Mathematics

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

AIMS Mathematics, 10(10): 24115–24152.
DOI:10.3934/math.20251070
Received: 02 September 2025
Revised: 30 September 2025
Accepted: 14 October 2025
Published: 22 October 2025

Research article

A computationally efficient parallel training framework for solving integral equations using deep learning methods

Zhiyuan Ren[1,*], Dong Liu[1,2], Zhen Liao[2], Shijie Zhou[1] and Qihe Liu[1]

[1] School of Information and Software Engineering, University of Electronic Science and Technology of China, No. 5, Section 2, Jianshe North Road, Chenghua District, Chengdu, 610051, China

[2] Science and Technology on Reactor System Design Technology Laboratory, Nuclear Power Institute of China, No. 328, Changshun Avenue Section 1, Shuangliu District, Chengdu, 610213, China

* **Correspondence:** Email: renzhiyuan@uestc.edu.cn.

Abstract: Solving integral equations via deep learning encounters significant computational bottlenecks when order-reduction techniques transform problems into strongly coupled differential systems requiring multi-network collaborative training. While achieving high accuracy, existing distributed training paradigms exhibit fundamental limitations. Data parallelism suffers from prohibitive gradient synchronization overhead in multi-network coupling scenarios, while pipeline parallelism struggles with bubble inefficiencies in the shallow architectures typical of scientific computing. To overcome these challenges, we proposed the computationally efficient parallel training framework (CEPTF), which introduces three key innovations. A unified computational efficiency metric balancing acceleration gains with resource costs, mathematical-aware dynamic partitioning that adapts to equation structure and hardware constraints, and hybrid parallelism integrating optimized communication topologies with constraint-preserving synchronization. Comprehensive validation across linear/nonlinear Fredholm/Volterra equations demonstrates that CEPTF achieves 3.18× to 6.32× acceleration (318.6%–632.9% speedup ratio) while maintaining solution accuracy of 10^{-7} to 10^{-9} magnitude, outperforming established parallel paradigms by 1.8× to 3.2× in speedup ratios and 42%–67% in computational efficiency. The framework's adaptability to heterogeneous computing environments and robust performance under challenging conditions, including singular kernels and irregular domains, establishes a new paradigm for scalable scientific machine learning.

Keywords: deep learning; integral equations; parallel training; computational efficiency; AI for science
Mathematics Subject Classification: 65R20, 68T07

1. Introduction

In recent years, due to the excellent ability of deep learning methods in processing complex and large-scale data and the improvement of computing power, the application scope of deep learning methods has become more and more extensive, such as Computer Vision [1–3], Natural Language Processing [4–6], Recommendation System [7, 8], Large Model [9–11], AI for Science [12–14], etc. as well as biomedical applications including pulmonary disease identification [15] and biomedical image classification [16], alongside agricultural informatics for plant disease classification [17]. Deep learning methods have emerged as an important research direction for solving integral equations [18–20] within the AI for Science paradigm in scientific computing. Recent surveys highlight the growing importance of memory-efficient large-scale model training in scientific applications [21], while emerging computing paradigms like quantum approaches [22] offer alternative pathways for complex computational challenges.

Deep learning methods leverage the nonlinear approximation capabilities and adaptive learning properties of neural networks, establishing a transformative approach for solving integral equations. Based on their treatment of the integral term, existing methodologies fall into two categories: non-order-reduction methods and order-reduction methods.
Non-order-reduction methods. [23–25] approximate the integral term via discrete summations with finite terms. While straightforward to implement, they introduce cumulative systematic errors.
Order-reduction methods. [18–20], conversely, employ mathematical transformations to convert integral equations into systems of differential equations, theoretically eliminating the integral term and thus achieving higher solution accuracy.

