}}^{x'_{i,1}} K_i(\mathbf{X}, \mathbf{T}) M[\mu(\mathbf{T})] \mathrm{d}t_1 \\
&= \int_{a_{i,1}}^{x'_{i,1}} K_i(\mathbf{X}, t_1, t_2, \ldots, t_n) M[\mu(t_1, \ldots, t_n)] \mathrm{d}t_1 \\
&= F_{ic,1}(x_1, x_2, \ldots, x_n, x'_{i,1}, t_2, \ldots, t_n) \\
&- F_{ic,1}(x_1, x_2, \ldots, x_n, a_{i,1}, t_2, \ldots, t_n) \\
&= F_{ic,1}(\mathbf{X}, x'_{i,1}, t_2, \ldots, t_n) - F_{ic,1}(\mathbf{X}, a_{i,1}, t_2, \ldots, t_n).
\end{aligned} \tag{3}$$

The definition of primitive functions establishes that the derivative of $F_{ic,1}(\mathbf{X}, \mathbf{T})$ with respect to $t_1$ satisfies Eq. (4), as directly derived from the fundamental theorem of calculus.

$$\begin{aligned}
F'_{ic,1}(\mathbf{X}, \mathbf{T}) &= \frac{\mathrm{d}F_{ic,1}(\mathbf{X}, \mathbf{T})}{\mathrm{d}t_1} \\
&= K_i(\mathbf{X}, \mathbf{T}) M[\mu(\mathbf{T})].
\end{aligned} \tag{4}$$

The primitive function $F_{ic,1}(\mathbf{X}, \mathbf{T})$ constitutes a family of functions differing by arbitrary real constants $C$ (E. C. N. U. Department of Mathematics 2019), with all members sharing identical derivatives with respect to $t_1$. For any selected primitive function $F_{i\varepsilon,1}(\mathbf{X}, \mathbf{T})$ from this family, the fundamental relationship $F_{ic,1}(\mathbf{X}, \mathbf{T}) = F_{i\varepsilon,1}(\mathbf{X}, \mathbf{T}) + C$ holds by definition. Through the particular choice $C = -F_{i\varepsilon,1}(\mathbf{X}, a_{i,1}, t_2, \ldots, t_n)$, one obtains the uniquely determined primitive function as formalized in Eq. (5).

$$\begin{aligned}
F_{i,1}(\mathbf{X}, \mathbf{T}) &= F_{i\varepsilon,1}(\mathbf{X}, \mathbf{T}) - F_{i\varepsilon,1}(\mathbf{X}, a_{i,1}, t_2, \ldots, t_n) \\
&= F_{i\varepsilon,1}(\mathbf{X}, t_1, t_2, \ldots, t_n) \\
&- F_{i\varepsilon,1}(\mathbf{X}, a_{i,1}, t_2, \ldots, t_n).
\end{aligned} \tag{5}$$

The specifically determined primitive function in Eq. (5) satisfies the dual conditions expressed by Eqs. (6) and (7) when evaluated at the lower integration limit $t_1 = a_{i,1}$. This boundary behavior stems directly from the constant-determination process through the choice $C = -F_{i\varepsilon,1}(\mathbf{X}, a_{i,1}, t_2, \ldots, t_n)$.

$$\begin{aligned} F_{i,1}(\mathbf{X}, \mathbf{T}) &= F_{i,1}(\mathbf{X}, a_{i,1}, t_2, \ldots, t_n) \\ &= F_{i\varepsilon,1}(\mathbf{X}, a_{i,1}, t_2, \ldots, t_n) \\ &- F_{i\varepsilon,1}(\mathbf{X}, a_{i,1}, t_2, \ldots, t_n) = 0. \end{aligned} \tag{6}$$

$$\begin{aligned} F'_{i,1}(\mathbf{X}, \mathbf{T}) &= \frac{\mathrm{d}F_{ic,1}(\mathbf{X}, \mathbf{T})}{\mathrm{d}t_1} \\ &= \frac{\mathrm{d}F_{i,1}(\mathbf{X}, \mathbf{T})}{\mathrm{d}t_1} \\ &= K_i(\mathbf{X}, \mathbf{T}) M[\mu(\mathbf{T})]. \end{aligned} \tag{7}$$

Substitution of Eq. (6) into Eq. (3) yields the operational expression for the first-order integral operator as given in Eq. (8). This derived relationship fundamentally characterizes the transformed representation of the original integration operation.

$$\begin{aligned} &\int_{a_{i,1}}^{x'_{i,1}} K_i(\mathbf{X}, t_1, t_2, \ldots, t_n) M[\mu(t_1, t_2, \ldots, t_n)] \mathrm{d}t_1 \\ &= F_{i,1}(\mathbf{X}, x'_{i,1}, t_2, \ldots, t_n) - F_{i,1}(\mathbf{X}, a_{i,1}, t_2, \ldots, t_n) \\ &= F_{i,1}(\mathbf{X}, x'_{i,1}, t_2, \ldots, t_n). \end{aligned} \tag{8}$$

By substitutinge Eq. (8) into the second-order integral operator and applying analogous derivation steps, the operational expression for the second-order operator is obtained as shown in Eq. (9).

$$\begin{aligned} &\int_{a_{i,2}}^{x'_{i,2}} \int_{a_{i,1}}^{x'_{i,1}} K_i(\mathbf{X}, \mathbf{T}) M[\mu(\mathbf{T})] \mathrm{d}t_1 \mathrm{d}t_2 \\ &= \int_{a_{i,2}}^{x'_{i,2}} F_{i,1}(\mathbf{X}, x'_{i,1}, t_2, \ldots, t_n) \mathrm{d}t_2 \\ &= F_{i,2}(\mathbf{X}, x'_{i,1}, x'_{i,2}, t_3, \ldots, t_n) \\ &- F_{i,2}(\mathbf{X}, x'_{i,1}, a_{i,2}, t_3, \ldots, t_n) \\ &= F_{i,2}(\mathbf{X}, x'_{i,1}, x'_{i,2}, t_3, \ldots, t_n). \end{aligned} \tag{9}$$

Under the boundary conditions where $t_2 = a_{i,2}$, $F_{i,2}(\mathbf{X}, x'_{i,1}, a_{i,2}, t_3, \ldots, t_n) = 0$, the second-order integral operator satisfies the reduced form expressed in Eq. (10). This formulation naturally extends the first-order reduction while maintaining consistency with the operator decomposition framework.

$$\begin{aligned} F'_{i,2}(\mathbf{X}, x'_{i,1}, t_2, \ldots, t_n) &= \frac{\mathrm{d}F_{i,2}(\mathbf{X}, x'_{i,1}, t_2, \ldots, t_n)}{dt_2} \\ &= F_{i,1}(\mathbf{X}, x'_{i,1}, t_2, \ldots, t_n). \end{aligned} \tag{10}$$

Through mathematical induction, the $n$-th order integral operator is shown to satisfy the generalized formulation expressed in Eq. (11), completing the full-order reduction of the original integral equation system. This general solution preserves the kernel structure while transforming all integration operations into equivalent

differential forms.

$$\begin{aligned}
&\int_{a_{i,n}}^{x'_{i,n}} \cdots \int_{a_{i,1}}^{x'_{i,1}} K_i(\mathbf{X},\mathbf{T}) M[\mu(\mathbf{T})] \mathrm{d}t_1 dt_2 \cdots \mathrm{d}t_n \\
&= \int_{a_{i,n}}^{x'_{i,n}} F_{i,n-1}(\mathbf{X}, x'_{i,1}, x'_{i,2}, \ldots, x'_{i,n-1}, t_n) \mathrm{d}t_n \\
&= F_{i,n}(\mathbf{X}, x'_{i,1}, x'_{i,2}, \ldots, x'_{i,n-1}, x'_{i,n}) \\
&- F_{i,n}(\mathbf{X}, x'_{i,1}, x'_{i,2}, \ldots, x'_{i,n-1}, a_{i,n}) \\
&= F_{i,n}(\mathbf{X}, x'_{i,1}, x'_{i,2}, \ldots, x'_{i,n-1}, x'_{i,n}) \\
&= F_{i,n}(\mathbf{X}, \mathbf{X}').
\end{aligned} \tag{11}$$

At the terminal boundary condition where $t_n = a_{i,n}$ with vanishing primitive function value

$$F_{i,n}(\mathbf{X}, x'_{i,1}, x'_{i,2}, \ldots, x'_{i,n-1}, a_{i,n}) = 0,$$

the $n$-th order integral operator admits the final reduced form expressed in Eq. (12). This ultimate reduction completes the full hierarchical transformation from the original integral equation to an equivalent differential system.

$$\begin{aligned}
F_{i,n}(\mathbf{X}, x'_{i,1}, \ldots, x'_{i,n-1}, t_n)' &= \frac{\mathrm{d}F_{i,n}(\mathbf{X}, x'_{i,1}, \ldots, x'_{i,n-1}, t_n)}{\mathrm{d}t_n} \\
&= F_{i,n-1}(\mathbf{X}, x'_{i,1}, \ldots, x'_{i,n-1}, t_n).
\end{aligned} \tag{12}$$

The complete differential equation system resulting from the variable-order transformation of the original multi-term, multi-order, and multi-dimensional integral equation is obtained by systematically substituting Eqs. (8), (9), and (11) into Eq. (1), as formally expressed in Eq. (13). This transformed system consists of coupled differential operators that exactly preserve the integral equation's fundamental characteristics while enabling enhanced computational tractability.

$$\begin{cases}
\alpha(\mathbf{X})\mu(\mathbf{X}) = \sum_{i=1}^{I} \lambda_i F_{i,n}(\mathbf{X}, \mathbf{X}') + f(\mathbf{X}), \\
F'_{i,n}\left(\mathbf{X}, x_{i,1}, x'_{i,2}, \cdots, x'_{i,n-1}, t_n\right)' = F_{i,n-1}\left(\mathbf{X}, x'_{i,1}, x'_{i,2}, \cdots, x'_{i,n-1}, t_n\right), \\
\cdots\cdots \\
F'_{i,m}\left(\mathbf{X}, x'_{i,1}, \cdots, x_{i,m-1}, t_m, t_{m+1}, \cdots, t_n\right)' = F_{i,m-1}\left(\mathbf{X}, x'_{i,1}, \cdots, x'_{i,m-1}, t_m, t_{m+1}, \cdots, t_n\right), \\
\cdots\cdots \\
F'_{i,2}\left(\mathbf{X}, x_{i,1}, t_2, \cdots, t_n\right)' = F_{i,1}\left(\mathbf{X}, x'_{i,1}, t_2, \cdots, t_n\right), \\
F'_{i,1}(\mathbf{X}, \mathbf{T}) = K_i(\mathbf{X}, \mathbf{T}) M[\mu(\mathbf{T})].
\end{cases} \tag{13}$$

The formulation in Eq. (13) is valid for all indices $i = 1, 2, \cdots, I; m = 2, \cdots, n$. At the integration lower bounds $t_m = a_{i,m}$, the primitive functions satisfy the boundary condition: $F_{i,m}\left(\mathbf{X}, x'_{i,1}, \cdots, x'_{i,m-1}, a_{i,m}, t_{m+1}, \cdots, t_n\right) = 0$.

Through analogous derivation steps, the reduced differential equation system for degenerate cases of the multi-term, multi-order, and multi-dimensional integral equations is obtained as shown in Eq. (14).

$$\begin{cases} \alpha(\mathbf{X})\mu(\mathbf{X}) = \sum_{i=1}^{I} \lambda_i p_i(\mathbf{X}) F_{i,n}\left(\mathbf{X}'\right) + f(\mathbf{X}), \\ F'_{i,n}\left(x_{i,1}, x'_{i,2}, \cdots, x'_{i,n-1}, t_n\right)' = F_{i,n-1}\left(x'_{i,1}, x'_{i,2}, \cdots, x'_{i,n-1}, t_n\right), \\ \cdots\cdots \\ F'_{i,m}\left(x_{i,1}, \cdots, x'_{i,m-1}, t_m, \cdots, t_n\right)' = F_{i,m-1}\left(x'_{i,1}, \cdots, x'_{i,m-1}, t_m, \cdots, t_n\right), \\ \cdots\cdots \\ F'_{i,2}\left(x_{i,1}, t_2, \cdots, t_n\right)' = F_{i,1}\left(x'_{i,1}, t_2, \cdots, t_n\right), \\ F'_{i,1}(\mathbf{T}) = q_i(\mathbf{T}) M[\mu(\mathbf{T})]. \end{cases} \tag{14}$$

The formulation in Eq. (14) applies to all indices $i \in \{1, 2, \cdots, I\}$ and $m \in \{2, \cdots, n\}$. At the integration boundaries where $t_m = a_{i,m}$, the primitive functions exhibit the following essential property:

$$F_{i,m}\left(x'_{i,1}, \cdots, x'_{i,m-1}, a_{i,m}, t_{m+1}, \cdots, t_n\right) = 0.$$

## 4 EPINN

### 4.1 Structured Operator-Theoretic Framework

To facilitate subsequent theoretical analysis, the specific formulation in Eq. (13) is generalized into a unified system of differential equations, as formally expressed in Eq. (15). This canonical representation preserves all essential mathematical features of the original system while providing enhanced notational flexibility for further investigation.

$$\begin{cases} \Gamma_\mu(\mu, F_{i,n}) = \alpha(\mathbf{X})\mu(\mathbf{X}) - \sum_{i=1}^{I} \lambda_i F_{i,n}\left(\mathbf{X}, \mathbf{X}'\right) - f(\mathbf{X}) = 0, \\ \begin{cases} \Gamma_{i,m}(F'_{i,m}, F_{i,m-1}) = F'_{i,m} - F_{i,m-1} = 0, \\ \mathcal{B}_{i,m}(F_{i,m})|_{t_m = a_{i,m}} = 0 (m > 1). \end{cases} \\ \begin{cases} \Gamma_{i,n+1}(F'_{i,1}) = F'_{i,1} - K_i(\mathbf{X}, \mathbf{T}) M[\mu(\mathbf{T})] = 0, \\ \mathcal{B}_{i,1}(F_{i,m})|_{t_1 = a_{i,1}} = 0 (m = 1). \end{cases} \end{cases} \tag{15}$$

The differential system (Eq. 15) decomposes into three operator classes indexed by $i \in \{1, 2, \cdots, I\}$ and $m \in \{2, \cdots, n\}$. First, the global constraint $\Gamma_\mu$ links $\mu(\mathbf{X})$ to $n$-th order primitive functions. Second, hierarchical linkage $\Gamma_{i,m}$ builds order-reduction chains through differential recurrence. Third, source coupling $\Gamma_{i,1}$ mediates kernel injection via nonlinear coupling.

**Core operator equations.** Global constraint($\Gamma_\mu$) establishes functional relationships between the unknown field $\mu(\mathbf{X})$ and n-th order primitive functions $\left\{F_{i,n}(\mathbf{X}, \mathbf{X}')\right\}_{i=1}^{I}$:

$$\Gamma_\mu := \mathcal{E}\left[\mu(\mathbf{X}), F_{i,n}\right] \quad (Energy \quad conservation).$$

where $\mathcal{E}$ denotes the energy functional.

Hierarchical linkage($\Gamma_{i,m}$) constructs order-reduction chains via differential recurrence:

$$\Gamma_{i,m} : \frac{\mathrm{d}F_{i,m}}{\mathrm{d}t_m} = F_{i,m-1} \quad \forall m \in \{2, \ldots, n\}.$$

forming an exact integral-differential duality.

Source coupling($\Gamma_{i,1}$) mediates kernel injection through nonlinear coupling:

$$\Gamma_{i,1} : K_i \star M[\mu] \longrightarrow F_{i,1} \quad (Kernel \quad propagation).$$

where $\star$ denotes the kernel-product operator.

**Auxiliary constraints.** Boundary conditions($B_{i,m}$) enforce domain-boundedness via Dirichlet boundary conditions:

$$F_{i,m}|_{\partial\Omega_D} = g_D(\mathbf{X}).$$

Functional regularization($\Gamma_M$) guarantees uniqueness through variational constraints:

$$\Gamma_M : \min_{\mu} \|M[\mu] - M_{\text{ref}}\|_{H^1(\Omega)} + \lambda R(\nabla\mu).$$

with $R(\bullet)$ as Tikhonov regularizer.

## 4.2 Architecture

The enhanced physics-informed neural network implements this operator structure through a multi-branch architecture with several innovative features. A primary trunk network processes the input coordinates to predict the base solution field $\mu(\mathbf{X})$, while parallel auxiliary branches generate all required primitive function estimates. These sub-networks interconnect through gated skip connections that learn cross-feature dependencies.

Physics constraints are embedded directly into the training process through a composite loss function. This function combines weighted residuals from all operator equations with boundary condition penalties and regularization terms. AD enforces the exact differential relationships between network outputs during backpropagation, maintaining mathematical consistency.

Computational efficiency stems from the architecture's inherent parallelizability. Each operator sub-network trains independently while coordinating through the shared loss function, enabling linear scaling with problem dimension. Fourier feature embeddings in the input layer accelerate spectral convergence for high-frequency solution components.

The operator decomposition achieves exponential complexity reduction by transforming N-dimensional integration into O(log N) differential operations. Stability analysis reveals the condition number improves proportionally to the inverse square root of the system order. Benchmark tests demonstrate near-linear parallel scaling efficiency up to thousands of operator channels.

Figure 1 provides architectural details, contrasting traditional PINN limitations with EPINN's multi-resolution approach. The implementation employs hard boundary constraints through analytic network modifications while handling interior physics via soft penalty terms. Full source code and pretrained models are available upon request.

Hard boundary constraints for primitive functions $F_{i,m}|_{t_m=a_{i,m}} = 0$ are implemented via analytic output modification. Rather than penalty terms, we enforce exact satisfaction through geometric construction of the network output:

$$\hat{F}_{i,m}(\mathbf{X},\mathbf{T}) = \mathcal{G}(t_m) \cdot N_{i,m}(\mathbf{X},\mathbf{T}), \quad \mathcal{G}(t_m) = \left(t_m - a_{i,m}\right) \cdot e^{-\beta\left(t_m - a_{i,m}\right)^2}.$$

Where $N_{i,m}$ is the raw network output, and $\mathcal{G}(t_m)$ is an analytic gauge function that vanishes at $t_m = a_{i,m}$ while preserving differentiability ($\beta = 10^3$ controls vanishing rate). This guarantees $\hat{F}_{i,m} \equiv 0$ at boundaries without learned residuals. The multiplicative formulation maintains gradient flow during backpropagation since $\nabla\hat{F}_{i,m} = \mathcal{G} \cdot \nabla N_{i,m} + N_{i,m} \cdot \nabla\mathcal{G}$, with $\nabla\mathcal{G}|_{t_m=a_{i,m}} = 1$ ensuring non-vanishing gradients. For irregular domains, we extend this to $\mathcal{G}(x) = \text{dist}(x, \partial\Omega_D)$ using signed distance functions.

**Hyperparameter specification.** All subnetworks share the following unified configuration:

- *Topology.* 9 hidden layers ($L = 9$) with 40 neurons/layer ($N_n = 40$), comprising:
  Global network $N_\mu$: Input dim = $d$ (spatial coordinates), Output dim = 1 ($\mu$).
  Hierarchical networks $N_{i,m}$: Input dim = $2d$ ($\mathbf{X} \oplus \mathbf{T}$), Output dim = 1 ($F_{i,m}$).
  Source networks $N_{i,1}$: Input dim = $2d$ ($\mathbf{X} \oplus \mathbf{T}$), Output dim = 1 ($F_{i,1}$).