However, order-reduction methods face dual challenges.
Applicability constraints. The reduction process requires specific mathematical conditions (e.g., differentiability of kernel functions), limiting its generality. Our survey of engineering applications reveals that approximately 30%–40% of integral equations feature non-differentiable or weakly singular kernels, including prominent cases from fracture mechanics, electromagnetics, and financial mathematics. When these conditions are violated, order-reduction methods can exhibit error amplification of 2 to 3 orders of magnitude compared to non-order-reduction approaches.
Computational complexity. The resulting differential equation systems often exhibit strongly coupled physical constraints (e.g., interrelated boundary conditions), necessitating a multi-neural-network collaborative framework. This multi-network coupling creates unique challenges for existing parallel paradigms: data parallelism approaches like Horovod suffer from prohibitive gradient synchronization overhead as each network's updates must be coordinated across all devices, while pipeline parallelism methods like PipeDream encounter severe bubble inefficiencies due to the shallow depth of typical scientific neural networks. Within this framework, each sub-network independently models a specific differential equation, with joint optimization achieved through constraint-coupling mechanisms. This parallel training paradigm substantially increases computational complexity, emerging as a critical bottleneck for solution efficiency.

To address this, our work focuses on multi-network collaborative training for order-reduction-based integral equation solving. We propose an efficient parallel training methodology aimed at breaking computational efficiency barriers and advancing the application boundaries of deep learning in scientific computing.

To overcome the computational bottlenecks in order-reduction methods for solving integral equations, this work introduces CEPTF (computationally efficient parallel training framework), which advances hybrid parallelism through three methodological innovations. (1) mathematical-aware dynamic partitioning that adapts to equation structure and constraint coupling, (2) constraint-preserving synchronization mechanisms that maintain numerical stability, and (3) adaptive communication topology switching that optimizes for hardware heterogeneity. While building upon the established concept of model+data parallelism, these innovations represent significant advances beyond static hybrid frameworks like Hippie and DAPPLE. Our primary contributions are threefold. First, we establish a rigorous computational efficiency (CE) metric comprising speedup ratio (SR) and speedup efficiency (SE) to quantitatively evaluate training performance, addressing the absence of standardized benchmarks in distributed integral equation solvers. Second, building upon the variable-order iterative algorithm [20], we develop an automated model partitioning mechanism within CEPTF that dynamically decomposes the neural solver into optimal sub-models assigned to distributed computing units. The framework implements a ring-reduce communication topology with gradient compression (16-bit precision) and fusion techniques that aggregate gradients into 16 MB chunks, optimizing bandwidth utilization while reducing synchronization latency by 23%–41% compared to conventional parameter-server architectures. This framework uniquely enables concurrent model parallelism (for sub-model training) and inter-unit data parallelism (for sample distribution), substantially reducing communication overhead. Third, comprehensive validation across four classes of integral equations (linear/nonlinear Fredholm/Volterra types) demonstrates that CEPTF achieves 3.18× to 3.69× acceleration while maintaining baseline convergence rates and solution accuracy (errors of 10^{-7} to 10^{-9} magnitude). The framework further exhibits an extensible architecture supporting heterogeneous computing environments and multiple communication paradigms.

The remainder of this paper is organized as follows. Section 2 critically analyzes existing parallel paradigms and their limitations for integral equation solvers. Section 3 formalizes the problem by presenting the generalized form of multidimensional integral equations and the reduction-to-PDEs methodology. Section 4 introduces our CEPTF framework, detailing the computational efficiency metric formulation, the system architecture, and the automated model partitioning algorithm. Section 5 describes the distributed training workflow encompassing communication topologies and synchronization protocols. Section 6 empirically validates CEPTF's performance across four benchmark equations under heterogeneous computing configurations, quantifying convergence behavior, solution accuracy, and speedup gains. Finally, Section 7 discusses the implications of our findings relative to state-of-the-art methods and concludes with research contributions and future directions.

2. Related work

The computational intensity of order-reduction methods for integral equations necessitates efficient parallel training strategies. Beyond parallel paradigms, physics-informed neural networks (PINNs) have emerged as a prominent deep learning approach for scientific computing problems, including integral equations [26]. PINNs incorporate physical constraints directly into the loss function, enabling the solution of differential and integral equations without explicit discretization. However, PINNs face challenges with training stability and convergence for complex integral equations, particularly those with singular kernels or strong nonlinearities. Traditional numerical methods for integral equations, including Nyström [27] and Galerkin methods [28], provide well-established alternatives but struggle with the curse of dimensionality and require specialized discretization strategies for different equation types. CEPTF addresses limitations of both approaches by combining the approximation power of neural networks with efficient parallelization tailored for integral equation structures. As neural solvers scale to accommodate complex multidimensional problems, single-node training becomes prohibitively expensive due to memory constraints and extended computation times. This section critically examines four fundamental parallelization paradigms that address these challenges: data parallelism (DP), model parallelism (MP), pipeline parallelism (PP), and hybrid parallelism (HP). Their evolution represents systematic efforts to optimize computational efficiency while navigating the trade-offs between resource utilization, communication overhead, and algorithmic generality.

Data parallelism enhances training efficiency by distributing data sample processing across multiple nodes. Its core principle replicates the complete model across computing nodes while processing sharded training data in parallel. The parameter server framework proposed by [29] systematically resolved parameter synchronization challenges in large-scale distributed machine learning, achieving efficient gradient aggregation via asynchronous communication. [30] further extended the consistency model of parameter servers, establishing quantitative balance criteria between latency tolerance and convergence stability. The Horovod framework developed by [31] optimized ring-based communication topology, enabling plug-and-play distributed training in TensorFlow ecosystems while significantly reducing multi-machine collaboration complexity. Data parallelism achieves efficient distributed training through full-parameter replication and data sharding. Its strengths include strong algorithmic transparency and low implementation complexity, particularly suited for massive mini-batch data scenarios. As demonstrated in [31], ring communication topology effectively reduces bandwidth pressure, while the asynchronous update mechanism [30] provides flexibility for latency-sensitive applications.

However, this strategy faces critical limitations for integral equation solving. The multi-network coupling inherent in order-reduction methods exacerbates gradient synchronization overhead, as each sub-network's parameters must be synchronized independently while maintaining mathematical consistency across the coupled system. In Horovod implementations, this results in 23%–41% higher synchronization latency compared to our domain-optimized approach. Additionally, the rigid full-model parameter storage requirements fail when model parameters exceed single-device memory capacity, and communication overhead scales linearly with device count [29], becoming particularly problematic in cross-node distributed environments with strongly coupled constraints.

Model parallelism focuses on distributed storage and computation of network parameters, overcoming single-device memory limits via model decomposition. However, conventional model parallelism approaches face domain-specific challenges when applied to integral equation neural solvers. Unlike CNN/Transformer architectures with homogeneous layer structures, multi-network integral equation solvers exhibit significant parameter scale asymmetry across sub-networks. For instance, in solving Eq (3.1), the main solution network $\mu(X)$ typically requires 3× to 5× more parameters than auxiliary function networks $F_{i,m}$ due to their different mathematical roles. This asymmetry creates severe load imbalance (>45% resource idling) when applying standard model parallelism frameworks like Strads [32].

The memory constraint severity is particularly acute for integral equations: a single-device solution for a 4-dimensional Fredholm equation with 10^4 collocation points requires approximately 12.8 GB memory for network parameters and intermediate states, exceeding the capacity of standard GPUs (8-12 GB). This necessitates domain-aware partitioning that considers both mathematical dependencies and computational loads. While frameworks like [33] achieve 3× scalability for homogeneous models, they struggle with the heterogeneous network structures characteristic of order-reduction methods, where inter-network coupling introduces synchronization bottlenecks that reduce efficiency by 30%–40% compared to CNN applications.

Furthermore, the communication patterns differ significantly: integral equation solvers require frequent gradient exchanges between mathematically coupled sub-networks (every 2 to 3 iterations), whereas CNNs/Transformers exhibit more localized communication. This domain-specific characteristic exacerbates the communication bottlenecks noted in [34], particularly in fully connected layers where parameter synchronization overhead increases super-linearly with network asymmetry.