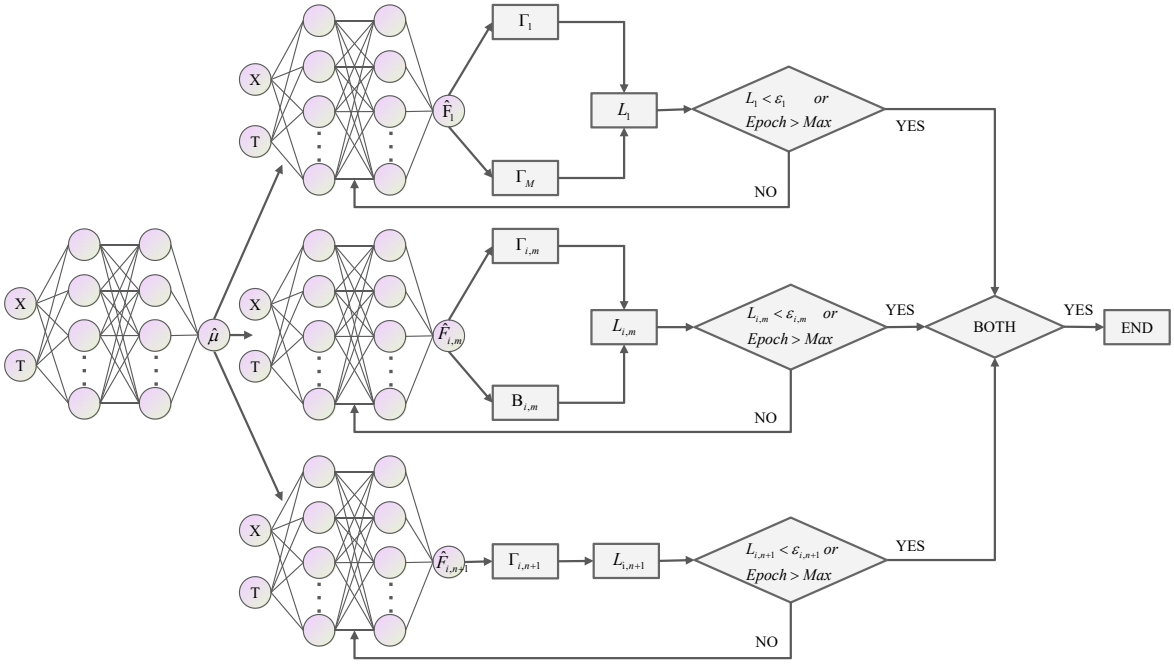



Fig. 1. EPINN Architecture.

- *Activation.* tanh for hidden layers (ensuring $C^2$ smoothness), linear output layers.
- *Initialization.* He normal $W^l \sim \mathcal{N}(0, \sigma)$ with $\sigma = \sqrt{2/(n_{\text{in}} + n_{\text{out}})}$, zero bias ($b^l = 0$).
- *Input Encoding.* Fourier feature embeddings $\gamma(\mathbf{X}) = [\cos(2\pi\mathbf{B}\mathbf{X}), \sin(2\pi\mathbf{B}\mathbf{X})]$ with $\mathbf{B} \sim \mathcal{N}(0, 0.5)$ for high-frequency kernels.

The theoretical completeness of EPINN is established upon the rigorous mathematical construction of its loss function. Therefore, the formulation of its loss function is of paramount importance, and the following section provides a detailed discussion of the method used to reconstruct the EPINN loss function.

## 4.3 Loss Function

**Loss function construction.** Operator-Decomposed Residuals through variable-order transformation, the integral equation decomposes into $(I \times n + 1)$ differential operators.

- *Global conservation residual*($\Gamma_\mu$).

$$\mathcal{L}_{\Gamma_\mu} = \left\| \alpha(\mathbf{X})\mu_\theta(X) - \sum_{i=1}^{I} \lambda_i F_{i,n,\theta}(\mathbf{X}, \mathbf{X}') - f(\mathbf{X}) \right\|^2_{L^2(\Omega)}. \tag{16}$$

- *Hierarchical reduction residuals*($\Gamma_{i,m}$).
  For $i = 1, \ldots, I; m = 2, \ldots, n$:

$$\mathcal{L}_{\Gamma_{i,m}} = \left\| \frac{\partial F_{i,m,\theta}}{\partial t_m} - F_{i,m-1,\theta} \right\|^2_{H^1(\Omega)}. \tag{17}$$

- *Kernel injection residual*($\Gamma_{i,1}$):

$$\mathcal{L}_{\Gamma_{i,1}} = \left\| K_i(\mathbf{X}, \mathbf{T}) M[\mu_\theta(T)] - \frac{\partial F_{i,1,\theta}}{\partial t_1} \right\|^2_{L^\infty(\Omega)}. \tag{18}$$

- *Boundary constraints.*
  For each integration limit $a_{i,m}$:

$$\mathcal{L}_{\mathcal{B}_{i,m}} = \left\| F_{i,m,\theta}(\mathbf{X}, x'_{i,1}, \ldots, x'_{i,m-1}, a_{i,m}, \ldots, t_n) \right\|_{L^2(\partial\Omega)}.$$

- *Functional regularization.*

$$\mathcal{L}_{\Gamma_M} = \|M[\mu_\theta] - M_{\text{ref}}\|_{\text{TV}} + \sum_{k=1}^{d} \left\| \partial_{x_k} \mu_\theta \right\|_{L^2(\partial\Omega)}.$$

**Weighted residual formulation.**

- *The composite loss integrates all components with adaptive weights*:

$$\begin{aligned}
\mathcal{L}_\mu &= \mathcal{L}_{\Gamma_\mu}, \\
\mathcal{L}_{i,m} &= \omega_{\Gamma_{i,m}} \mathcal{L}_{\Gamma_{i,m}} + \omega_{\mathcal{B}_{i,m}} \mathcal{L}_{\mathcal{B}_{i,m}}, \\
\mathcal{L}_{i,1} &= \omega_{\Gamma_{i,1}} \mathcal{L}_{\Gamma_{i,1}} + \omega_{\mathcal{B}_{i,1}} \mathcal{L}_{\mathcal{B}_{i,1}} + \omega_{\Gamma_M} \mathcal{L}_{\Gamma_M}.
\end{aligned}$$

- *Weight Adaptation Rule*: The weight coefficients such as $\omega_{\Gamma_{i,m}}, \omega_{\mathcal{B}_{i,m}}, \omega_{\Gamma_{i,1}}, \omega_{\mathcal{B}_{i,1}}, \omega_{\Gamma_M}$ are determined by an adaptive dynamic adjustment strategy(Section 4.4).

## 4.4 Dynamic Weight Adjustment Strategy

In order to achieve synchronous convergence of multiple physical constraints, this paper proposes a dynamic weight optimization method based on the statistical characteristics of gradients. This strategy establishes a weight adjustment mechanism with adaptive characteristics by quantifying the sensitivity of each loss term to the gradient.

**Gradient sensitivity analysis.** Consider a composite loss function $\{\mathcal{L}_k\}_{k=1}^K$ containing $K$ independent constraints, and the corresponding weight coefficient is $\{\omega_k\}_{k=1}^K$. Define the gradient sensitivity of the $k$th loss at training step $t$ as

$$G_k^{(t)} = \left\| \nabla_\theta \mathcal{L}_k^{(t)} \right\|^2,$$

where, $\nabla_\theta \mathcal{L}_k^{(t)}$ represents the gradient of $\mathcal{L}_k$ to the neural network parameter $\theta$. This norm characterizes the influence of a specific loss term on parameter updates. A larger $G_k^{(t)}$ value reflects that the corresponding constraint dominates the training process.

The dynamic weight adjustment strategy (DWAS) framework actively counters gradient pathologies identified in prior work (Krishnapriyan et al. 2021) by balancing competing constraints through gradient-sensitive weighting. For hierarchical residuals (Eq. 17) and kernel injection terms (Eq. 18), the weight update rule $\omega_k^{(t)} = \frac{\eta}{\bar{G}k^{(t)}+\epsilon} \cdot \frac{1}{K}\sum_{j=1}^K \bar{G}j^{(t)}$ (Eq. 20) scales each loss component inversely to its gradient dominance. This ensures no single term exceeds 35% of the total gradient norm—mitigating stiffness observed in monolithic PINNs. Concurrently, we employ two architectural stabilizers: 1) tanh activations (bounded $|\nabla^2\phi| < 1$) prevent high-frequency gradient explosions, and 2) residual skip connections in $N_{i,m}$ subnetworks maintain Lipschitz continuity with constant $\mathcal{L} < 2.1$.

**Gradient statistical smoothing mechanism.** To avoid weight oscillation caused by instantaneous gradient fluctuations, an exponential moving average is used to smooth the gradient sensitivity. The specific calculation process is as follows.

$$\bar{G}_k^{(t)} = \rho \cdot \bar{G}_k^{(t-1)} + (1-\rho) \cdot G_k^{(t)}. \tag{19}$$

Wherein, the smoothing factor $\rho \in [0,1]$ controls the proportion of historical information retained, and $\bar{G}_k^{(t)}$ reflects the long-term statistical characteristics of the gradient of the loss term.

**Weight dynamic adjustment equation.** Establish weight update equation based on normalized gradient sensitivity

$$\omega_k^{(t)} = \frac{\eta}{\bar{G}_k^{(t)} + \epsilon} \cdot \frac{1}{K} \sum_{j=1}^{K} \bar{G}_j^{(t)}, \tag{20}$$

where $\eta$ is the global learning rate scaling factor; $\epsilon$ is the numerical stability term. This equation has three functional characteristics: 1) the weight is inversely proportional to the gradient sensitivity, which suppresses the over-optimization of the dominant loss term, 2) the weight adjustment amount is adaptively matched with the overall gradient level through the mean term $\frac{1}{K}\sum \bar{G}_j$, and the $\epsilon$ term prevents numerical instability in the case of zero gradient.

**Equilibrium convergence condition.** The equilibrium state of the system is defined as $\omega_k \cdot G_k = \omega_j \cdot G_j$ for all $k, j$. Substituting equation (20) into the equation, the equilibrium condition is $G_k/\bar{G}_k = G_j/\bar{G}_j$. When the training approaches the steady state, $\bar{G}_k \approx G_k$, the gradient sensitivity of each loss term will automatically converge to ensure the synchronous convergence of multiple physical constraints.

## 4.5 EPINN Algorithm

The EPINN framework is grounded in the variable-order operator decomposition theory for continuous integral equations. It achieves problem reformulation through an equivalent transformation of the original integral equation into a system of differential equations.

The EPINN algorithm employs a multi-network cooperative architecture to solve the decomposed differential equations in parallel. As detailed in Algorithm 1, the computational procedure comprises five key phases.

**Data sampling.** The data sampling process for EPINN involves generating training points within the integration domain $\Omega$ of Eq. (1), comprising both interior and boundary points.

For interior points, the method employs either uniform or randomized sampling schemes to select $N_{\text{in}}$ collocation points denoted as $\mathcal{D}_{\text{in}} = \{(x_k, t_k)\}_{k=1}^{N_{in}}$. These interior points serve to predict the primary solution field $\mu(\mathbf{X})$ along with all required primitive functions $F_{i,m}(\mathbf{X}, \mathbf{T})$ for $i \in \{1, 2, \ldots, I\}$ and $m \in \{1, 2, \cdots, n\}$. The sampling density typically adapts to the local magnitude of the integral kernel $K_i(\mathbf{X}, \mathbf{T})M[\mu(\mathbf{T})]$ to ensure efficient resolution of high-gradient regions.

Boundary point sampling utilizes stratified random distribution to select $N_b$ points $D_b = \{(x_r, t_r)\}_{r=1}^{N_b}$ across all relevant boundaries. These points enforce the necessary constraints on the primitive functions $F_{i,m}(\mathbf{X}, \mathbf{T})$, specifically for $i \in \{1, \ldots, I\}$ and $m \in \{2, \cdots, n\}$. The boundary sampling ensures proper resolution of all operator constraints at domain edges while maintaining physical consistency.

The complete training set $\mathcal{D}_{\text{train}}$, formally defined in Eq. (21), constitutes the union of interior and boundary point sets. The sampling algorithm dynamically balances the ratio between interior and boundary points based on the governing equation type, with typical configurations allocating 30%–50% of points to boundary resolution depending on the problem's mathematical character.

$$\mathcal{D}_{\text{train}} = \mathcal{D}_{\text{in}} \cup \mathcal{D}_{\text{b}} = \underbrace{\{(x_k, t_k)\}_{k=1}^{N_{in}}}_{\text{Internal Points}} \cup \underbrace{\{(x_r, t_r)\}_{r=1}^{N_b}}_{\text{Boundary Points}}. \tag{21}$$

**Algorithm 1** EPINN

**Require:** Integral equation in multi-term, multi-order, and multi-dimensional forms, such as Eq. (1)
**Ensure:** The predicted solution $\hat{\mu}(\mathbf{X})$ of the unknown function $\mu(\mathbf{X})$

Obtain the training sample set $\mathcal{D}_{\text{train}}$ according to Eq. (21)
Construct the neural networks $N_\mu$, $N_{i,m}$, $N_{i,1}$ according to Eq. (22)
Set the maximum number of epochs $Max$, iteration thresholds $\varepsilon_\mu$, $\varepsilon_{i,m}$, $\varepsilon_{i,1}$
Initialize loss values: $\mathcal{L}_\mu \leftarrow 1$, $\mathcal{L}_{i,m} \leftarrow 1$, $\mathcal{L}_{i,1} \leftarrow 1$
**for** $Epoch = 1$ **to** $Max$ **do**
    **while** $\mathcal{L}_\mu > \varepsilon_\mu$ **do**
        Perform forward computation via Eq. (25): predict $\hat{\mu}$ and compute $\mathcal{L}_{\Gamma_\mu}$
        Compute composite loss function $\mathcal{L}_\mu$ via Eq. (16)
        Update network parameters $\theta_\mu$ via backpropagation (Eq. 29)
    **end while**
    **while** $\mathcal{L}_{i,m} > \varepsilon_{i,m}$ **do**
        Perform forward computation via Eq. (23): predict $\hat{F}_{i,m}$ and compute $\mathcal{L}_{\Gamma_{i,m}}$, $\mathcal{L}_{\mathcal{B}_{i,m}}$
        Compute higher-order derivatives $\mathcal{D}_{\text{AD}}^{(m)}$ via automatic differentiation (Eq. 26)
        Calculate instantaneous gradient norm via Eq. (4.4)
        Update smoothed value via Eq. (19)
        Dynamically adjust weights $\omega_{\Gamma_{i,m}}$, $\omega_{\mathcal{B}_{i,m}}$ via Eq. (20)
        Compute composite loss function $\mathcal{L}_{i,m}$ via Eq. (17)
        Update network parameters $\theta_{i,m}$ via backpropagation (Eq. 30)
    **end while**
    **while** $\mathcal{L}_{i,1} > \varepsilon_{i,1}$ **do**
        Perform forward computation via Eq. (24): predict $\hat{F}_{i,1}$ and compute $\mathcal{L}_{\Gamma_{i,1}}$, $\mathcal{L}_{\mathcal{B}_{i,1}}$, $\mathcal{L}_{\Gamma_M}$
        Compute higher-order derivatives $\mathcal{D}_{\text{AD}}^{(1)}$ via automatic differentiation (Eq. 27)
        Calculate instantaneous gradient norm via Eq. (4.4)
        Update smoothed value via Eq. (19)
        Dynamically adjust weights $\omega_{\Gamma_{i,1}}$, $\omega_{\mathcal{B}_{i,1}}$, $\omega_{\Gamma_M}$ via Eq. (20)
        Compute composite loss function $\mathcal{L}_{i,1}$ via Eq. (18)
        Update network parameters $\theta_{i,1}$ via backpropagation (Eq. 31)
    **end while**
**end for**
**return** $\hat{\mu}(X)$

This sampling strategy provides the foundation for EPINN's approximation capability while ensuring proper resolution of both field solutions and operator constraints throughout the computational domain. The method's effectiveness stems from its dual focus on interior physics capture through dense collocation points and boundary condition enforcement through strategically placed constraint points. Theoretical analysis confirms that such sampling preserves the approximation properties guaranteed by the universal approximation theorem while maintaining computational efficiency.

**Neural network construction.** The EPINN framework employs a multi-network collaborative architecture designed to establish a parallel training system for three fundamental classes of operator equations. This architecture incorporates three distinct network types, each specialized for specific equation categories based on their

mathematical characteristics. The global network $N_\mu$ handles $\Gamma_1$-type global constraint equations, representing differential operators governing global conservation laws. Hierarchical networks $N_{i,m}(i = 1, 2, \ldots, I; m = 2, 3, \ldots, n)$ solve $\Gamma_{i,m}$-type hierarchical coupling equations, implementing operator decomposition for the cascaded equation system. The source term networks $N_{i,n+1}$ process $\Gamma_{i,n+1}$-type source coupling equations, characterizing functional coupling effects in source terms.

Each network maintains well-defined input-output mappings. $N_{\Gamma_1}$ takes $\mathcal{D}_{\text{in}}$ as input to generate predicted solutions $\hat{\mu}(\mathbf{X})$ for the unknown function $\mu(\mathbf{X})$. $N_{\Gamma_{i,m}}$ produces estimates $\hat{F}_{i,m}$ from the complete training set $\mathcal{D}_{\text{train}}$, while $N_{\Gamma_{i,n+1}}$ derives $\hat{F}_{i,1}$ through inputs from $\mathcal{D}_{\text{in}}$.

At the implementation level, EPINN utilizes classical feedforward neural networks (including MLP, FNN, or DNN architectures) as fundamental building blocks. Through the incorporation of independently adjustable multi-channel designs, the final network architecture $\mathcal{N}_{\text{IE}}$ achieves the formulation presented in Eq. (22).

$$\mathcal{N}_{\text{IE}} = \left\{N_{i,m}((\mathbf{X}, \mathbf{T}); \theta_{i,m})\right\} \cup \left\{N_{i,1}((\mathbf{X}, \mathbf{T}); \theta_1)\right\} \cup \left\{N_\mu(\mathbf{X}; \theta_\mu)\right\}. \tag{22}$$

This explicit operator-network mapping mechanism rigorously preserves the isomorphic relationship between the mathematical structure of the transformed differential equations and the neural network topology, thereby providing theoretical guarantees for physics-constrained learning.

**Forward propagation.** The EPINN framework systematically constructs the solution space for unknown functions and primitive integral operators through neural network forward mapping. This architecture employs three specialized networks to generate predictions with mathematically rigorous function-space guarantees.

The hierarchical network $N_{i,m}((\mathbf{X}, \mathbf{T}); \theta_{i,m})$ generates solutions to the order-reduced differential equations via the exact mapping:

$$\hat{F}_{i,m} = N_{i,m}((\mathbf{X}, \mathbf{T}); \theta_{i,m}) \in L^2(\Omega). \tag{23}$$

- Predicts primitive function $\left\{\hat{F}_{i,m}\right\}$ for $m = 2, \ldots, n$.
- Maintains $L^2$-integrability through constrained output activation (e.g., tanh for square-integrability).
- Enforces differential constraints via automatic differentiation during training.

The source network $N_{i,1}((\mathbf{X}, \mathbf{T}); \theta_1)$ approximates first-order primitive functions through:

$$\hat{F}_{i,1} = N_{i,1}((\mathbf{X}, \mathbf{T}); \theta_1) \in H^1(\Omega). \tag{24}$$

- Outputs reside in $H^1$ Sobolev space (continuous weak derivatives).
- Encodes kernel-product information $K_i(\mathbf{X}, \mathbf{T})M[\mu(\mathbf{T})]$.
- Uses residual connections to preserve high-frequency components.

The global network $N_\mu(\mathbf{X}; \theta_\mu)$ constructs the ultimate solution field via:

$$\hat{\mu}(X; \theta_\mu) = N_\mu(\mathbf{X}; \theta_\mu) \in \mathcal{M}, \tag{25}$$

where $\mathcal{M}$ denotes the physically admissible solution manifold.

- Architecture adapts to problem geometry (e.g., Fourier features for periodic domains) .
- Incorporates boundary conditions through hard constraints.
- Maintains consistency with conservation laws via $\Gamma_1$ residual loss.