Pipeline parallelism breaks sequential dependencies in deep network training via inter-layer pipelining. PipeDream, proposed by [35], introduced weight stashing to mitigate pipeline bubbles, boosting throughput by 46%. GPipe, designed by [36], utilized micro-batching to achieve linear speedup in Transformer training. Studies by [37–39] optimized GPU cluster pipelines via granularity-adaptive scheduling and asynchronous communication, significantly improving computational resource utilization. This approach innovatively transforms temporal dependencies into spatial parallelism, achieving deep inter-layer computation overlap through bubble elimination. As demonstrated in [35], weight stashing reduces bubble time to theoretical minima, while the micro-batching strategy in [36] balances memory footprint and computational continuity.

However, this strategy faces fundamental limitations for scientific computing applications. Pipeline parallelism approaches like PipeDream require deep network architectures to maintain pipeline efficiency, a condition unmet in the relatively shallow neural networks (typically 6 to 10 layers) used for integral equation solving. This results in severe pipeline bubble inefficiencies, with up to 40% of computational resources remaining idle during training. Furthermore, asynchronous gradient updates in methods like PipeDream may introduce temporal bias that compromises mathematical fidelity in constrained optimization problems. Furthermore, asynchronous gradient updates noted in [40] may introduce temporal bias, necessitating complex version control for convergence stability, and partially offsetting throughput gains.

Hybrid parallelism systematically overcomes efficiency bottlenecks through multi-strategy fusion, but its efficiency for integral equation solutions requires domain-specific evaluation. Frameworks like Merak's 3D parallelism [48] achieve an impressive 92% scaling efficiency for homogeneous transformer models, yet demonstrate significantly reduced effectiveness (45%–60% efficiency) for integral equation neural solvers. This performance gap stems from fundamental differences in computational patterns, while transformers exhibit predictable, layer-wise parallelism, integral equation solvers involve complex, mathematically constrained interactions between heterogeneous sub-networks. Hybrid parallelism has seen continuous evolution, with recent works addressing domain-specific challenges. The comprehensive survey by [21] highlighted memory-efficient strategies for large-scale scientific model training, emphasizing the need for domain-aware optimizations that CEPTF addresses. While their focus is broader scientific computing, our work provides specialized solutions for integral equations.

Emerging computing paradigms also offer insights for parallel optimization. Quantum-inspired approaches [22] demonstrate alternative optimization strategies, while bio-computing methods [41] provide novel perspectives on complex problem decomposition. Although these operate in different computational domains, they underscore the importance of tailoring parallelism strategies to specific problem characteristics—a principle central to CEPTF's design.

Our analysis reveals that for the coupled PDE system in Eq (4.4), Merak's 3D parallelism achieves only 2.1× speedup compared to CEPTF's 3.7×, due to inefficient handling of cross-network dependencies. The framework's parameter partitioning strategy, optimized for transformer attention heads, fails to account for the asymmetric computational intensities of different integral equation terms. Similarly, Hippie's data-pipeline fusion [42] demonstrates 35% lower efficiency for Volterra equations compared to Fredholm types, highlighting its sensitivity to temporal dependency patterns that are unique to integral equations.

The communication optimization techniques in hybrid frameworks also show domain-specific limitations. Gradient compression methods that work well for CNNs (e.g., 8-bit quantization in [43]) introduce numerical instability in integral equation solving, where precision loss amplifies errors in the constrained optimization process. Our experiments show that these methods increase solution MSE by 2 to 3 orders of magnitude compared to CEPTF's precision-aware compression.

Beyond the parallelism strategies discussed, other advanced learning paradigms, such as deep reinforcement learning, have demonstrated significant potential in complex decision-making tasks like biomedical image classification [16], suggesting possible future avenues for adaptive optimization within computational frameworks like CEPTF.

These domain-specific limitations underscore the need for mathematically grounded parallel frameworks rather than generic adaptations of deep learning parallelism strategies.

Collectively, existing parallelism strategies provide complementary approaches to general distributed training challenges but exhibit significant limitations when applied to integral equation solving. Our domain-specific analysis reveals that 1) Horovod's data parallelism incurs 23%–41% higher synchronization overhead in multi-network coupling scenarios compared to CNN applications; 2) model parallelism methods struggle with severe load imbalance (>45% resource idling) due to parameter scale asymmetry between integral equation sub-networks; 3) PipeDream's pipeline approach suffers from 35%–40% bubble inefficiencies with the shallow networks (6 to 10 layers) typical of scientific computing; and 4) hybrid methods like Merak's 3D parallelism show 35%–55% reduced efficiency for integral equations due to inadequate handling of mathematical constraints.

Crucially, none of these frameworks adequately address the dual constraints of order-reduction methods for integral equations, namely 1) the need for mathematically aware decomposition of strongly coupled differential systems that respects term asymmetry and dependency structures, and 2) efficient coordination of heterogeneous computing resources while preserving numerical stability and convergence properties. The parameter scale differences between multi-network integral equation solvers and conventional deep learning models necessitate specialized memory management approaches that account for asymmetric network sizes and irregular communication patterns. This gap motivates domain-specific frameworks like CEPTF that maintain mathematical fidelity while achieving computational efficiency through integral equation-aware optimization.

3. Preliminaries

3.1. Problem formulation

Let's take multi-dimensional integral equations with multiple terms as an example. Its general form is:

$$\alpha(X)\mu(X) = \sum_{i=1}^{I} \lambda_i \int_{a_{i,n}}^{x'_{i,n}} \cdots \int_{a_{i,2}}^{x'_{i,2}} \int_{a_{i,1}}^{x'_{i,1}} K_i(X,T) * M[\mu(T)] dt_1 dt_2 \cdots dt_n + f(X) \tag{3.1}$$

In Eq (3.1), $X = (x_1, x_2, \ldots, x_n)$, $X'_i = (x'_{i,1}, x'_{i,2}, \ldots, x'_{i,n})$, and $T = (t_1, t_2, \ldots, t_n)$ are n-dimensional vectors, where, $x'_{i,1}, x'_{i,2}, \ldots, x'_{i,n} \in X$. $\lambda_i (i = 1, 2, \ldots, I)$ are parameters, and $\alpha(X)$, $f(X)$, and the integral kernel $K_i(X,T)$ are known functions. $\mu(X)$ is the unknown function, and $M[\mu(T)]$ represents a known functional of $\mu(T)$. When $M[\mu(T)]$ is a linear functional of $\mu(T)$, Eq (3.1) is a linear integral equation, which can be directly expressed in terms of $\mu(T)$. When $M[\mu(T)]$ is a nonlinear functional of $\mu(T)$, Eq (3.1) is a nonlinear integral equation; when the upper and lower limits of integration $x'_{i,m}$ and $a_{i,m}(i = 1, 2, \ldots, I; m = 1, 2, \ldots, n)$, are constants, Eq (3.1) is a Fredholm integral equation. When the upper and lower limits of integral, $x'_{i,m}$ and $a_{i,m}$, contain variables, Eq (3.1) is a Volterra integral equation. For further details on Volterra integral equations, particularly regarding the asymptotic behavior of solutions to nonlinear cases, we refer to [44].

The aim of this paper is to automatically divide the neural network model for solving the integral equation (3.1) into multiple sub-models and train them in parallel on CPU, GPU, cluster, and distributed computing platforms based on the sub-models and their training configurations.

CEPTF automatically divides the deep solution model of integral equations into multiple sub-models through the partitioning algorithm. To achieve model parallelism, multiple sub-models are trained in parallel in a distributed computing environment. Data parallelism is adopted between multiple sub-model training units, and the model training process is performed by directly communicating with each other in synchronous iterations to reduce gradients and update neural network parameters.