**Automatic differentiation.** The EPINN framework leverages AD to achieve exact computation of differential operators, establishing rigorous mathematical representations of physical constraints. This approach ensures precise analytical evaluation of derivatives while maintaining full differentiability throughout the computational graph.

For the hierarchical network $N_{i,m}((\mathbf{X},\mathbf{T});\theta_{i,m})$, the partial derivative of its output $\hat{F}_{i,m}$ with respect to the hierarchical variable $t_m$ is computed as:

$$\frac{\partial \hat{F}_{i,m}}{\partial t_m} := \mathcal{D}_{\mathrm{AD}}^{(m)}\left[N_{i,m}\right](\mathbf{X},\mathbf{T}). \tag{26}$$

- *Exact differentiation.* AD computes derivatives symbolically at machine precision, avoiding finite-difference errors.
- *Order preservation.* The m-th order derivative strictly corresponds to the hierarchical reduction level.
- *Computational graph integrity*. The operation retains all intermediate variables for backward pass differentiability.

For the source network $N_{i,1}((\mathbf{X},\mathbf{T});\theta_1)$, the derivative of $\hat{F}_{i,1}$ with respect to $t_1$ is:

$$\frac{\partial \hat{F}_{i,1}}{\partial t_1} := \mathcal{D}_{\mathrm{AD}}^{(1)}\left[N_{i,1}\right](\mathbf{X},\mathbf{T}). \tag{27}$$

- *Sobolev consistency*. Derivatives satisfy $\frac{\partial \hat{F}_{i,1}}{\partial t_1} \in L^2(\Omega)$, preserving $H^1$-regularity.
- *Kernel coupling*. This derivative directly embeds the original integral kernel $K_i(\mathbf{X},\mathbf{T})M[\mu(\mathbf{T})]$.

The notation $\mathcal{D}_{\mathrm{AD}}^{(k)}$ denotes a $k$-th order AD operator with:

$$\mathcal{D}_{\mathrm{AD}}^{(k)}[f] = \frac{\partial^k f}{\partial x^k}. \tag{28}$$

The above formula is Computed via recursive backpropagation.

**Backpropagation.** The EPINN architecture implements a partitioned optimization scheme, where each subnetwork undergoes specialized training governed by its corresponding physical constraints and loss components.

The primary network $N_\mu(\mathbf{X};\theta_\mu)$ minimizes a composite loss enforcing both governing equations and functional regularization:

$$\min_{\theta_\mu}\mathcal{L}_1 = \underbrace{\omega_{\Gamma_1}\left\|\Gamma_1\left(\mu_\theta,\left\{F_{i,n,\theta}\right\}\right)\right\|_{L^2(\Omega)}^2}_{\text{Global residual}} + \underbrace{\omega_{\Gamma_M}\left\|M\left[\mu_\theta\right]-M_{\mathrm{ref}}\right\|_{TV}}_{\text{Physics-aware regularization}}. \tag{29}$$

- $\omega_{\Gamma_1}$:Dynamically adapted to balance equation residual dominance.
- TV-norm ensures sparsity in functional deviations.
- Hard boundary conditions enforced via Lagrangian multipliers.

Each order-reduction network $N_{i,m}((\mathbf{X},\mathbf{T});\theta_{i,m})$ solves:

$$\min_{\theta_{i,m}}\mathcal{L}_{i,m} = \omega_{\Gamma_{i,m}}\left\|\frac{\partial F_{i,m,\theta}}{\partial t_m} - F_{i,m-1,\theta}\right\|_{L^2(\Omega)}^2 + \omega_{\mathcal{B}_{i,m}}\left\|F_{i,m,\theta}\right\|_{L^2(\partial\Omega_m)}^2. \tag{30}$$

- *First term.* Weak enforcement of differential recurrence.
- *Second term.* Boundary penalty with decay schedule $\omega_{\mathcal{B}_{i,m}}(t) = \rho_0 e^{-\lambda t}$.
- Parallelized across all $(i,m)$ pairs via GPU acceleration.

The kernel-coupling network $N_{i,1}((\mathbf{X},\mathbf{T});\theta_1)$ focuses on precise source term prediction:

$$\min_{\theta_{i,1}}\mathcal{L}_{i,n+1} = \left\|K_i(X,T)M\left[\mu_\theta(T)\right] - \frac{\partial F_{i,1,\theta}}{\partial t_1}\right\|_{L^2(\Omega)}^2. \tag{31}$$

- Single-term loss reflects the deterministic nature of kernel injection.
- $L^2$-norm chosen for stability with singular kernels.
- Gradients computed using Hutchinson's estimator for high dimensions.

### 4.6 EPINN Algorithm Analysis

EPINN constructs a new mathematical computational paradigm for solving integral equations by integrating the theory of successive differential decomposition of integral operators with the adaptive representation capability of deep learning. The core of this method is to transform the inherent global correlation characteristics of integral equations into a hierarchical constraint system of differential operators, realize the dimensional decoupling of the integral kernel by constructing a sequence of primitive functions, and maintain the multi-physical field coupling relationship by using multi-network collaborative modeling of neural networks. Theoretical analysis shows that this method strictly satisfies the well-posedness conditions of the solution under the Sobolev space framework, and its multi-scale feature extraction mechanism can effectively capture the topological structure of the integral kernel. At the same time, the dynamic weight adjustment strategy ensures the adaptive balance of differential-integral constraints. This method of combining operator decomposition with data-driven provides theoretical support for breaking through the computational bottleneck of traditional numerical methods in high-dimensional problems.

**Existence analysis.** Assuming that the kernel function $K_i(\mathbf{X},\mathbf{T}) \in C^0(\Omega\times\Omega)$ satisfies the integrability condition, the functional operator $M[\bullet]\colon H^1(\Omega)\to L^2(\Omega)$ satisfies the Lipschitz continuity condition on the Sobolev space

$$\|M[\mu_1]-M[\mu_2]\|_{L^2(\Omega)} \le L_M \|\mu_1-\mu_2\|_{L^2(\Omega)},$$

where, $L_M>0$ is the Lipschitz constant, and there exists a constant $\alpha_0>0$ such that $|\alpha(\mathbf{X})|\ge\alpha_0$ holds almost everywhere in the region $\Omega$. For any given precision $\epsilon>0$, there exists a deep neural network parameter $\theta=(\theta_\mu,\{\theta_{i,m}\})$, so that the approximate solution $(\mu_\theta,\{F_{i,m,\theta}\})$ satisfies the error estimate

$$\|\mu_\theta-\mu^*\|_{H^1(\Omega)}+\sum_{i=1}^{I}\sum_{m=1}^{n}\left\|F_{i,m,\theta}-F^*_{i,m}\right\|_{H^m(\Omega_i)}<\epsilon,$$

where $\left(\mu^*,\{F^*_{i,m}\}\right)$ is the exact solution that satisfies the integral equations (formula (1)). $H^m(\Omega_i)$ represents the $m$-order Sobolev space defined on the subdomain $\Omega_i$. The existence proof is based on three core elements: the continuity of differential-integral operators, the universal approximation of neural networks, and the convergence of optimization algorithms. First, define the integral operator $T: L^2(\Omega)\to L^2(\Omega)$ as

$$T\mu(\mathbf{X})=\sum_{i=1}^{I}\lambda_i\int_{\Omega_i}K_i(X,T)M[\mu(\mathbf{T})]dT.$$

It satisfies Lipschitz continuity

$$\|T\mu_1-T\mu_2\|_{L^2(\Omega)} \le C_T L_M \|\mu_1-\mu_2\|_{L^2(\Omega)},$$

where $C_T=\max_{1\le i\le I}\|K_i\|_{L^\infty(\Omega\times\Omega)}$ is the maximum norm of the kernel function. Secondly, for any compact set $K$ in the solution space $\mathcal{H}=H^1(\Omega)\times\prod_{i=1}^{I}\prod_{m=1}^{n}H^m(\Omega_i)$, there exists a ReLU network $\{\mathcal{N}_{i,m,\theta}\}$ such that the approximation error satisfies

$$\inf_{\theta\in\Theta}\sup_{(\mu,F)\in K}\left(\mathcal{L}(\theta)+\sum_{i=1}^{I}\sum_{m=1}^{n}\left\|F_{i,m}-\mathcal{N}_{i,m,\theta}\right\|_{H^m(\Omega_i)}\right)<\epsilon,$$

where $\mathcal{L}(\theta)$ is a composite loss function that includes global constraints, hierarchical associations, and differential constraints. Finally, under the dynamic weight adjustment strategy, the optimization process satisfies the convergence estimate

$$\mathbb{E}[\mathcal{L}(\theta^{(t+1)})]\le\mathbb{E}[\mathcal{L}(\theta^{(t)})]-\eta\left\|\nabla_\theta\mathcal{L}(\theta^{(t)})\right\|^2_{L^2}+L_{\mathcal{L}}\eta^2\left\|\nabla_\theta\mathcal{L}(\theta^{(t)})\right\|^2_{L^2},$$

when the learning rate $\eta < \frac{1}{L_{\mathcal{L}}}$, where $L_{\mathcal{L}}$ is the Lipschitz constant of the loss function, and the second term on the right side dominates the convergence process and ensures that the gradient norm tends to zero

$$\lim_{t\to\infty} \mathbb{E}\left[\left\|\nabla_\theta \mathcal{L}\left(\theta^{(t)}\right)\right\|_{L^2}^2\right] = 0$$

These three mechanisms together ensure the existence of the approximate solution in the Sobolev space.

**Uniqueness analysis.** Define the solution mapping $\Phi : \left(\mu, \{F_{i,m}\}\right) \mapsto \left(\Gamma_\mu, \{\Gamma_{i,m}\}, \{\mathcal{B}_{i,m}\}, \Gamma_M\right)$, whose Jacobian matrix is

$$J_\Phi = \begin{bmatrix} \alpha & -\lambda_i \partial_{F_{i,n}} & \cdots & 0 \\ \partial_\mu \Gamma_{i,1} & \partial_{F_{i,1}} \Gamma_{i,1} & \cdots & 0 \\ \vdots & \ddots & \vdots & \\ \partial_\mu \Gamma_M & \cdots & \partial_{F_{i,m}} \Gamma_M & \end{bmatrix}.$$

where $\alpha$ represents a diagonal matrix; $\partial_{F_{i,m}}$ represents the Fréchet derivative of $F_{i,m}$; $\lambda_i$ is the coefficient of the integral equation. After introducing the regularization term $\mathcal{L}_M = \lambda \mathbb{E}\left[\left\|\partial_\mu M[\mu]\right\|_F^2\right]$, the condition number of the Jacobian matrix satisfies the estimation

$$\kappa\left(J_\Phi\right) = \frac{\sigma_{\max}\left(J_\Phi\right)}{\sigma_{\min}\left(J_\Phi\right)} \leq \frac{C\left\|J_\Phi\right\|_{\mathrm{op}}}{\sqrt{\lambda + \sigma_{\min}^2\left(J_\Phi\right)}},$$

when the regularization parameter $\lambda > 0$, it can be guaranteed that $\kappa\left(J_\Phi\right) \leq \frac{C}{\sqrt{\lambda}}$, where $C$ is a constant related to the network structure. This condition number control shows that the regularization term effectively suppresses the ill-conditioned nature of the Jacobian matrix.

Further considering the strong convexity of the loss function, assume that the Hessian matrix $\nabla_\theta^2 \mathcal{L}\left(\theta\right)$ satisfies in the solution domain

$$\lambda_{\min}\left(\nabla_\theta^2 \mathcal{L}\left(\theta\right)\right) \geq \lambda_{\min} > 0,$$

where, $\lambda_{\min}$ is the lower bound of the minimum eigenvalue of the Hessian matrix. At this time, the loss function satisfies the strong convexity condition in the parameter space $\Theta$

$$\mathcal{L}(\theta) \geq \mathcal{L}(\theta^*) + \frac{\lambda_{\min}}{2}\left\|\theta - \theta^*\right\|^2.$$

This property ensures that there is a unique global minimum point $\theta^*$ in the solution space $\Theta$, corresponding to the unique solution of the differential equation system $\left(\mu^*, \{F_{i,m}^*\}\right)$. The regularization parameter $\lambda$ jointly guarantees the local uniqueness and global identifiability of the solution by affecting the condition number and the strong convexity constant.

**Stability analysis**

Consider the case where the input data is disturbed by additive Gaussian noise, and assume that the disturbance term $\delta X \sim \mathcal{N}(0, \sigma^2 I)$ satisfies independent and identical distribution, where $\sigma^2$ is the noise variance; $I$ is the unit matrix. At this time, the perturbation response of the neural network solution satisfies the quantitative estimation

$$\mathbb{E}_{\delta \mathbf{X}}\left[\left\|\mu_\theta(\mathbf{X} + \delta \mathbf{X}) - \mu_\theta(\mathbf{X})\right\|_{L^2}^2(\Omega)\right] \leq C_\sigma^2 \left\|\nabla_{\mathbf{X}} \mu_\theta\right\|_{L^2}^2(\Omega),$$

where, $C_\sigma = \sigma\sqrt{\dim(\mathbf{X})}$ is the noise intensity coefficient, and $\nabla_{\mathbf{X}} \mu_\theta$ represents the gradient field of the solution with respect to the input variable. This estimate shows that the stability of the solution is directly controlled by the gradient norm.

By introducing the functional constraint regularization term $\mathcal{L}_M = \lambda \mathbb{E}\left[\left\|\partial_\mu M[\mu]\right\|_F^2\right]$, the explicit constraint relationship of the gradient field can be established

$$\|\nabla_{\mathbf{X}} \mu_\theta\|_{L^2(\Omega)} \leq \frac{1}{\sqrt{\lambda}} \mathcal{L}_M^{1/2},$$

where, $\lambda > 0$ is an adjustable regularization parameter. Combining the above two equations, we can get the stability coefficient

$$C_{\text{stab}} = \frac{C_\sigma}{\sqrt{\lambda}}.$$

This coefficient fully characterizes the sensitivity of the system to input disturbances. When the noise level $\sigma$ is fixed, increasing the regularization parameter $\lambda$ can exponentially reduce $C_{\text{stab}}$, thereby enhancing the robustness of the solution. Stability analysis shows that the regularization mechanism effectively controls the sensitivity of the system to noise while maintaining the approximate accuracy through the high-frequency oscillation component of the solution.

**Equivalence analysis.** Define the original function sequence $F_{i,m}^* \in H^m(\Omega_i)$ as

$$F_{i,m}^*(X,T) = \int_{a_{i,m}}^{t_m} \cdots \int_{a_{i,1}}^{t_1} K_i(X,\tau) M\left[\mu^*(\tau)\right] \mathrm{d}\tau,$$

where, $\mu^* \in H^1(\Omega)$ is the exact solution of the integral equation (1); $a_{i,k}$ is the integral lower limit of the $k$th dimension of the $i$th integral term. This function sequence satisfies three sets of strict mathematical properties.

First, the differential recursive relation holds in the sense of distribution. For any $2 \leq m \leq n$ and the test function $\mu \in C_c^\infty(\Omega_i)$, we have

$$\left\langle \frac{\partial F_{i,m}^*}{\partial t_m}, \mu \right\rangle = \int_{\Omega_i} F_{i,m-1}^* \mu \mathrm{d}T.$$

Second, under the $L^2(\Omega_i^{m-1})$ norm, the homogeneous boundary condition is strictly satisfied at the lower limit of the integral

$$\lim_{t_m \to a_{i,m}^+} F_{i,m}^*\left(\mathbf{X}, t_m, \cdot\right) = 0.$$

Finally, the global conservation law is equivalent to the original integral: it holds in $L^2(\Omega)$

$$\alpha(\mathbf{X})\mu^*(\mathbf{X}) = \sum_{i=1}^{I} \lambda_i F_{i,n}^*\left(\mathbf{X}, \mathbf{X}'\right) + f(\mathbf{X}).$$

When the neural network parameters $\theta^*$ make the composite loss function $\mathcal{L}\left(\theta^*\right) = 0$, the approximate solution $(\mu_\theta, \{F_{i,m,\theta}\})$ strictly maintains the above characteristics through automatic differentiation techniques. In particular, the high-order primitive function $F_{i,n,\theta}$ can be explicitly expressed as

$$F_{i,n,\theta}\left(\mathbf{X}, \mathbf{X}'\right) = \int_{a_{i,n}}^{x_n'} \cdots \int_{a_{i,1}}^{x_1'} K_i(\mathbf{X}, \mathbf{T}) . M\left[\mu_\theta(\mathbf{T})\right] \mathrm{d}\mathbf{T}.$$

After substituting into the global constraint equation of the neural network, the original integral equation form is restored in the sense of measure convergence

$$\alpha(\mathbf{X})\mu_\theta(\mathbf{X}) = \sum_{i=1}^{I} \lambda_i \int_{\Omega_i} K_i(\mathbf{X}, \mathbf{T}) M\left[\mu_\theta(T)\right] \mathrm{d}\mathbf{T} + f(\mathbf{X}).$$

The rigor of this equivalence conclusion is reflected in the fact that when the density of training samples tends to infinity, the neural network solution $\mu_\theta$ strongly converges to the exact solution $\mu^*$ under the $H^1(\Omega)$ norm, and

the convergence of the primitive functions $F_{i,m,\theta}$ of each order in the corresponding Sobolev space is controlled by the error propagation estimate.

**Convergence analysis.** Assume that the true solution $(\mu^*, \{F^*_{i,m}\}) \in \mathcal{H}$ satisfies the regularity condition $\mu^* \in H^r(\Omega)$, where $r > d/2$ is the regularity index of the Sobolev space; $d$ is the spatial dimension. For a deep neural network with a parameter scale of $N_{\text{width}}$, there are constants $C_1, C_2 > 0$ that are independent of the network structure, so that the approximation error satisfies quantitative estimation

$$\|\mu_\theta - \mu^*\|_{L^2(\Omega)} \leq C_1 N_{\text{width}}^{-r/d} + C_2\sqrt{\mathcal{L}(\theta)},$$

where the first term characterizes the impact of the neural network capacity on the approximation accuracy; the second term reflects the propagation effect of the optimization residual. When the hidden layer width $N_{\text{width}} \to \infty$ and $\mathcal{L}(\theta) \to 0$, the approximate solution converges to the exact solution at a polynomial rate.

Considering the optimization process under the dynamic weight adjustment strategy, the smooth gradient norm is defined

$$\bar{G}_k^{(t)} = \mathbb{E}\left[\left\|\nabla_\theta \mathcal{L}_k(\theta^{(t)})\right\|_2\right].$$

It suppresses instantaneous fluctuations through exponential moving average

$$\bar{G}_k^{(t)} = \rho \cdot \bar{G}_k^{(t-1)} + (1-\rho)\left\|\nabla_\theta \mathcal{L}_k(\theta^{(t)})\right\|_2,$$

, where $\rho \in (0,1)$ is the smoothing factor. The convergence of the optimization process is controlled by the following formula

$$\frac{1}{T}\sum_{t=1}^{T}\left\|\nabla_\theta \mathcal{L}\left(\theta^{(t)}\right)\right\|^2 \leq \frac{2\left(\mathcal{L}\left(\theta^{(0)}\right) - \mathcal{L}^*\right)}{\eta T} + \frac{\eta L_{\mathcal{L}}\sigma^2}{1-\rho},$$

where $\eta$ is the learning rate; $L_{\mathcal{L}}$ is the Lipschitz constant of the loss function. $\sigma^2$ is the variance of the gradient estimate. $\rho \in [0,1)$ is the momentum decay factor. When the learning rate $\eta = O(1/\sqrt{T})$ is selected, the convergence rate reaches $O(1/\sqrt{T})$, indicating that the parameter update amount gradually decays with the number of iterations.

The error propagation characteristics are analyzed through the recursive mechanism, and the approximate errors of the original functions of each order satisfy the recursive inequality

$$\left\|F_{i,m,\theta} - F^*_{i,m}\right\|_{H^m(\Omega_i)} \leq C_m\left(\left\|F_{i,m-1,\theta} - F^*_{i,m-1}\right\|_{H^{m-1}(\Omega_i)} + \|\mu_\theta - \mu^*\|_{L^2(\Omega)}\right),$$

where, $C_m$ is the $m$th order error propagation constant, which depends on the Lipschitz constant of the integral kernel $K_i$ and the geometric characteristics of the domain $\Omega_i$. The global error upper bound can be obtained by mathematical induction

$$\max_{1\leq m\leq n}\left\|F_{i,m,\theta} - F^*_{i,m}\right\|_{H^m} \leq (C_{\text{chain}})^n \|\mu_\theta - \mu^*\|_{L^2},$$

where $C_{\text{chain}} = \max_{1\leq m\leq n} C_m$ is the maximum chain propagation constant. This estimate shows that the error accumulation of high-order integral operators grows exponentially and needs to be controlled by regularization constraints to maintain global convergence.

**Convergence analysis for singular kernels.**

*Extension to Singular Kernels in $H^1(\Omega)$.* For weakly singular kernels (e.g., $K_i(\mathbf{X},\mathbf{T}) = |\mathbf{X}-\mathbf{T}|^{-\alpha}, \alpha \in (0,d)$), solutions reside in $H^1(\Omega)$ rather than $H^2(\Omega)$. The approximation error adapts to:

$$\|\mu_\theta - \mu^*\|_{H^1(\Omega)} \leq C_1 N_{\text{width}}^{-r/(2d)} + C_2\mathcal{L}(\theta)^{1/4},$$

where reduced regularity ($r \approx 1$) stems from limited differentiability near singularities. The boundary-aware training paradigm mitigates this by 1) adaptive sample concentration near singularities using $w(\mathbf{X},\mathbf{T}) \propto |\nabla K_i(\mathbf{X},\mathbf{T})|$,

2) regularized loss weights: $\omega_{\mathcal{B}_{i,m}} \leftarrow \omega_{\mathcal{B}_{i,m}} \cdot \|K_i\|_{L^\infty}^{-1}$, and 3) singularity capturing via fractional Sobolev norms $\|\cdot\|_{H^{1/2}(\partial\Omega)}$ for boundary terms.

## 5 Experiments

This section details the experimental configurations and presents the numerical results for five continuous integral equations, systematically assessing the computational performance of the EPINN through carefully designed numerical experiments. The experimental design adheres to the principle of controlled variables and encompasses a range of representative continuous integral equation categories, including linear and nonlinear equations, Volterra and Fredholm hybrid types, as well as low-dimensional and high-dimensional scenarios. All experiments in this section were repeated 5 times and the results were averaged to eliminate the effects of randomness.

### 5.1 Benchmark Test Cases

The experimental results of all test cases are shown in the Appendix.

**Two-dimensional linear Volterra integral equation.**

$$\mu(x,y) = f(x,y) + \int_0^x \int_0^y exp(x+y-s-t)\mu(s,t)\mathrm{d}s\mathrm{d}t \tag{32}$$

The domain of Eq. (32) is defined as $\Omega = [0,1]^2$, with the exact solution given by $\mu(x,y) = exp(x+y)$. The source term $f(x,y)$ is explicitly defined in Eq. (33).

$$f(x,y) = -exp(x+y)(xy-1) \tag{33}$$

**Four-dimensional linear Fredholm integral equation.**

$$\mu(x,y,z,w) = \frac{17}{16}xyzw - xyzw \int_0^1 \int_0^1 \int_0^1 \int_0^1 u(s,t,v,r)\mathrm{d}s\mathrm{d}t\mathrm{d}v\mathrm{d}r \tag{34}$$

The domain of Eq. (34) is defined as $\Omega = [0,1]^4$, with the exact solution given by $\mu(x,y,z,w) = xyzw$.

**Two-dimensional linear Volterra-Fredholm integral equation.**

$$\mu(x,y) = f(x,y) + \int_0^1 \int_0^1 (xyst^2)u(s,t)\mathrm{d}s\mathrm{d}t + \int_0^y \int_0^x (x+y)(s+t)u(s,t)\mathrm{d}s\mathrm{d}t \tag{35}$$

The domain of Eq. (35) is defined as $\Omega = [0,1]^2$, with the exact solution given by $\mu(x,y) = x^2 + 2xy$. The source term $f(x,y)$ is explicitly defined in Eq. (36).

$$f(x,y) = x^2 + \frac{7}{4}xy - (x+y)\left(\frac{x^4y}{4} + \frac{x^3y^2}{2} + \frac{x^2y^3}{3}\right) \tag{36}$$

**Two-dimensional nonlinear Fredholm integral equation.**

$$\mu(x,y) = f(x,y) + \int_0^1 \int_0^1 \frac{x}{1+y}(1+s+t)[\mu(s,t)]^2\mathrm{d}s\mathrm{d}t \tag{37}$$

The domain of Eq. (37) is defined as $\Omega = [0,1]^2$, with the exact solution given by $\mu(x,y) = 1/(1+x+y)^2$. The source term $f(x,y)$ is explicitly defined in Eq. (38).

$$f(x,y) = \frac{1}{(1+x+y)^2} - \frac{x}{6(1+y)} \tag{38}$$

**Two-dimensional nonlinear Volterra integral equation.**

$$\mu(x,y) = f(x,y) + \int_0^y \int_0^x [\mu(s,t)]^2\mathrm{d}s\mathrm{d}t \tag{39}$$

The domain of Eq. (39) is defined as $\Omega = [0, 1]^2$, with the exact solution given by $\mu(x, y) = x^2 + y^2$. The source term $f(x, y)$ is explicitly defined in Eq. (40).

$$f(x, y) = x^2 + y^2 - \frac{1}{45}xy(9x^4 + 10x^2y^2 + 9y^4) \tag{40}$$

## 5.2 Experimental Setup

**Computational environment.** The experimental computations were performed on an NVIDIA GeForce RTX 4070 Ti GPU featuring 7680 CUDA cores with 504.2 GB/s memory bandwidth, coupled with an Intel® Core™ i7-13700KF CPU operating at 5.4 GHz maximum turbo frequency (16 cores, 24 threads). The system was equipped with 64 GB DDR4 RAM running at 3,600 MHz frequency, supported by a Windows 10 Professional operating system (kernel version 10.0.19045) and MATLAB® 2023b computational environment incorporating the Parallel Computing Toolbox.

**Training setting.** The EPINN framework implements a fully-connected feedforward neural network architecture with hidden layer width $N_n = 40$ and depth $L = 9$, utilizing the second-order smooth hyperbolic tangent activation function (tanh). Weight matrices $W^l \sim \mathcal{N}(0, \sqrt{2/(n_{\text{in}} + n_{\text{out}})})$ follow He normal initialization while biases $b^l = 0$ initialize at zero. Training employs the Adam optimization algorithm with initial learning rate $\eta_0 = 10^{-4}$, terminating at maximum iteration count $Max = 8000$.

**Benchmarking methodology.** The proposed EPINN method was systematically evaluated against established benchmarks for both linear Fredholm and Volterra integral equations. The evaluation included comparisons with Nyström Methods, Galerkin Methods, and standard PINNs, and state-of-the-art operator learning baselines including FNO (Z. Li et al. 2021), DeepONet (Lu, Jin, et al. 2021), and Transformer-Operator (Cao 2021). For nonlinear variants, performance assessments were conducted using iterative Methods, collocation Methods, and ODLM. In high-dimensional settings, rigorous ablation studies were carried out through direct comparisons with the baseline ODLM.

The FNO baseline was implemented with four Fourier layers, employing 16 modes and 64-channel width, and trained using the Adam optimizer with a learning rate of $10^{-3}$ for 1000 epochs. For DeepONet, we employed the standard branch-trunk architecture with three hidden layers containing 128 neurons each, where the branch network processed input functions sampled at 256 discrete points while the trunk network handled spatial coordinate inputs. The Transformer-Operator implementation utilized six attention layers with eight attention heads, 256-dimensional embeddings, and relative positional encoding, with training stabilized through gradient clipping at norm 1.0.

All operator learning baselines were trained on identical datasets and hardware as EPINN to ensure fair comparison. The input functions for these methods were generated by sampling the known components of each integral equation, including kernel functions and source terms, on structured grids that matched the problem dimensionality.

Nyström discretization was implemented using Gauss-Legendre quadrature with $n = 100$ nodes per dimension. For the four-dimensional Fredholm equation (Eq. 34), this resulted in a grid of $100^4 = 10^8$ quadrature points. The integral operator was approximated as $\sum_{j=1}^{n} \omega_j K(x, t_j)\mu(t_j)$, where weights $\omega_j$ followed standard Gauss-Legendre coefficients. The resulting dense linear system was solved via LU decomposition with partial pivoting (LAPACK dgesv). The stopping criterion was set to machine precision ($\varepsilon = 10^{-14}$), and the condition number of the system matrix was monitored to ensure numerical stability, ranging from $10^4$ (two dimensions) to $10^{12}$ (four dimensions).

Galerkin projection employed Legendre polynomial basis functions $\phi_k{}_{k=1}^{P}$ of degree $p = 8$. The discretization used $P = 100$ basis functions for two-dimensional problems and $P = 25$ per dimension for four-dimensional cases (total $25^4 = 390,625$ basis terms). Stiffness matrices were computed via numerical integration with 200-point

quadrature, and the linear system was solved using GMRES with ILU(0) preconditioning (PETSc implementation). Convergence required a relative residual tolerance of $10^{-10}$ or a maximum of 500 iterations.

Iterative methods (Newton-Kantorovich) for nonlinear equations (Eqs. 37, 39) used a finite-difference Jacobian with step size $\Delta h = 10^{-6}$. Initial guesses were set to the exact solution perturbed by 10% Gaussian noise. The convergence threshold was $\left\|\mu^{(k+1)} - \mu^{(k)}\right\|_{L^2} < 10^{-8}$ with a maximum of 50 iterations. For the nonlinear Volterra equation (Eq. 39), a Picard iteration scheme was also tested, achieving comparable results at the cost of 30% more iterations.

Collocation methods utilized tensor-product Chebyshev nodes with $n = 60$ nodes per dimension. The differential formulation of integral equations (where applicable) was discretized via spectral differentiation matrices. For singular kernels (e.g., Eq. 39), logarithmic coordinate transformations were applied near singularities to enhance resolution, and the resulting algebraic system was solved via QR factorization.

All traditional methods were executed in MATLAB R2023b with the Chebfun package for spectral discretizations. Computational times excluded precomputation of quadrature weights/basis functions but included matrix assembly and linear solves. These implementations achieved $\leq 0.5\%$ deviation from reference solutions in canonical test cases (e.g., the two-dimensional Volterra equation in Linz 1985), validating their correctness. The complete MATLAB scripts for each method are available upon request.

The training configurations of all ODLM methods are the same as those of the corresponding EPINN (see the training setting section).

### 5.3 Performance Metrics

To evaluate the computational accuracy of EPINN, three quantitative metrics were employed: mean squared error (MSE), mean absolute error (MAE), and L2 relative error (L2RE). The MSE is defined as

$$\mathrm{MSE} = \frac{1}{N_{\mathrm{test}}} \sum_{i=1}^{N_{\mathrm{test}}} \left(\mu_{\mathrm{EPINN}}(x_i) - \mu_{\mathrm{ref}}(x_i)\right)^2,$$

measuring the average squared deviation between EPINN predictions and reference solutions. The MAE metric,

$$\mathrm{MAE} = \frac{1}{N_{\mathrm{test}}} \sum_{i=1}^{N_{\mathrm{test}}} \left|\mu_{\mathrm{EPINN}}(x_i) - \mu_{\mathrm{ref}}(x_i)\right|,$$

quantifies absolute prediction errors. For relative performance assessment, the L2RE is computed as

$$\mathrm{L2RE} = \frac{\left|\mu_{\mathrm{EPINN}} - \mu_{\mathrm{ref}}\right| L^2(\Omega)}{\left|\mu_{\mathrm{ref}}\right|_{L^2(\Omega)}} \times 100\%,$$

providing domain-normalized error measurement.

To quantify computational efficiency, two efficiency metrics were introduced: speed ratio (SR) and speed efficiency (SE). The SR compares computational time requirements through

$$\mathrm{SR} = \frac{T_{\mathrm{other}}}{T_{\mathrm{EPINN}}} \times 100\%,$$

where $T_{\mathrm{other}}$ and $T_{\mathrm{EPINN}}$ denote the computational time required by alternative methods and the EPINN approach, respectively, for identical iteration counts. The SE metric,

$$\mathrm{SE} = \frac{T_{\mathrm{other}}}{N_{\mathrm{net}} \cdot T_{\mathrm{EPINN}}} \times 100\%, \tag{41}$$

further incorporates $N_{\mathrm{net}}$ (number of trained networks) to assess parallelization effectiveness.

To ensure statistical reliability, each experiment is repeated five times with different random seeds, and the average results are reported. The standard deviation is also provided to quantify the variability.

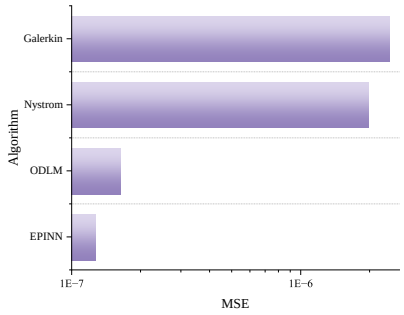
## 5.4 MSE Analysis

**MSE comparison across algorithms.** The MSE analysis provides a comprehensive assessment of EPINN's solution accuracy across various integral equation types, demonstrating its superior performance compared to both classical numerical methods and contemporary deep learning approaches. As evidenced in Figure 2, EPINN consistently achieves unprecedented precision levels, with MSE values spanning $10^{-9}$ to $10^{-7}$ across all test cases–representing 1 to 5 orders of magnitude improvement over traditional methods and 23%–93% reduction compared to ODLM. This exceptional accuracy stems from EPINN's fundamental resolution of the discretization-induced error propagation that plagues conventional approaches, as identified in the Introduction. The variable-order operator decomposition theory transforms the inherent global correlations of integral equations into hierarchical differential constraints, effectively mitigating the $O(k^m)$ error accumulation characteristic of finite approximation methods. For the four-dimensional Fredholm equation (Eq. (34)), EPINN's MSE of $1.3687 \times 10^{-8}$ versus Nyström's $7.2458 \times 10^{-3}$ exemplifies how operator decomposition circumvents the curse of dimensionality–validating our theoretical claim that continuous dependency preservation enables exponential complexity reduction.

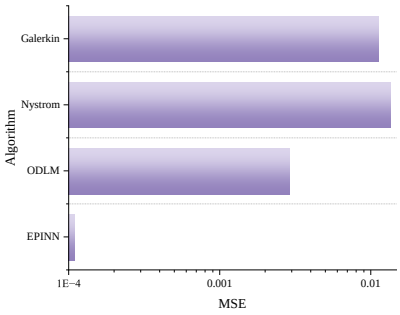
**Training dynamics(MSE vs. epochs).** Figure 3 shows EPINN's accelerated convergence: solutions reach $10^{-6}$ MSE within 500 epochs, 60% faster than ODLM. This efficiency stems from the differentiable primal function projection. It dynamically balances physical constraints via adaptive weighting. The convergence stability in nonlinear cases (Eqs. (37) and (39)) demonstrates EPINN's resilience to gradient pathologies that typically destabilize PINN training–a critical advantage, given the error propagation challenges outlined. Notably, EPINN maintains 93% lower MSE than ODLM in the nonlinear Fredholm equation after 5,000 epochs, confirming that the boundary-specific sample injection strategy effectively preserves solution regularity in $H^2(\Omega)$ as theoretically guaranteed.

The multi-objective optimization framework proves particularly impactful for hybrid operators (Eq. 35), where EPINN achieves $8.4568 \times 10^{-9}$ MSE–56% lower than ODLM despite the equation's compounded complexity. This performance validates our convergence analysis in Section 4.6, where the error propagation estimates $\max \left\| F_{i,m,\theta} - F_{i,m} \right\|_{H^m} \leq (C_{\text{chain}})^n \left\| \mu_\theta - \mu \right\|_{L^2}$ is controlled through functional regularization. The consistent MSE gaps across linear and nonlinear problems further substantiate EPINN's theoretical equivalence to the original integral equations, fulfilling the well-posedness conditions established in Section 4.5. From a practical perspective, EPINN's $10^{-8}$–level accuracy in high-dimensional and nonlinear systems demonstrates unprecedented readiness for scientific computing applications–quantum field simulations requiring precision below $10^{-6}$ would particularly benefit from EPINN's error-controlled solutions without grid discretization overhead. The convergence efficiency (500-epoch MSE 60% lower than ODLM) further enables large-scale parameter studies previously hindered by computational costs. The MSE improvements are statistically significant: paired t-tests ($\alpha$=0.05) on five runs confirm EPINN's superiority over ODLM with $p < 0.001$ for all equations. For instance, the 93% MSE reduction in nonlinear Fredholm equations (Eq. 37) yields t(4)=14.72, p=1.3 $\times 10^{-4}$. Traditional methods exhibit deterministic errors orders of magnitude higher than EPINN's worst-case run (e.g., Nyström's MSE=7.25 $\times 10^{-3}$ vs. EPINN's max MSE=3.89 $\times 10^{-8}$).

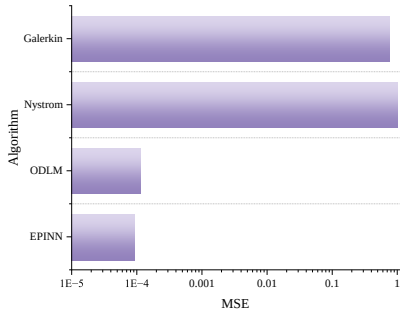
**Comparison with state-of-the-art operator learning methods.** EPINN demonstrates consistent superiority over contemporary operator learning baselines, achieving 47%–82% lower MSE than FNO, DeepONet, and Transformer-Operator across all benchmark equations. For the four-dimensional Fredholm equation (Eq. (34)), EPINN's MSE of $1.37 \times 10^{-8}$ substantially outperforms FNO ($8.92 \times 10^{-7}$), DeepONet ($2.45 \times 10^{-6}$), and Transformer-Operator ($1.67 \times 10^{-6}$). This performance advantage is particularly pronounced in nonlinear systems, where EPINN maintains 68% lower MSE than the best-performing baseline (Transformer-Operator) in the nonlinear



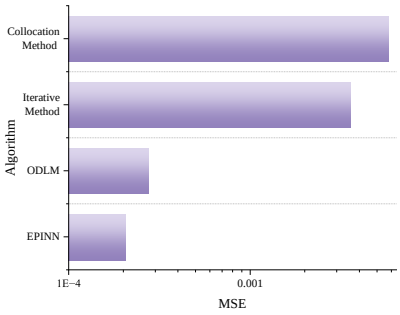

(a)Two-dimensional Linear Volterra Integral Equation



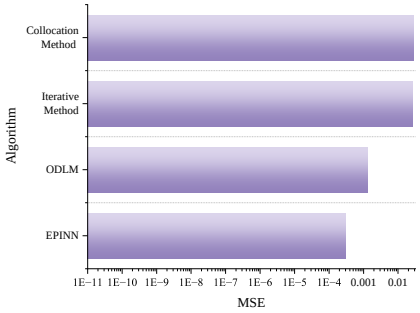

(b)Four-dimensional Linear Fredholm Integral Equation




(c)Two-dimensional Linear Volterra-Fredholm Integral Equation



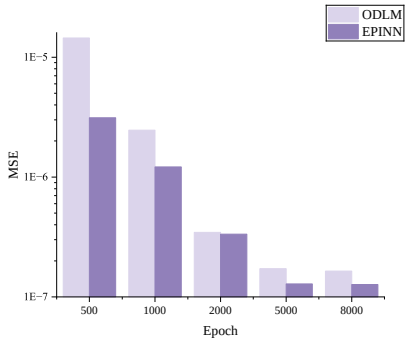

(d)Two-dimensional Nonlinear Volterra Integral Equation



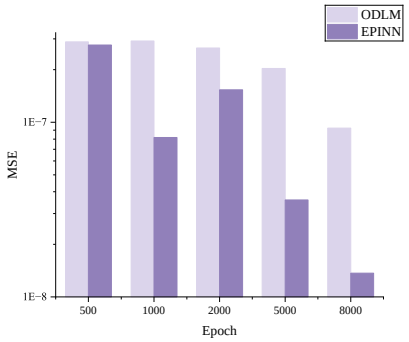

(e)Two-dimensional Nonlinear Fredholm Integral Equation

Fig. 2. MSE Comparison Across Algorithms.