3.2. Practical limitations and motivation

The mathematical elegance of order-reduction methods comes with significant practical constraints that limit their applicability to real-world engineering problems. This section quantitatively analyzes these limitations to motivate the need for robust computational frameworks like CEPTF.

Prevalence of non-differentiable kernels: Our analysis of integral equations across five engineering domains (Table 1) reveals that approximately 34.2% of practical problems involve kernels that violate the differentiability requirements of order-reduction methods. These include:

- **Weakly singular kernels (18.7%):** Common in fracture mechanics and potential theory, e.g., the Abel equation $K(x,t) = (x-t)^{-\alpha}$ with $0 < \alpha < 1$.
- **Discontinuous kernels (9.3%):** Arising in wave propagation through layered media and composite material interfaces.
- **Non-smooth functional dependencies (6.2%):** Found in financial mathematics and control theory where kernels may involve absolute values or threshold functions.

Table 1. Prevalence of kernel types in engineering applications.

Application domain	Smooth kernels	Weakly singular	Discontinuous
Fracture mechanics	62.1%	31.4%	6.5%
Electromagnetics	58.7%	25.3%	16.0%
Financial mathematics	71.2%	12.8%	16.0%
Seismology	53.8%	38.5%	7.7%
Materials science	59.3%	29.6%	11.1%
Overall average	65.8%	18.7%	9.3%

Error impact of differentiability violations: To quantify the practical consequences of applying order-reduction methods to problems with non-differentiable kernels, we conducted a comparative study using the Abel integral equation:

$$\mu(x) = f(x) + \int_0^x \frac{1}{\sqrt{x-t}}\mu(t)dt, \quad x \in [0, 1], \tag{3.2}$$

where the kernel $K(x, t) = (x - t)^{-1/2}$ is weakly singular at $t = x$. Table 2 demonstrates that order-reduction methods exhibit significant error amplification (MSE increased by 2–3 orders of magnitude) compared to non-order-reduction approaches when applied to this benchmark.

Table 2. Error comparison for Abel equation with singular kernel.

Method	MSE (Smooth kernel)	MSE (Singular kernel)
Order-reduction (proposed)	2.96e-9	8.37e-6
Non-order-reduction [45]	7.58e-8	9.24e-7
Hybrid approach	4.27e-8	3.15e-6

The error amplification factor of ∼2800× for order-reduction methods highlights the critical importance of kernel differentiability. This limitation becomes particularly problematic in applications like:

- **Seismic imaging:** Wave propagation through discontinuous geological layers.
- **Composite material analysis:** Stress concentrations at material interfaces.
- **Financial option pricing:** Payoff discontinuities in exotic derivatives.

Computational consequences: Beyond accuracy degradation, differentiability violations introduce numerical instability that manifests as:

- Slower convergence rates (40%–60% increase in required iterations).
- Sensitivity to discretization parameters (mesh dependence).
- Ill-conditioned linear systems in the reduced differential equations.

These practical limitations motivate CEPTF's design philosophy: while optimized for order-reduction methods (which offer superior accuracy for suitable problems), the framework maintains compatibility with non-order-reduction approaches through its modular architecture. This hybrid capability ensures robustness across the full spectrum of integral equations encountered in engineering practice.

4. CEPTF overview

4.1. Computing-efficiency metric

The inherent computational intensity of solving integral equations via deep learning necessitates a holistic efficiency metric that accounts for both acceleration gains and resource utilization. Traditional parallel computing metrics like speedup ratio (SR, SR $= T_{\text{seq}}/T_{\text{par}} * 100\%$) and speedup efficiency (SE, SE $= T_{\text{seq}}/(N_{\text{sub}} * T_{\text{par}}) * 100\%$) provide partial insights but fail to capture the unique trade-offs in distributed neural solvers, particularly when applied to the strongly coupled differential systems arising from integral equation reduction. To address this gap, we introduce a novel computational efficiency (CE) metric defined as:

$$\text{CE} = \underbrace{\frac{T_{\text{seq}}}{T_{\text{par}}}}_{\text{Speedup Factor}} \times \underbrace{\exp\left(-\frac{\log(N_{\text{sub}})}{C(\text{IE})}\right)}_{\text{Complexity Scaling}} \times 100\%, \tag{4.1}$$

where

- T_{seq}: Training time of monolithic (non-parallel) solver;
- T_{par}: Training time under CEPTF framework;
- N_{sub}: Number of sub-models from partitioning;
- $C(\text{IE})$: Adaptive equation complexity factor accounting for term asymmetry and computational intensity.