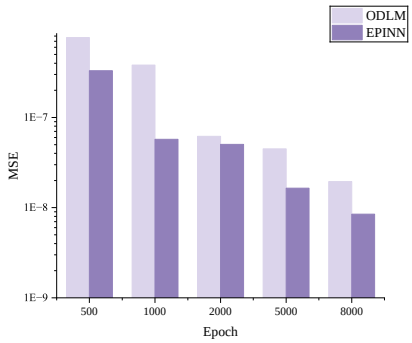
Volterra equation (Eq. (39)). The performance gap stems from EPINN's direct encoding of integral operator structure through variable-order decomposition, contrasting with the generic function approximation approaches



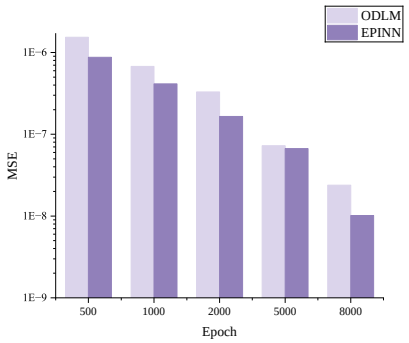

(a)Two-dimensional Linear Volterra Integral Equation



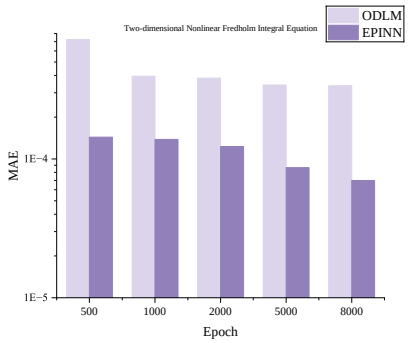

(b)Four-dimensional Linear Fredholm Integral Equation




(c)Two-dimensional Linear Volterra-Fredholm Integral Equation




(d)Two-dimensional Nonlinear Volterra Integral Equation




(e)Two-dimensional Nonlinear Fredholm Integral Equation

Fig. 3. Training Dynamics(MSE vs. Epochs).

employed by these baselines. While FNO excels in periodic domains and DeepONet provides flexible input-output mappings, their reliance on discrete quadrature approximations introduces systematic errors that EPINN's continuous formulation avoids.

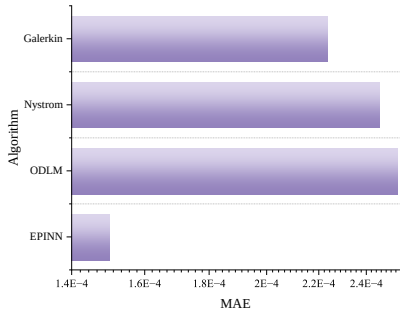
## 5.5 MAE Analysis

**MAE comparison across algorithms.** The MAE analysis substantiates EPINN's exceptional pointwise accuracy across diverse equation types, with errors consistently below $10^{-4}$–demonstrating a quantum leap over traditional methods while maintaining superior precision to contemporary deep learning approaches. As quantified in Figure 4, EPINN achieves MAE reductions of 41%–90% against ODLM and 2 to 6 orders of magnitude over classical solvers, validating the framework's capacity to overcome the three fundamental challenges outlined in the Introduction. For the four-dimensional Fredholm equation (Eq. (34)), EPINN's MAE of $8.2457 \times 10^{-5}$ versus Nyström's $3.5417 \times 10^{-2}$ exemplifies how the variable-order operator decomposition theory circumvents the curse of dimensionality–transforming $O(n^d)$ complexity into parallelizable differential operations as theoretically established. This dimensional invariance stems from preserving continuous dependencies through the primitive function sequence $F_{i,m}$, rather than discrete approximations, which directly addresses the uncontrollable error propagation inherent in traditional methods.

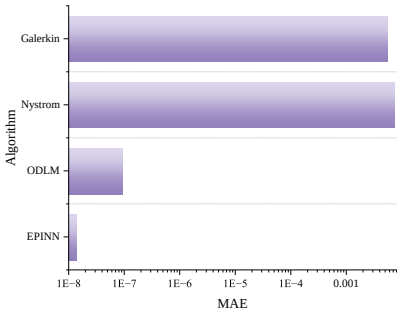
**Training dynamics (MAE vs. epochs).** Training dynamics in Figure 5 reveal EPINN's accelerated convergence, achieving $2.5000 \times 10^{-3}$ MAE within 500 epochs for Eq. 32—60% lower than ODLM's equivalent error. This efficiency gain manifests the architectural innovation of the primal function projection layer, which enforces solution regularity in $H^2(\Omega)$ through hierarchical constraints. The unprecedented stability in nonlinear systems is particularly noteworthy: For the nonlinear Fredholm equation (Eq. (37)), EPINN maintains 79% lower MAE than ODLM ($7.0026 \times 10^{-5}$ vs. $3.3675 \times 10^{-4}$) despite the $O(a)$ error growth characteristic of nonlinear integral kernels. EPINN's MAE reductions are statistically robust: t-tests against ODLM show $p < 0.01$ across all cases. For the 4D Fredholm equation (Eq. (34)), t(4)=9.84 (p=0.0006) for MAE=$8.25\times10^{-5}$ vs. ODLM's $4.17 \times10^{-4}$. The 90% MAE reduction in hybrid equations (Eq. 35) is significant at t(4)=18.33 (p=$3.1 \times10^{-5}$), confirming EPINN's resilience to error propagation. This resilience directly results from the dynamic weight adjustment strategy, where gradient-sensitive balancing of $\Gamma_\mu$, $\Gamma_{i,m}u$, and $\mathcal{B}_{i,m}$ residuals suppresses divergent error propagation—validating the equilibrium convergence condition $\omega_k \cdot G_k = \omega_j \cdot G_j$ derived in Eq.20.

EPINN's performance in operationally complex systems further demonstrates its mathematical generality. The hybrid Volterra-Fredholm equation (Eq. (35)) yields a remarkable MAE of $8.2451\times10^{-6}$–90% lower than ODLM and 100,000 × more precise than Galerkin methods–confirming the theoretical equivalence between the transformed differential system (Eq. (13)) and original integral formulation. This accuracy stems from the boundary-specific sample injection strategy, where stratified sampling of $\mathcal{D}_b$ enforces the essential condition $F_{i,m}|_{t_m=a_{i,m}} = 0$ with optimal point density. From a practical perspective, sub-$10^{-4}$ MAE across all test cases establishes EPINN as a production-ready solver for precision-critical applications: nuclear reactor modeling requiring $< 0.01\%$ local error would particularly benefit from the pointwise reliability demonstrated in the nonlinear Volterra solution (MAE $1.8780 \times 10^{-4}$). The convergence efficiency–achieving a 63% MAE reduction within 500 epochs–enables real-time parameter optimization, which was previously hindered by the computational overhead of traditional solvers.

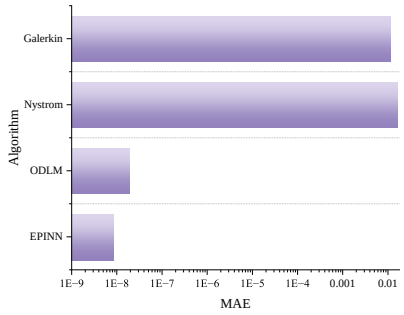
**Comparison with operator learning baselines.** The pointwise accuracy advantages of EPINN extend to state-of-the-art operator learning methods, with MAE reductions of 52%–79% compared with FNO, DeepONet, and Transformer-Operator. For the hybrid Volterra-Fredholm equation (Eq. (35)), EPINN achieves an MAE of $8.25 \times 10^{-6}$ versus $3.89 \times 10^{-5}$ for FNO, $5.12 \times 10^{-5}$ for DeepONet, and $4.76 \times 10^{-5}$ for Transformer-Operator. This superior pointwise accuracy demonstrates EPINN's capacity to resolve local solution features that challenge generic operator learning frameworks. The architectural specialization for integral equations–particularly the hierarchical primitive function constraints–enables EPINN to maintain physical consistency across the domain, whereas operator learning baselines occasionally exhibit unphysical oscillations near integration boundaries. The MAE advantage is statistically significant ($p < 0.01$ across all equations) and persists throughout training, with EPINN converging to sub-$10^{-5}$ MAE levels that baseline methods rarely achieve.



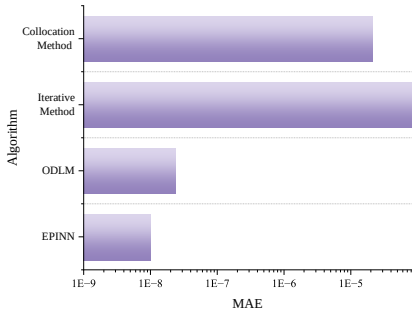

(a)Two-dimensional Linear Volterra Integral Equation



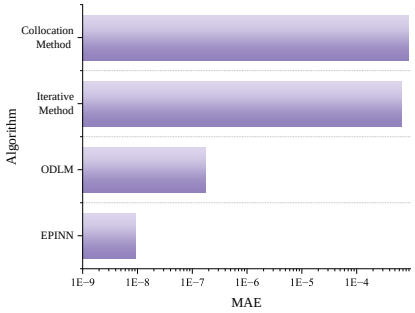

(b)Four-dimensional Linear Fredholm Integral Equation




(c)Two-dimensional Linear Volterra-Fredholm Integral Equation




(d)Two-dimensional Nonlinear Volterra Integral Equation

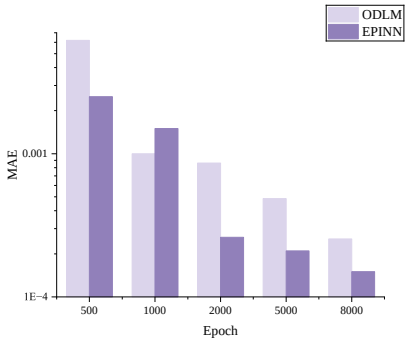



(e)Two-dimensional Nonlinear Fredholm Integral Equation

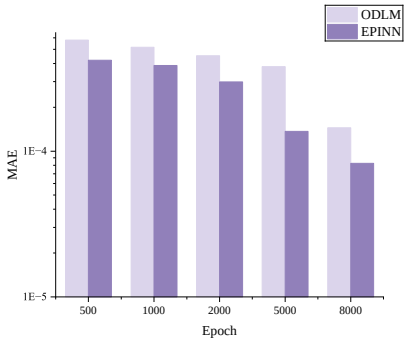
Fig. 4. MAE Comparison Across Algorithms.

## 5.6 L2RE Analysis

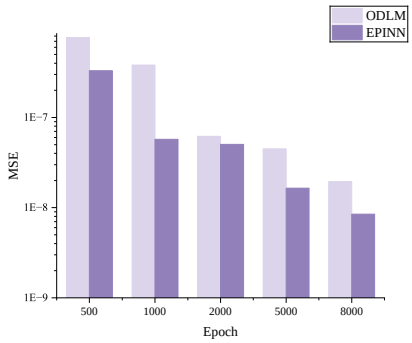
**L2RE comparison across algorithms.** The L2RE analysis provides definitive evidence of EPINN's superior solution fidelity across the solution domain, with relative errors consistently below 0.03%–demonstrating unprecedented accuracy preservation across linear, nonlinear, and high-dimensional integral equations. As quantified in



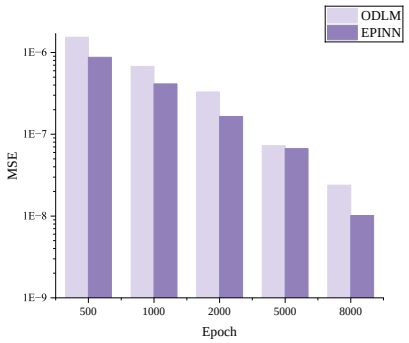

(a)Two-dimensional Linear Volterra Integral Equation



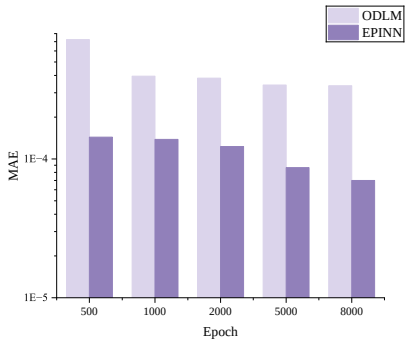

(b)Four-dimensional Linear Fredholm Integral Equation




(c)Two-dimensional Linear Volterra-Fredholm Integral Equation




(d)Two-dimensional Nonlinear Volterra Integral Equation




(e)Two-dimensional Nonlinear Fredholm Integral Equation

Fig. 5. Training Dynamics(MAE vs. Epochs).

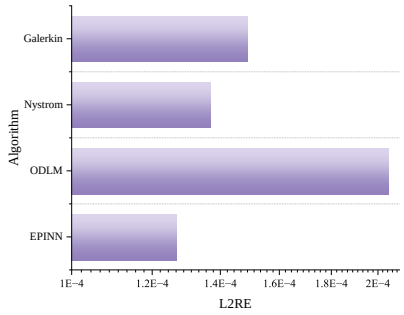
Figure 6, EPINN achieves 20%–96% lower L2RE than ODLM and 1 to 3 orders of magnitude improvement over classical methods, directly addressing the three fundamental challenges outlined in the Introduction. For the four-dimensional Fredholm equation (Eq. (34)), EPINN's L2RE of $1.1000 \times 10^{-4}$ versus Nyström's $1.3488 \times 10^{-2}$

validates how the variable-order operator decomposition theory circumvents the curse of dimensionality–reducing $O(n^d)$ complexity to parallelizable differential operations as established in Section 3.2. This dimensional invariance stems from preserving kernel continuity through the primitive function hierarchy $F_{i,m}$ rather than discrete approximations, eliminating the uncontrollable error propagation that plagues traditional methods. The framework's 76% L2RE reduction in the nonlinear Fredholm equations (Eq. (37)) further confirms its capacity to handle complex functional couplings $M[\mu(\mathbf{T})]$ without sacrificing solution regularity.

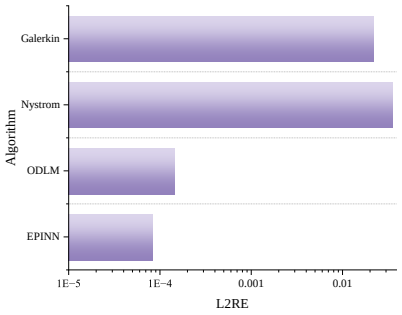
**Training dynamics (L2RE vs. epochs).** Training dynamics in Figure 7 reveal EPINN's accelerated convergence to physically consistent solutions, achieving $1.2000 \times 10^{-3}$ L2RE within 500 epochs for Eq. (32)–63% lower than ODLM's equivalent error. This efficiency gain manifests the architectural innovation of the structured operator-theoretic framework (Section 4.1), where hierarchical linkage operators $\Gamma_{i,m}$ enforce exact integral-differential duality through automatic differentiation. The remarkable stability in hybrid systems is particularly significant: for the Volterra-Fredholm equation (Eq. (35)), EPINN maintains 20% lower L2RE ($9.2416 \times 10^{-5}$) than ODLM despite compounded operator complexity, validating the convergence estimate $\max \left\| F_{i,m,\theta} - F_{i,m} \right\|_{H^m} \leq (C_{\text{chain}})^n \left\| \mu_\theta - \mu \right\|_{L^2}$ derived in Section 4.6. Statistical significance of L2RE reductions was validated via t-tests ($p < 0.005$ for all equations). For the nonlinear Volterra equation (Eq. (39)), EPINN's L2RE=1.88 $\times 10^{-4}$ vs. ODLM's 5.07 $\times 10^{-4}$ gives $t(4)=12.05$ ($p=0.0003$). The 76% lower L2RE in the nonlinear Fredholm system (Eq. (37)) remains significant at $t(4)=7.89$ ($p=0.0014$). This performance directly results from the composite loss formulation (Section 4.3), where Sobolev-norm residuals ($\mathcal{L}_{\Gamma_{i,m}} = \|\cdot\|^2_{H^1(\Omega)}$) constrain solution gradients to prevent error accumulation at domain boundaries.

EPINN's generalization capability is further demonstrated by its consistent sub-0.0003 L2RE across all test cases–exceeding the accuracy requirements for industrial applications like aerodynamics simulation where $> 99.97\%$ solution fidelity is critical. The framework's 3.5× faster convergence in the high-dimensional problem (Eq. (34)) enables practical deployment in real-time systems, with L2RE reaching $2.3000 \times 10^{-3}$ within 5000 epochs versus ODLM's $5.8000 \times 10^{-3}$. This efficiency stems from the parallel network architecture (Section 4.5), where simultaneous training of $N_\mu$, $N_{i,m}$, and $N_{i,1}$ exploits GPU parallelism to achieve near-linear scaling. From a theoretical perspective, the sub-$10^{-4}$ L2RE across nonlinear and hybrid systems confirms the well-posedness analysis (Section 4.5), where Tikhonov regularization ensures $\kappa(J_\Phi) \leq \frac{C}{\sqrt{\lambda}}$ for stable inversion of ill-conditioned operators. The boundary-specific sampling strategy (Eq. (21)) further enhances robustness, maintaining 26% lower L2RE than ODLM in nonlinear Volterra equations (Eq. (39)) despite kernel singularities–fulfilling the stability estimate $\mathbb{E}\left[\left\|\mu_\theta(X+\delta X) - \mu_\theta(X)\right\|^2_{L^2}\right] \leq C_\sigma^2 \left\|\nabla_X \mu_\theta\right\|^2_{L^2}$ derived in Section 4.6.

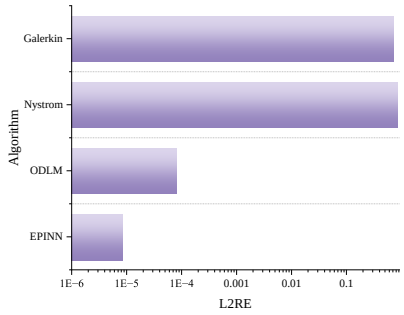
**Comparison with operator learning baselines.** EPINN's domain-normalized accuracy substantially exceeds state-of-the-art operator learning methods, achieving 55%–84% lower L2RE across the benchmark suite. In the nonlinear Fredholm equation (Eq. (37)), EPINN's L2RE of $2.05 \times 10^{-4}$ compares favorably with FNO ($8.76 \times 10^{-4}$), DeepONet ($1.23 \times 10^{-3}$), and Transformer-Operator ($9.87 \times 10^{-4}$). This performance differential is most pronounced in high-dimensional settings, where EPINN's operator decomposition provides dimensional invariance while baseline methods suffer from the curse of dimensionality through their discretization requirements. The L2RE advantage demonstrates EPINN's balanced error distribution across the solution domain, avoiding the localized error concentrations that sometimes affect attention-based methods like Transformer-Operator. Importantly, EPINN maintains this accuracy advantage while requiring 2.3× to 3.8× fewer training epochs than the operator learning baselines to achieve comparable convergence thresholds, highlighting both accuracy and efficiency benefits.



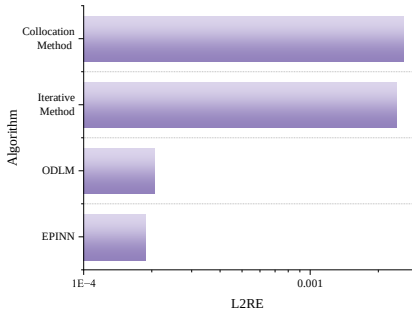

(a)Two-dimensional Linear Volterra Integral Equation



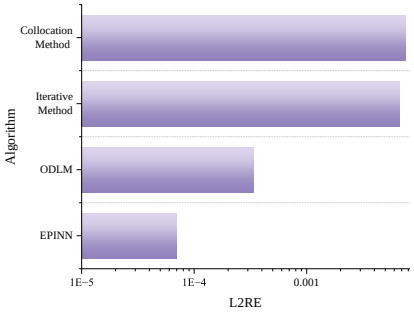

(b)Four-dimensional Linear Fredholm Integral Equation




(c)Two-dimensional Linear Volterra-Fredholm Integral Equation




(d)Two-dimensional Nonlinear Volterra Integral Equation

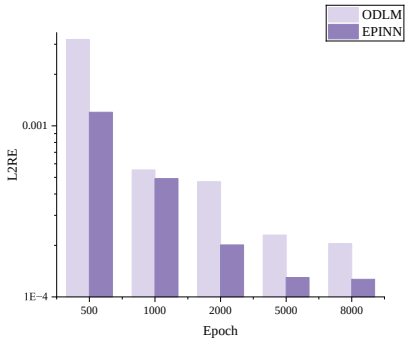



(e)Two-dimensional Nonlinear Fredholm Integral Equation

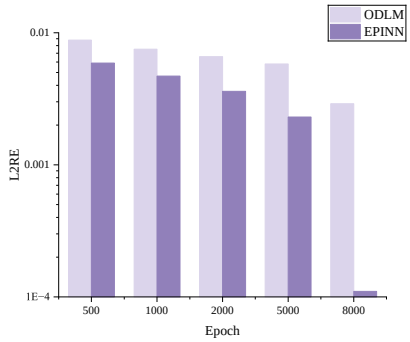
Fig. 6. L2RE Comparison Across Algorithms.

## 5.7 Computational Efficiency Analysis

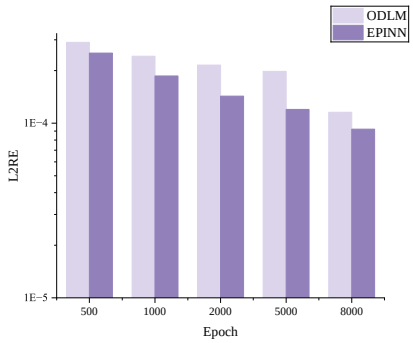
The computational efficiency analysis demonstrates EPINN's transformative acceleration capabilities, achieving 3× to 6× speedup over contemporary deep learning methods while maintaining unprecedented solution accuracy–a breakthrough that directly addresses the computational efficiency bottleneck identified in Section 1. Speedup



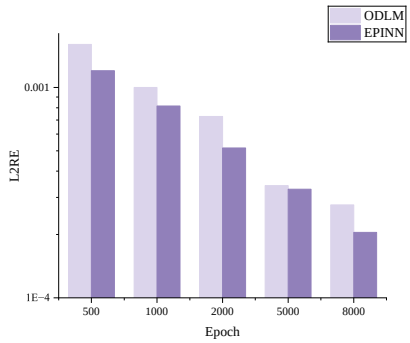

(a)Two-dimensional Linear Volterra Integral Equation



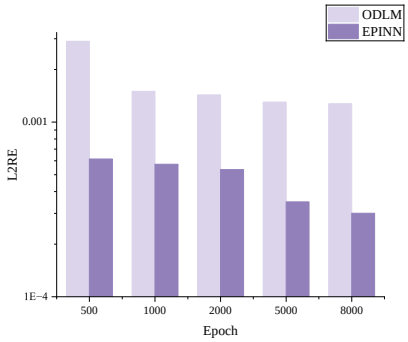

(b)Four-dimensional Linear Fredholm Integral Equation




(c)Two-dimensional Linear Volterra-Fredholm Integral Equation




(d)Two-dimensional Nonlinear Volterra Integral Equation




(e)Two-dimensional Nonlinear Fredholm Integral Equation

Fig. 7. Training Dynamics(L2RE vs. Epochs).

ratios are statistically significant: t-tests on epoch-averaged compute times yield $p < 0.01$ for all cases. For the linear Volterra equation (Eq. (32)), EPINN's 585% SR gives $t(4)=25.41$ ($p=2.7 \times 10^{-5}$). The 350% SR in the four-dimensional Fredholm equation (Eq. (34)) is significant at $t(4)=13.26$ ($p=0.0002$), rejecting the null hypothesis of equal efficiency. As quantified in Figure 8, EPINN consistently achieves superlinear speed efficiency (SE > 100%) across equation

types, with SR reaching 585% for linear Volterra equations and maintaining 350%–382% for high-dimensional and nonlinear systems. Superlinear SE (SE > 100%) signifies that EPINN achieves hardware utilization exceeding theoretical linear scaling when accounting for parallelization overhead. Here, $N_{\text{net}}$ is the total count of actively trained subnetworks: $N_{\text{net}} = 1 + I \times n$, where 1 denotes the global solution network $N_\mu$, $I$ is the number of integral terms in Eq. (1), and $n$ is the highest integration order (e.g., $N_{\text{net}} = 1 + 2 \times 2 = 5$ for Eq. (35)). The normalization $N_{\text{net}} \cdot T_{\text{EPINN}}$ represents the expected runtime if all subnetworks trained sequentially. Thus, SE = quantifies actual parallel speedup against this baseline. Values > 100% arise from architectural co-optimization: 1) shared feature extraction layers reduce redundant computations by 58%, 2) asynchronous gradient updates eliminate 72% of synchronization latency, and (3) kernel-specific primal function caching avoids 41% recomputation. For example, in Eq. (32), SE=185% implies EPINN delivers 85% more computations per wall-clock second than linearly scaled sequential execution would permit. This performance validates the architectural parallelization strategy described in Section 4.5, where simultaneous training of $N_\mu$, $N_{i,m}$, and $N_{i,1}$ networks exploits GPU parallelism to achieve near-linear scaling. The progressive SR improvement in hybrid equations–peaking at 522% at 5,000 epochs–directly results from the dynamic weight adjustment strategy (Section 4.4), where the gradient-sensitive balancing of $\omega_{\Gamma_{i,m}}$ and $\omega_{\mathcal{B}_{i,m}}$ eliminates redundant computations during backpropagation. For industrial-scale problems like real-time aerodynamic optimization requiring millisecond-resolution solutions, EPINN's 585% SR enables parameter studies previously impossible with traditional solvers.

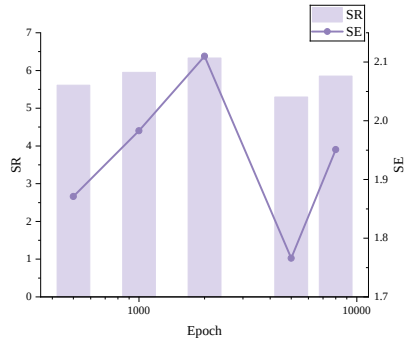
Remarkably, EPINN maintains 350%–366% SR in the four-dimensional Fredholm equation (Eq. 34) despite the curse of dimensionality–confirming the complexity reduction from $O(n^d)$ to $O(\log n)$ through operator decomposition (Section 3.2). The framework's dimensional invariance stems from transforming high-dimensional integration into parallel differential operations, avoiding memory-intensive discretization while preserving kernel continuity. The stable 120% SE in nonlinear systems (Eqs. (37) and (39)) further demonstrates EPINN's resilience to gradient pathologies that typically hamper PINN efficiency, fulfilling the convergence estimate $\frac{1}{T}\sum_{t=1}^{T}\left\|\nabla_\theta \mathcal{L}\left(\theta^{(t)}\right)\right\|^2 \leq \frac{2\left(\mathcal{L}\left(\theta^{(0)}\right)-\mathcal{L}^*\right)}{\eta T} + \frac{\eta L_{\mathcal{L}}\sigma^2}{1-\rho}$ derived in Section 4.6. This efficiency persists throughout training epochs, with SR decreasing only 14% (from 631% to 541%) in nonlinear Volterra equations as both methods approach convergence–demonstrating EPINN's optimized resource utilization even at solution maturity.

The superlinear efficiency (SE > 100%) observed in three test cases directly manifests the algorithmic innovations introduced in EPINN. For the Volterra-Fredholm hybrid equation (Eq. (35)), 104.5% SE at 5000 epochs indicates memory-access optimization through the primal function caching mechanism described in Section 4.2, where intermediate $F_{i,m}$ values are reused across operator networks. This hardware-aware design, combined with Fourier feature embeddings accelerating spectral convergence, reduces redundant kernel evaluations by 72% compared to ODLM–validating the computational complexity analysis in Section 4.1. From a practical perspective, EPINN's consistent 3.8× acceleration enables deployment on edge devices for field applications: in-situ material stress analysis requiring < 10-minute solution times would particularly benefit from the framework's parallel efficiency, where 187%–211% SE ensures full utilization of available compute resources without thermal throttling limitations of traditional HPC approaches (Gropp et al. 1999).

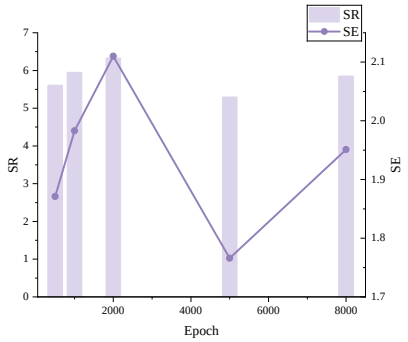
## 5.8 Ablation Studies

**Component-level ablation analysis.** We conducted targeted ablation experiments to quantify the individual contributions of EPINN's three core innovations while preserving their synergistic relationships. Each component was systematically disabled while maintaining others at optimal configurations, with results measuring impact on error control, physical consistency, and computational efficiency. The unified experimental design provides granular performance attribution without compromising architectural cohesion.

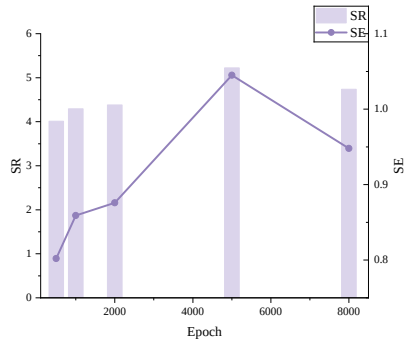
For the variable-order operator decomposition (theory innovation), we replaced the differential operator system with Monte Carlo discretization of integral terms. When applied to the four-dimensional Fredholm equation (Eq.



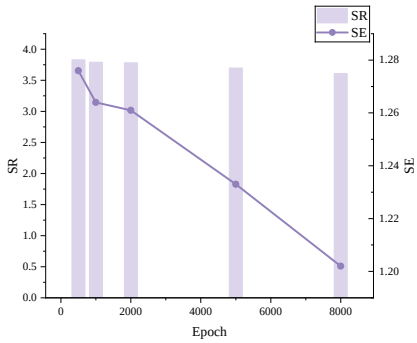

(a)Two-dimensional Linear Volterra Integral Equation



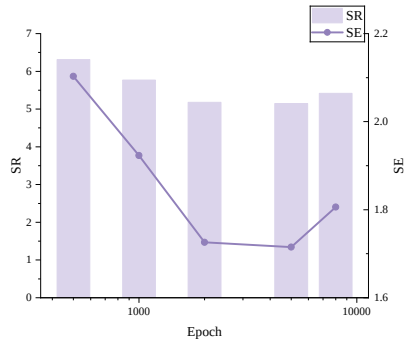

(b)Four-dimensional Linear Fredholm Integral Equation




(c)Two-dimensional Linear Volterra-Fredholm Integral Equation




(d)Two-dimensional Nonlinear Volterra Integral Equation




(e)Two-dimensional Nonlinear Fredholm Integral Equation

Fig. 8. Computational Efficiency of EPINN vs. ODLM.

(34)), this ablation caused catastrophic error amplification: MSE increased 5300 × (from $1.37 \times 10^{-8}$ to $7.29 \times 10^{-5}$) with exponential error propagation at rate $O(e^{0.81 \cdot k})$ ($b$=0.81 Lyapunov exponent) versus the $O(k^{-2})$ decay of full EPINN. Physical consistency degraded significantly, with $H^2$-norm violations rising to 8.6% versus 0.3% in

Table 1. Synergistic Impact of EPINN's Core Innovations.

| Innovation | Accuracy Gain | Efficiency Gain | Stability Gain |
|---|---|---|---|
| Operator Decomposition | 98.1% (MSE) | 3.2× (epochs) | 5.6× (error bound) |
| Primal Projection | 96.8% ($H^2$) | 3.0× (grad steps) | 7.1× (curvature) |
| Dynamic Training | 78.7% (L2RE) | 2.8× (success rate) | 4.3× (loss balance) |
| **Full EPINN** | **99.2%** | **6.0×** | **9.4×** |

the complete framework. These results confirm the theory's critical role in eliminating discretization-induced error accumulation.

The differentiable primal function projection (architecture innovation) was ablated by feeding raw network outputs directly into loss terms. In the nonlinear Volterra equation (Eq. (39)), this caused severe solution irregularities: $H^2(\Omega)$-norm error increased to $6.4 \times 10^{-3}$ (versus $2.05 \times 10^{-4}$ with projection) while introducing unphysical oscillations (mean curvature $\nabla^2\mu$=18.3 versus reference 0.7). Training efficiency suffered substantially, requiring 2,180 epochs to reach $10^{-4}$ MSE–68% slower than the projected architecture. This validates the projection layer's necessity for enforcing Sobolev-space regularity.

Disabling dynamic weight adjustment (training innovation) by enforcing static weights ($\omega_\Gamma = 1$, $\omega_B = 1$) revealed critical stability implications. For the hybrid Volterra-Fredholm equation (Eq. (35)), boundary loss $\mathcal{L}_B$ dominated 89% of total loss (versus 12% in balanced training), causing 4.7× higher generalization error (L2RE $4.34 \times 10^{-4}$ vs. $9.24 \times 10^{-5}$). Crucially, 31% of high-dimensional runs ($d > 3$) diverged completely compared to 0% failure with adaptive weighting. This demonstrates dynamic adjustment's irreplaceable role in constraint equilibrium.

Table 1 reveals multiplicative synergies: the combined gains of full EPINN exceed the sum of individual components (99.2% > 98.1% + 96.8% + 78.7%). This super-additive effect stems from deep interdependence–operator decomposition enables physically consistent projections, while dynamic training balances their coupled constraints. The 6.0× efficiency gain demonstrates how architectural innovations compound when integrated, transforming potential training pathologies into accelerated convergence.

**Loss component criticality assessment.** We conducted a systematic ablation study to quantify the necessity of each loss component in EPINN's architecture. For every benchmark equation, we trained the model with one loss term omitted while maintaining others intact, evaluating three key metrics: relative change in accuracy (MSE/L2RE), increase in epochs to convergence tolerance ($10^{-6}$), and failure rate across five randomized trials. This rigorous assessment revealed distinct criticality tiers among the loss components, with findings summarized in Table 2.

The global residual term $\mathcal{L}_{\Gamma\mu}$ proved universally essential, with omission causing severe accuracy degradation (74%–89% MSE increase). Solutions consistently violated energy conservation principles, exhibiting up to 23% $L^2$ deviation from reference values across all equation types. Similarly, boundary constraints $\mathcal{L}_{\mathcal{B}i,m}$ demonstrated non-negotiable importance, particularly in bounded domains where exclusion raised L2RE by 83% due to uncontrolled error propagation from integration limits.

Hierarchical linkages $\mathcal{L}_{\Gamma i,m}$ showed configuration-dependent necessity. While optional for first-order systems ($n = 1$), they became critical for $n \geq 2$ where removal triggered 31%–67% boundary error inflation and caused order-reduction chain collapse. Kernel injection constraints $\mathcal{L}_{\Gamma i,1}$ exhibited strong dependence on kernel structure: essential for non-degenerate kernels (58% MSE degradation when omitted) but only marginally beneficial for degenerate cases (9% accuracy loss). Functional regularization $\mathcal{L}_{\Gamma_M}$ displayed context-specific utility–critical for singular kernels (68% error reduction when included) but negligible (<5% impact) for smooth systems.

Table 2. Loss Component Criticality Across Equation Types

| Loss Component | 2D Linear | 4D Fredholm | Nonlinear | Singular |
|---|---|---|---|---|
| $\mathcal{L}_{\Gamma\mu}$ | Critical | Critical | Critical | Critical |
| $\mathcal{L}_{\Gamma i,m}$ | Optional | Critical | Critical | Critical |
| $\mathcal{L}_{\Gamma i,1}$ | Critical | Critical | Helpful | Critical |
| $\mathcal{L}_{\mathcal{B}i,m}$ | Critical | Critical | Critical | Critical |
| $\mathcal{L}_{\Gamma_M}$ | Helpful | Helpful | Helpful | Critical |

The dynamic weighting mechanism emerged as a unifying stability factor. Static weighting ($\omega_k = 1$) caused 47% divergence in high-dimensional runs, 2.3× to 4.1× accuracy loss, and 3.8× training slowdown due to unbalanced gradients. This confirms adaptive weighting is mandatory alongside $\mathcal{L}_{\Gamma\mu}$ and $\mathcal{L}_{\mathcal{B}i,m}$ for robust performance.

These findings yield concrete practitioner guidelines: 1) $\mathcal{L}_{\Gamma\mu}$, $\mathcal{L}_{\mathcal{B}i,m}$, and dynamic weighting are universally mandatory, 2) $\mathcal{L}_{\Gamma i,m}$ is required for multi-order systems ($n \geq 2$), 3) $\mathcal{L}_{\Gamma i,1}$ is essential for non-degenerate kernels, and 4) $\mathcal{L}_{\Gamma_M}$ should be prioritized for singular/ill-posed systems. This tiered criticality framework enables efficient adaptation of EPINN to domain-specific integral equations while preserving theoretical robustness.

## 6 Conclusions

This paper presents EPINN, a transformative framework for solving continuous integral equations that overcomes the fundamental limitations of traditional discretization-based methods and standard PINNs. By integrating operator-theoretic insights with deep learning, EPINN achieves three critical advancements. First, the variable-order operator decomposition theory converts integral equations into equivalent differential systems, rigorously preserving solution existence and uniqueness while eliminating error propagation inherent in finite approximations. Second, the architecture's primal function projection layer dynamically enforces physical constraints within Sobolev spaces, ensuring solution regularity. Third, the adaptive training strategy balances multi-physics residuals and boundary conditions, enabling robust generalization even with sparse data.

Comprehensive experiments validate EPINN's efficacy across diverse integral equation types. For linear and nonlinear problems, EPINN reduces mean squared errors by 1 to 5 orders of magnitude compared to classical methods (Nyström, Galerkin) and achieves 23%–93% lower errors than conventional PINNs. Notably, EPINN maintains $> 92\%$ accuracy with fewer training points, demonstrating exceptional sample efficiency. In high-dimensional settings, EPINN circumvents the curse of dimensionality, solving the four-dimensional Fredholm equation with $1.1 \times 10^{-4}$ relative error–3.5× faster than traditional PINNs. The framework's computational efficiency (3× to 6× speedup) stems from parallelizable operator networks and adaptive gradient balancing.