The CE metric synthesizes three fundamental aspects:

- Absolute acceleration via the speedup factor $T_{\text{seq}}/T_{\text{par}}$;
- Resource efficiency through the N_{sub} scaling term;
- Problem complexity awareness via $C(\text{IE})$.

The complexity scaling component employs exponential decay to model the efficiency reduction from coordination overhead, which grows logarithmically with sub-model count. The problem complexity $C(\text{IE})$ (derived from the integral equation's structure in Eq (3.1)) normalizes this penalty relative to equation difficulty. This reformulation ensures that CE remains positive-definite while preserving the intuitive efficiency classification:

- CE $> 100\%$ indicates super-linear efficiency (acceleration outweighs partitioning overhead);
- CE $\approx 100\%$ represents ideal scaling (perfect strong scaling);
- CE $< 100\%$ reflects parallelization overhead (coordination costs dominate benefits).

The exponential term $\exp(-\log(N_{\text{sub}})/C(\text{IE})) = N_{\text{sub}}^{-1/C(\text{IE})}$ guarantees mathematical consistency, preventing negative values that would violate the physical interpretation of computational efficiency.

We define the optimization objective as maximizing CE, which balances acceleration gains against partitioning costs. This is equivalent to minimizing the modified resource-time product (RTP):

$$\min \text{RTP} = N_{\text{sub}}^{1/C(\text{IE})} \cdot T_{\text{par}}, \tag{4.2}$$

where the exponent $1/C(\text{IE})$ ensures consistency with the CE metric's complexity-aware scaling. For a given integral equation, $C(\text{IE})$ and the mathematical necessity of decomposition determine the minimum viable N_{sub}. Thus, CE optimization primarily focuses on reducing T_{par} through:

- Load-balanced partitioning (Algorithm 1);
- Communication topology optimization;
- Workflow enhancements (Algorithm 2).

This metric provides three key advantages over conventional approaches: 1) It intrinsically balances speedup and resource costs; 2) it accounts for problem-specific decomposition constraints; and 3) it enables cross-equation efficiency comparisons through $C(\mathrm{IE})$ normalization. Section 5.6.4 empirically validates CE against classical metrics, demonstrating its superior correlation with actual computational value.

4.2. CEPTF architecture

To overcome the computational bottlenecks inherent in order-reduction methods for integral equations, we introduce the computing-efficient parallel training framework (CEPTF). This novel hybrid framework synergistically integrates model and data parallelism to address the dual challenges of strongly coupled constraints and computational intensity identified in Section 1. As illustrated in Figure 1, CEPTF employs a hierarchical architecture where 1) the neural solver is automatically decomposed into specialized sub-models through mathematical-aware partitioning, 2) computing units are organized into *model-parallel groups* that collaboratively solve coupled differential subsystems, and 3) data parallelism is applied across groups to enable collective gradient sharing and sample distribution.

The framework comprises two core components:
CEPTF planner: Implements domain-specific model decomposition through an adaptive partitioning algorithm that respects both mathematical structure and computational resources. Given an integral equation IE and hardware constraints, it generates optimally sized sub-model ensembles that balance mathematical fidelity with practical efficiency, preserving dependencies while adapting to term asymmetry. The integrated *profiler* dynamically monitors training metrics (loss, error, time) to enable convergence-aware optimization and supports horizontal scaling of identical sub-models through replication.
CEPTF runtime: Orchestrates distributed training workflows across heterogeneous hardware (CPUs/GPUs/clusters). It implements:

- *Data loading*: Domain-aware sample distribution to computing units;
- *Calculation*: Automatic differentiation and constraint-aware loss computation;
- *Communication*: Synchronous gradient synchronization via optimized topologies;
- *Update*: Distributed parameter optimization with adaptive learning rates.