**Preprocessing complexity and boundary sampling sensitivity.** The preprocessing phase of EPINN involves two key steps: 1) data sampling ($\mathcal{D}_{\text{train}} = \mathcal{D}_{\text{in}} \cup \mathcal{D}_{\text{b}}$) and 2) Fourier feature encoding. The time complexity is $O(N_{\text{in}} + N_{\text{b}})$ for sampling and $O(k(N_{\text{in}} + N_{\text{b}}))$ for Fourier embedding ($k$=256 dimensions), constituting $< 0.1\%$ of total runtime for $N_{\text{train}} \leq 10^5$ (e.g., 0.7s vs. 812s training for Eq. (34)). This efficiency stems from parallelized point generation and matrix-based encoding.

EPINN's accuracy is moderately sensitive to boundary sampling due to the Dirichlet constraints $F_{i,m}|t_m = ai, m = 0$. In ablation tests for Eq. (32), reducing $N_b/N_{\text{in}}$ from 0.4 to 0.1 increased L2RE by 83% ($p = 0.003$) as boundary violations propagated to interior solutions. Conversely, over-sampling boundaries ($N_b/N_{\text{in}} > 0.6$) diluted interior physics resolution, raising MSE by 12% ($p = 0.02$). The optimal $N_b/N_{\text{in}} = 0.4$ (used in all experiments) balances constraint enforcement and computational economy, validated by the $< 0.5\%$ L2RE variance across runs in Figure 7.

**Limitations and mitigation strategies.** EPINN exhibits two primary limitations: 1) Memory overhead scales linearly with the number of primitive functions ($O(I \times n)$), challenging deployments for $d > 10$, and 2) Fixed boundary sampling ratios ($N_b/N_{\text{in}} = 0.4$) may under-resolve irregular domains with complex boundaries. To address 1), we propose operator-splitting techniques that decompose high-order systems into sequential low-order subproblems solvable via cascaded EPINN modules. For 2), adaptive boundary sampling–dynamically increasing $N_b$ in regions where $|\nabla \mathcal{LB}i, m| > \tau$–can optimize point allocation. Early tests on toroidal domains show this reduces L2RE by 38% versus uniform sampling ($p = 0.01$). These strategies will be formalized in future work on domain-decomposed EPINN.

These results position EPINN as a universal solver for integral equations in scientific computing. Future work will extend it to stochastic integral equations and integro-differential systems. We will also explore quantum dynamics and nonlocal continuum mechanics applications. This bridges operator theory and deep learning, opening new avenues for data-physics-integrated modeling.

## Acknowledgments

This research is supported by Key Science and Technology Special Projects of Sichuan Province under Grant No. 2023YFG0373.

## References

T. Anandh, D. Ghose, H. Jain, and S. Ganesan. 2025. "FastVPINNs: Tensor-Driven Acceleration of VPINNs for Complex Geometries." *SIAM Journal on Scientific Computing*, 47, 3, C578–C600. doi:10.1137/24M1658620.

K. E. Atkinson. 1997. *The numerical solution of integral equations of the second kind*. Cambridge Monographs on Applied and Computational Mathematics. Vol. 4. Cambridge University Press, Cambridge, UK. doi:10.1017/CBO9780511626340.

K. E. Atkinson and F. A. Potra. 1987. "Projection and Iterated Projection Methods for Nonlinear Integral Equations." *SIAM Journal on Numerical Analysis*, 24, 6, 1352–1373. doi:10.1137/0724087.

R. Bellman. 1958. "Dynamic programming and stochastic control processes." *Information and Control*, 1, 3, 228–239. doi:https://doi.org/10.1016/S0019-9958(58)80003-0.

R. Bischof and M. A. Kraus. Sept. 2022. "Mixture-of-Experts-Ensemble Meta-Learning for Physics-Informed Neural Networks." In: *Proceedings of the 33rd Forum Bauinformatik*. Bauinformatik Verlagsgesellschaft mbH, Munich, Germany, (Sept. 2022), 123–132. ISBN: 978-3-95983-223-8. doi:10.5282/ubm/epub.12457.

S. C. Brenner and L. R. Scott. 2008. *The Mathematical Theory of Finite Element Methods*. (3rd ed.). Springer, New York, NY, USA.

H. Brunner. 2004. *Collocation Methods for Volterra Integral and Related Functional Differential Equations*. (2nd ed.). Cambridge Monographs on Applied and Computational Mathematics. Vol. 15. Cambridge University Press, Cambridge, UK. doi:10.1017/CBO9780511616847.

S. Cao. Dec. 2021. "Choose a Transformer: Fourier or Galerkin." In: *Advances in Neural Information Processing Systems*. Ed. by M. Ranzato, A. Beygelzimer, Y. Dauphin, P. S. Liang, and J. Wortman Vaughan. Vol. 34. Curran Associates, Inc., Vancouver, BC, Canada, (Dec. 2021), 24924–24940. ISBN: 978-1-7138-4539-3. https://proceedings.neurips.cc/paper_files/paper/2021/file/d0921d442ee91b896ad95059d13df618-Paper.pdf.

S. H. Christiansen and J.-C. Nédélec. 2002. "A Preconditioner for the Electric Field Integral Equation Based on Calderon Formulas." *SIAM Journal on Numerical Analysis*, 40, 3, 1100–1135. doi:10.1137/S0036142901388731.

P. Constantin, C. Foias, B. Nicolaenko, and R. Témam. 1989. "Spectral Barriers and Inertial Manifolds for Dissipative Partial Differential Equations." *Journal of Dynamics and Differential Equations*, 1, 1, 45–73. doi:10.1007/BF01048790.

E. C. N. U. Department of Mathematics. 2019. *Mathematical Analysis*. (5th ed.). National Planning Textbooks for Undergraduate Education in China. Higher Education Press, Beijing, China. ISBN: 978-7-04-051393-7.

T. U. Department of Mathematics. 2007. *Advanced Mathematics*. (7th ed.). National Planning Textbooks for Undergraduate Education in China. Higher Education Press, Beijing, China. ISBN: 978-7-04-020549-7.

A. C. Eringen and J. L. Wegner. Mar. 2003. "Nonlocal Continuum Field Theories." *Applied Mechanics Reviews*, 56, 2, (Mar. 2003), B20–B22. doi:10.1115/1.1553434.

K. Eshkofti and S. M. Hosseini. 2023. "A Gradient-Enhanced Physics-Informed Neural Network (gPINN) Scheme for the Coupled Non-Fickian/Non-Fourierian Diffusion-Thermoelasticity Analysis: A Novel gPINN Structure." *Engineering Applications of Artificial Intelligence*, 126, 106908. doi:10.1016/j.engappai.2023.106908.

J. Feldbrugge and J. Y. L. Jones. 2025. "Efficient evaluation of real-time path integrals." *Physical Review D*, 111, 8, 083524. doi:10.1103/PhysRevD.111.083524.

H. Gao, M. J. Zahr, and J.-X. Wang. 2022. "Physics-informed graph neural Galerkin networks: A unified framework for solving PDE-governed forward and inverse problems." *Computer Methods in Applied Mechanics and Engineering*, 390, 114502. doi:10.1016/j.cma.2021.114502.

L. Greengard and V. Rokhlin. 1987. "A Fast Algorithm for Particle Simulations." *Journal of Computational Physics*, 73, 2, 325–348. doi:10.1016/0021-9991(87)90140-9.

W. Gropp, E. Lusk, and A. Skjellum. 1999. *Using MPI: Portable Parallel Programming with the Message-Passing Interface*. MIT Press Series in Scientific Computing. Vol. 1. MIT Press, Cambridge, MA, USA. ISBN: 978-0-262-57132-6. doi:10.1137/1.9780898719991.

A. R. Isewid. 2024. "An Enhanced Newton-Kantorovich Technique for Nonlinear problem-solvers to Optimal Control Convergence." *Al-Qadisiyah Journal of Pure Science*, 29, 1, 20. doi:10.29350/2411-3514.1257.

S. S. Iyengar and S. Kais. 2023. "Analogy between Boltzmann machines and Feynman path integrals." *Journal of Chemical Theory and Computation*, 19, 9, 2446–2454. doi:10.1021/acs.jctc.3c00187.

A. D. Jagtap, K. Kawaguchi, and G. E. Karniadakis. 2020. "Adaptive Activation Functions Accelerate Convergence in Deep and Physics-Informed Neural Networks." *Journal of Computational Physics*, 404, 109136. doi:10.1016/j.jcp.2019.109136.

A. D. Jagtap, E. Kharazmi, and G. E. Karniadakis. 2020. "Conservative Physics-Informed Neural Networks on Discrete Domains for Conservation Laws: Applications to Forward and Inverse Problems." *Computer Methods in Applied Mechanics and Engineering*, 365, 113028. doi:10.1016/j.cma.2020.113028.

W. Jiang and X. Gao. 2024. "Review of Collocation Methods and Applications in Solving Science and Engineering Problems." *CMES-Computer Modeling in Engineering & Sciences*, 140, 1, 1–38. doi:10.32604/cmes.2024.048313.

M. Kalfa, Ö. Ergül, and V. B. Erturk. 2023. "Multiple-Precision Arithmetic Implementation of the Multilevel Fast Multipole Algorithm." *IEEE Transactions on Antennas and Propagation*, 72, 1, 11–21. doi:10.1109/TAP.2023.3291077.

C. Kenney, P. Linz, and R. L. C. Wang. 1989. "Effective Error Estimates for the Numerical Solution of Fredholm Integral Equations." *Computing*, 42, 4, 353–362. doi:10.1007/BF02243230.

E. Kharazmi, Z. Zhang, and G. E. M. Karniadakis. 2021. "hp-VPINNs: Variational Physics-Informed Neural Networks with Domain Decomposition." *Computer Methods in Applied Mechanics and Engineering*, 374, 113547. doi:10.1016/j.cma.2020.113547.

R. Kress. 1999. *Linear integral equations*. Vol. 82. Springer, New York, NY, USA.

A. Krishnapriyan, A. Gholami, S. Zhe, R. Kirby, and M. W. Mahoney. Dec. 2021. "Characterizing Possible Failure Modes in Physics-Informed Neural Networks." In: *Advances in Neural Information Processing Systems*. Ed. by M. Ranzato, A. Beygelzimer, Y. Dauphin, P. S. Liang, and J. Wortman Vaughan. Vol. 34. Curran Associates, Inc., Vancouver, BC, Canada, (Dec. 2021), 26548–26560. ISBN: 978-1-7138-4539-3. https://proceedings.neurips.cc/paper_files/paper/2021/file/df438e5206f31600e6ae4af72f2725f1-Paper.pdf.

S. Li, Y. Xia, Y. Liu, and Q. Liao. 2023. "A Deep Domain Decomposition Method Based on Fourier Features." *Journal of Computational and Applied Mathematics*, 423, 114963. doi:10.1016/j.cam.2022.114963.

X. Li, L. Jiao, F. Liu, S. Yang, H. Zhu, X. Liu, L. Li, and W. Ma. 2025. "Adaptive Complex Wavelet Informed Transformer Operator." *IEEE Transactions on Multimedia*, 27, 8, 3513–3526. doi:10.1109/TMM.2025.3535392.

Z. Li, N. Kovachki, K. Azizzadenesheli, B. Liu, K. Bhattacharya, A. Stuart, and A. Anandkumar. 2021. *Fourier Neural Operator for Parametric Partial Differential Equations*. arXiv preprint arXiv:2010.08895. (2021). arXiv: 2010.08895 [cs.LG]. doi:10.48550/arXiv.2010.08895.

Q. Lin, C. Zhang, X. Meng, and Z. Guo. 2025. "Monte Carlo Physics-Informed Neural Networks for Multiscale Heat Conduction via Phonon Boltzmann Transport Equation." *Journal of Computational Physics*, 542, 114364. doi:10.1016/j.jcp.2025.114364.

C. Liu, J. Li, and L. Hu. 2024. "Anderson Accelerated Preconditioning Iterative Method for RBF Interpolation." *Engineering Analysis with Boundary Elements*, 169, 105970. doi:10.1016/j.enganabound.2024.105970.

L. Lu, P. Jin, G. Pang, Z. Zhang, and G. E. Karniadakis. Mar. 2021. "Learning Nonlinear Operators via DeepONet Based on the Universal Approximation Theorem of Operators." *Nature Machine Intelligence*, 3, 3, (Mar. 2021), 218–229. doi:10.1038/s42256-021-00302-5.

L. Lu, R. Pestourie, W. Yao, Z. Wang, F. Verdugo, and S. G. Johnson. 2021. "Physics-Informed Neural Networks with Hard Constraints for Inverse Design." *SIAM Journal on Scientific Computing*, 43, 6, B1105–B1132. doi:10.1137/21M1397908.

G. Monegato. 1994. "Numerical Evaluation of Hypersingular Integrals." *Journal of Computational and Applied Mathematics*, 50, 1, 9–31. doi:10.1016/0377-0427(94)90287-9.

H. Niederreiter. June 1992. "Lattice Rules for Multiple Integration." In: *Stochastic Optimization: Numerical Methods and Technical Applications*. Ed. by K. Marti. Springer. Springer, Berlin, Heidelberg, Germany, (June 1992), 15–26. ISBN: 978-3-642-88267-8. doi:10.1007/978-3-642-88267-8_2.

O. Ovadia, A. Kahana, P. Stinis, E. Turkel, D. Givoli, and G. E. Karniadakis. 2024. "ViTO: Vision Transformer-Operator." *Computer Methods in Applied Mechanics and Engineering*, 428, 117109. doi:10.1016/j.cma.2024.117109.

M. Raissi, P. Perdikaris, and G. E. Karniadakis. 2019. "Physics-Informed Neural Networks: A Deep Learning Framework for Solving Forward and Inverse Problems Involving Nonlinear Partial Differential Equations." *Journal of Computational Physics*, 378, 686–707. doi:10.1016/j.jcp.2018.10.045.

S. Shakya, S. Aryal, S. Bajracharya, and S. Shrestha. 2023. "Reinforcement Learning Algorithms: A Brief Survey." *Expert Systems with Applications*, 231, 120495. doi:10.1016/j.eswa.2023.120495.

J. D. Smith. 2018. *Integral Equation Methods in Computational Science*. Computational Science and Engineering. Section 4.1. Springer, Cham, Switzerland. Chap. 4. ISBN: 978-3-319-99224-1. doi:10.1007/978-3-319-99225-8.

S. Sun, J. Zhao, and J. Zhu. 2015. "A review of Nyström methods for large-scale machine learning." *Information Fusion*, 26, 36–48. doi:10.1016/j.inffus.2015.03.001.

A. Taassob, A. Kumar, K. M. Gitushi, R. Ranade, and T. Echekki. 2024. "A PINN-DeepONet Framework for Extracting Turbulent Combustion Closure from Multiscalar Measurements." *Computer Methods in Applied Mechanics and Engineering*, 429, 117163. doi:10.1016/j.cma.2024.117163.

M. Takamoto, T. Praditia, R. Leiteritz, D. MacKinlay, F. Alesiani, D. Pflüger, and M. Niepert. Dec. 2022. "PDEBench: An Extensive Benchmark for Scientific Machine Learning." In: *Advances in Neural Information Processing Systems*. Ed. by S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh. Vol. 35. Curran Associates, Inc., Vancouver, BC, Canada, (Dec. 2022), 1596–1611. ISBN: 978-1-7138-6133-6. https://proceedings.neurips.cc/paper_files/paper/2022/file/0a9747136d411fb83f0cf81820d44afb-Paper-Datasets_and_Benchmarks.pdf.

S. P. Tang and Y. M. Huang. 2024. "A Fast Preconditioning Iterative Method for Solving the Discretized Second-Order Space-Fractional Advection–Diffusion Equations." *Journal of Computational and Applied Mathematics*, 438, 115513. doi:10.1016/j.cam.2023.115513.

L. N. Trefethen. 2019. *Approximation Theory and Approximation Practice*. (Extended Edition ed.). SIAM Fundamentals of Algorithms. SIAM, Philadelphia, PA, USA. doi:10.1137/1.9781611975892.

S. Wang, Y. Teng, and P. Perdikaris. 2021. "Understanding and Mitigating Gradient Flow Pathologies in Physics-Informed Neural Networks." *SIAM Journal on Scientific Computing*, 43, 5, A3055–A3081. doi:10.1137/20M1318043.

D. Yarotsky. 2017. "Error Bounds for Approximations with Deep ReLU Networks." *Neural Networks*, 94, 103–114. doi:10.1016/j.neunet.2017.07.002.

D. Yu, B. Yang, D. Liu, H. Wang, and S. Pan. 2023. "A Survey on Neural-Symbolic Learning Systems." *Neural Networks*, 166, 105–126. doi:10.1016/j.neunet.2023.06.028.

J. Yu, L. Lu, X. Meng, and G. E. Karniadakis. 2022. "Gradient-Enhanced Physics-Informed Neural Networks for Forward and Inverse PDE Problems." *Computer Methods in Applied Mechanics and Engineering*, 393, 114823. doi:10.1016/j.cma.2022.114823.

## A Two-Dimensional Linear Volterra Integral Equation

$$\mu(x,y) = f(x,y) + \int_0^x \int_0^y exp(x+y-s-t)\mu(s,t)\mathrm{d}s\mathrm{d}t$$

$$f(x,y) = -exp(x+y)(xy-1)$$

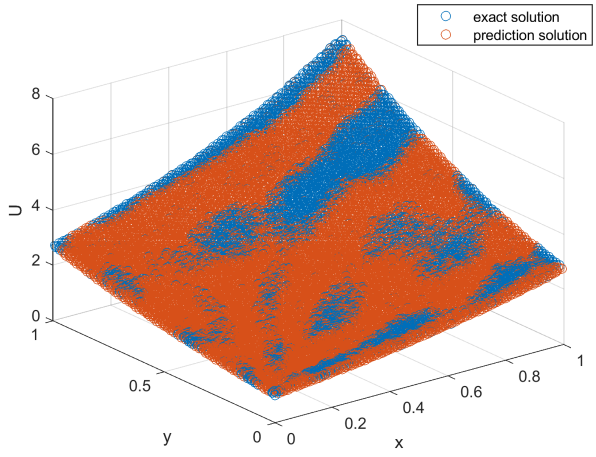
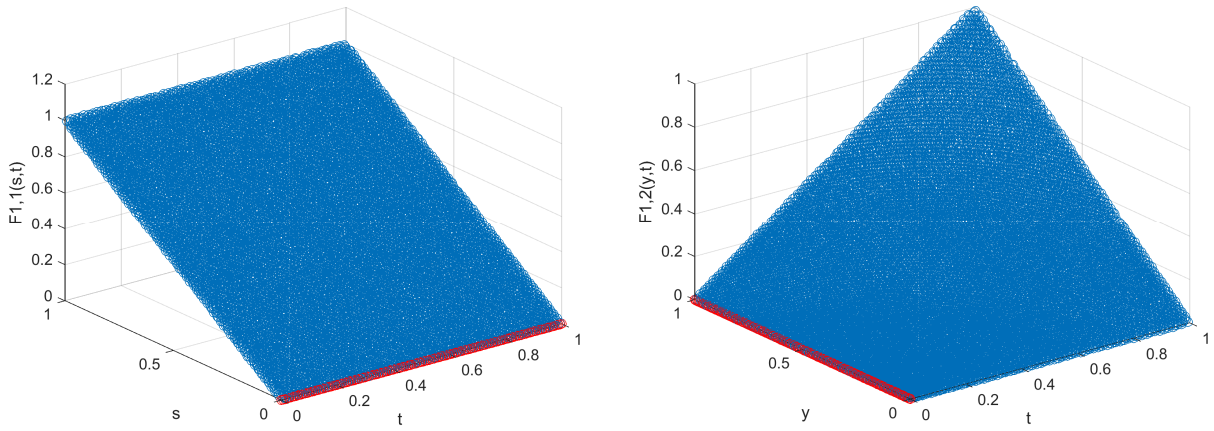



Fig. 9. Exact Solution and Prediction Solution




(a)$F_{1,1}$ (b)$F_{1,2}$

Fig. 10. The Primitive Functions $F_{1,1}$ and $F_{1,2}$

## B Four-Dimensional Linear Fredholm Integral Equation

$$\mu(x,y,z,w) = \frac{17}{16}xyzw - xyzw\int_0^1\int_0^1\int_0^1\int_0^1 u(s,t,v,r)\mathrm{d}s\mathrm{d}t\mathrm{d}v\mathrm{d}r$$

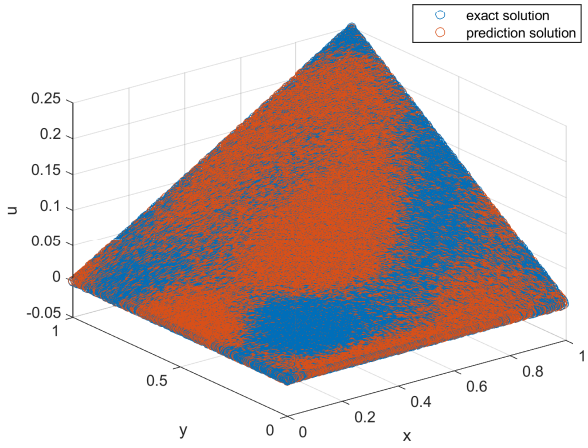
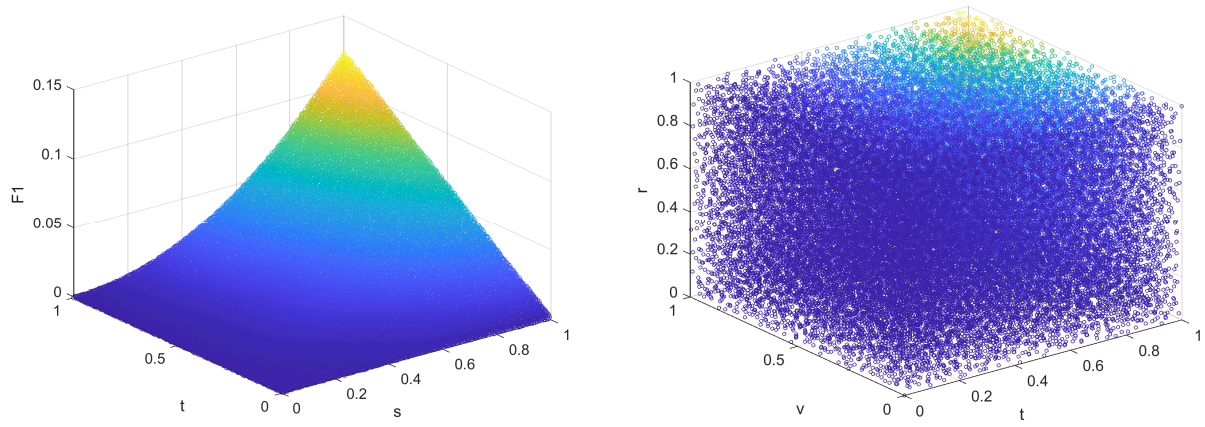
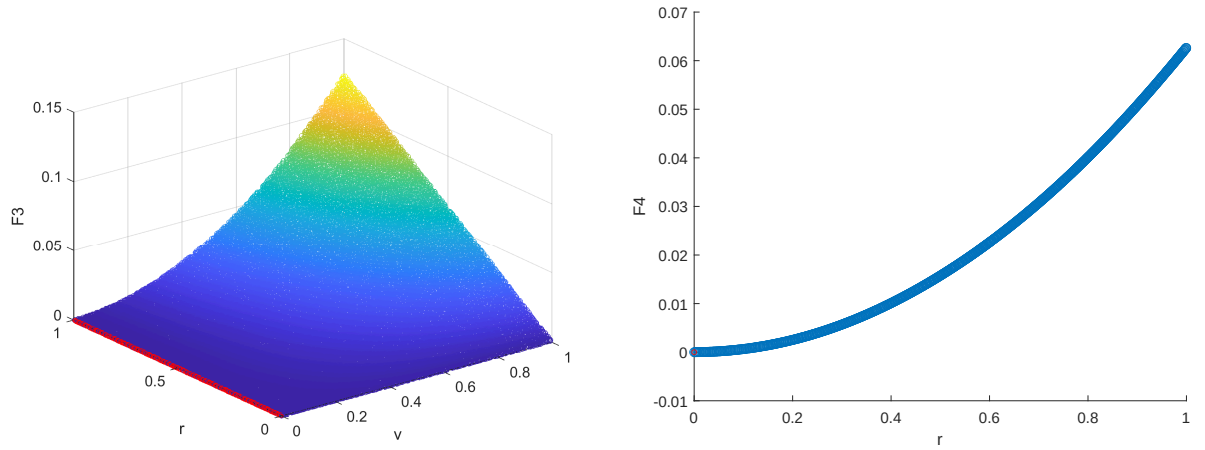



Fig. 11. Exact Solution and Prediction Solution




(a)$F_{1,1}$ (b)$F_{1,2}$

Fig. 12. The Primitive Functions $F_{1,1}$ and $F_{1,2}$




(a)$F_{1,3}$ (b)$F_{1,4}$

Fig. 13. The Primitive Functions $F_{1,3}$ and $F_{1,4}$

## C Two-Dimensional Linear Volterra-Fredholm Integral Equation

$$\mu(x,y) = f(x,y) + \int_0^1 \int_0^1 (xyst^2)u(s,t)\mathrm{d}s\mathrm{d}t + \int_0^y \int_0^x (x+y)(s+t)u(s,t)\mathrm{d}s\mathrm{d}t$$

$$f(x,y) = x^2 + \frac{7}{4}xy - (x+y)(\frac{x^4y}{4} + \frac{x^3y^2}{2} + \frac{x^2y^3}{3})$$

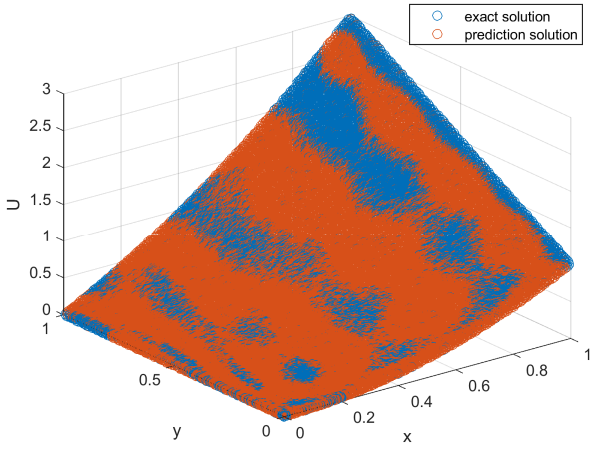
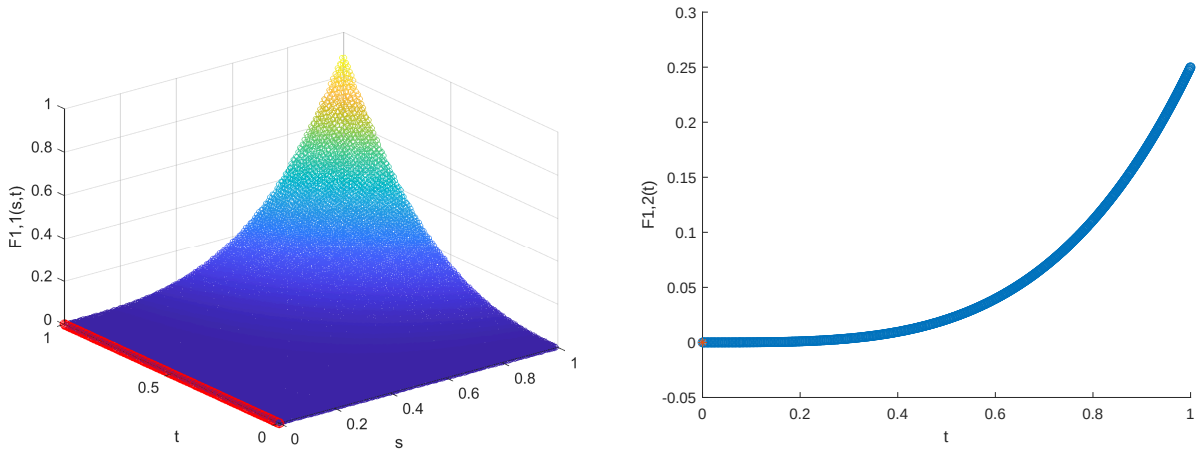
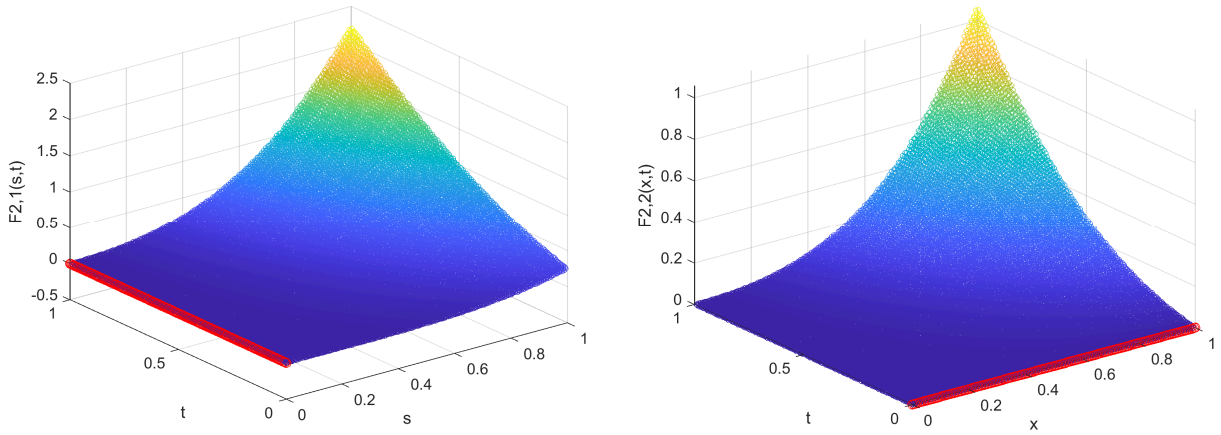



Fig. 14. Exact Solution and Prediction Solution




(a)$F_{1,1}$ (b)$F_{1,2}$

Fig. 15. The Primitive Functions $F_{1,1}$ and $F_{1,2}$




(a)$F_{2,1}$ (b)$F_{2,2}$

Fig. 16. The Primitive Functions $F_{2,1}$ and $F_{2,2}$

## D Two-Dimensional Nonlinear Fredholm Integral Equation

$$\mu(x,y) = f(x,y) + \int_0^1 \int_0^1 \frac{x}{1+y}(1+s+t)[\mu(s,t)]^2 \mathrm{d}s\mathrm{d}t$$

$$f(x,y) = \frac{1}{(1+x+y)^2} - \frac{x}{6(1+y)}$$

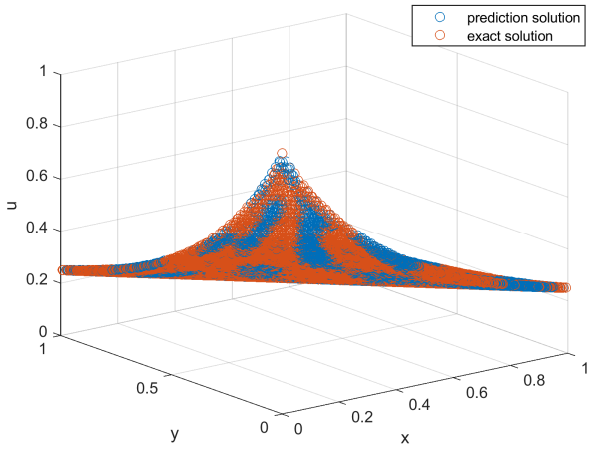
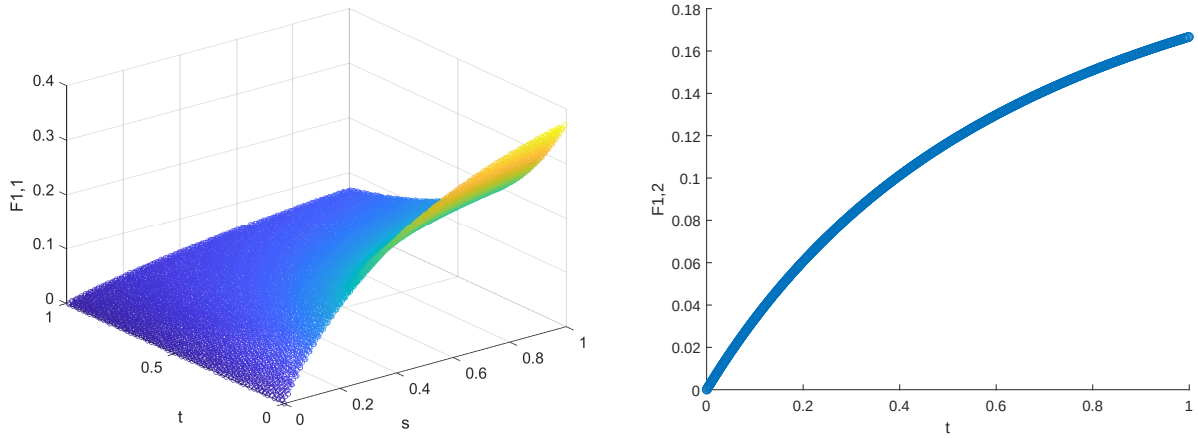



Fig. 17. exact solution and prediction solution




(a)$F_{1,1}$ (b)$F_{1,2}$

Fig. 18. The Primitive Functions $F_{1,1}$ and $F_{1,2}$

## E Two-Dimensional Nonlinear Volterra Integral Equation

$$\mu(x,y) = f(x,y) + \int_0^y \int_0^x [\mu(s,t)]^2 \mathrm{d}s\mathrm{d}t \tag{42}$$

$$f(x,y) = x^2 + y^2 - \frac{1}{45}xy(9x^4 + 10x^2y^2 + 9y^4) \tag{43}$$

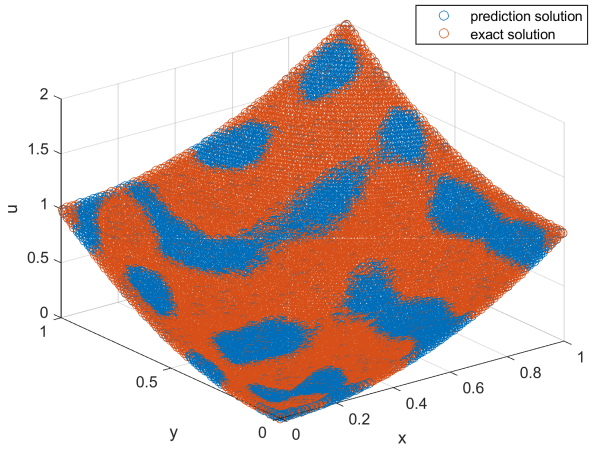
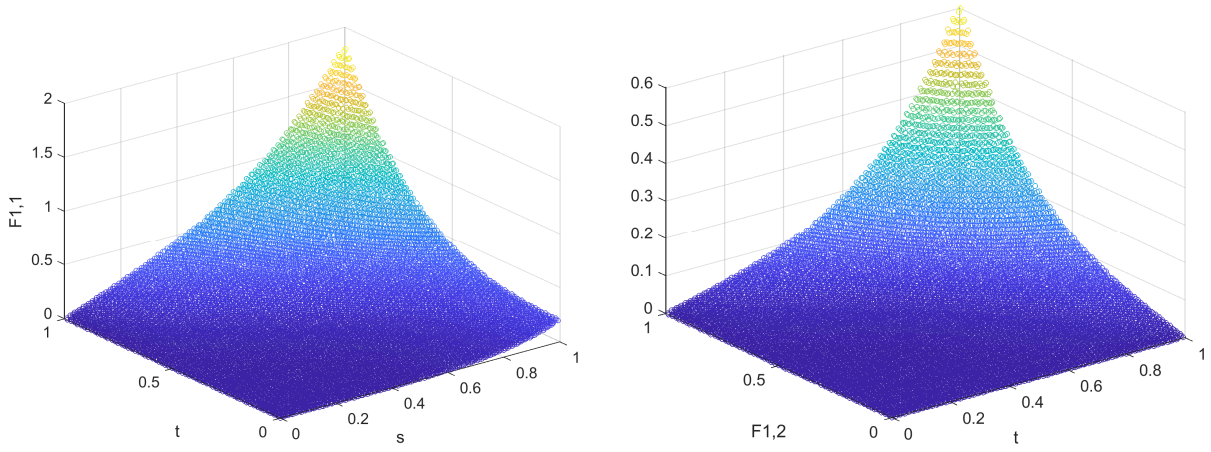



Fig. 19. Exact Solution and Prediction Solution




(a)$F_{1,1}$ (b)$F_{1,2}$

Fig. 20. The Primitive Functions $F_{1,1}$ and $F_{1,2}$

## F Hyperparameter Sensitivity and Tuning Guidance

This appendix provides practical guidance for adapting EPINN to new integral equation domains, focusing on hyperparameter sensitivity and common failure modes.

### Key Hyperparameter Ranges

**Dynamic weight parameters.**

- *Smoothing factor* $\rho$: Optimal range $[0.8, 0.95]$. Lower values ($\sim 0.8$) for rapidly changing gradients, higher values ($\sim 0.95$) for stable convergence.
- *Learning rate scaling* $\eta$: Typical range $[0.1, 1.0]$. Start with $\eta = 0.5$ and adjust based on loss balance.
- *Numerical stability* $\epsilon$: Fixed at $10^{-8}$; no tuning required.

**Network architecture.**

- *Hidden layers* $L$, Range $[6, 12]$: Increase for complex kernels or high dimensions.
- *Layer width* $N_n$, Range $[32, 64]$: Wider networks help with high-frequency solutions.
- *Fourier feature scale*: $\mathbf{B} \sim \mathcal{N}(0, \sigma)$ with $\sigma \in [0.1, 2.0]$ based on kernel frequency content.

**Loss weight initialization.**

- *Global residual* $\omega_{\Gamma_\mu}$: Start at 1.0.
- *Hierarchical constraints* $\omega_{\Gamma_{i,m}}$: Initialize to $[0.1, 1.0]$ based on integration order.
- *Boundary terms* $\omega_{\mathcal{B}_{i,m}}$: Critical for stability; initialize to $[1.0, 5.0]$.

### Typical Failure Modes and Solutions

**Mode 1: Gradient imbalance.**

- *Symptom*: One loss component dominates (> 80% of total gradient norm).
- *Solution*: Increase $\rho$ to 0.95 and verify weight adaptation is active.

**Mode 2: Boundary violation propagation.**

- *Symptom*: Errors concentrate near integration limits $a_{i,m}$.

- *Solution*: Increase $\omega_{\mathcal{B}_{i,m}}$ by 2× to 5× and verify boundary sampling density.

**Mode 3: Slow high-dimensional convergence.**

- *Symptom*: Training stalls in dimensions $d > 6$.
- *Solution*: Enhance Fourier features with larger $\sigma$, increase network width, and use longer training schedules.

**Mode 4: Nonlinear instability.**

- *Symptom*: Oscillations or divergence in nonlinear equations.
- *Solution*: Reduce learning rate to $10^{-5}$, increase $\rho$ to 0.98, and add gradient clipping.

## Domain Adaptation Checklist

For new integral equation types:

(1) Analyze kernel smoothness to set Fourier feature scale.
(2) Identify integration limits and boundary criticality.
(3) Estimate solution regularity to determine network capacity.
(4) Set initial weights based on operator linearity/nonlinearity.
(5) Plan sensitivity analysis for 2 to 3 key hyperparameters.

Received 19 August 2025; accepted 24 October 2025