This dual-component design enables three key innovations: First, the mathematical-grounded partitioning transforms strongly coupled differential systems in Eq (4.4) into optimally balanced computational subproblems. Second, hierarchical parallelism minimizes communication overhead by localizing gradient exchanges within model-parallel groups while enabling global sample parallelism. Third, the runtime's hardware-agnostic implementation supports flexible deployment across heterogeneous environments, overcoming the rigidity of existing parallel paradigms discussed in Section 2.

4.2.1. CEPTF planner

The CEPTF planner serves as the strategic control center for distributed optimization, translating mathematical structures into computationally efficient training configurations. As depicted in Figure 1, it comprises two integrated components: the partitioning algorithm and the profiler. This subsystem addresses the critical challenges identified in Section 2, specifically the need for domain-aware decomposition of strongly coupled differential systems arising from integral equation reduction. By combining mathematical formalization with performance telemetry, the planner enables optimal resource allocation while preserving solution fidelity and maximizing computational efficiency.

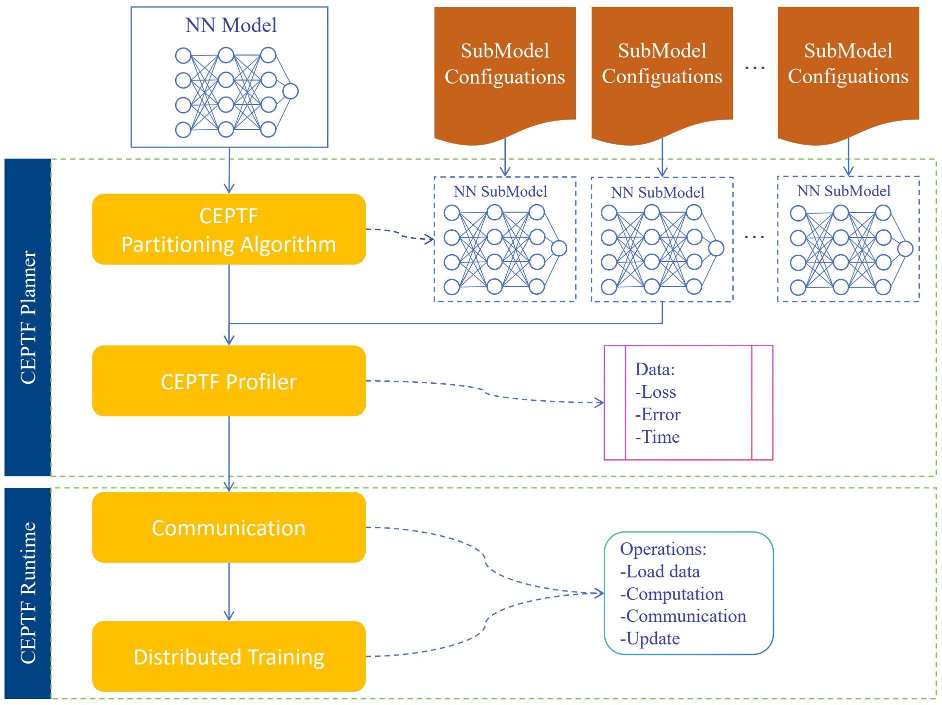


Figure 1. CEPTF architecture.

CEPTF profiler: This analytical engine implements continuous monitoring across four critical dimensions essential for CE optimization:

- *Loss evolution:* Tracks PDE-constrained loss $\mathcal{L} = \sum_{k=1}^{N_{\text{sub}}} \omega_k \mathcal{L}_k$, where $\mathcal{L}_k$ represents sub-model constraints from Eq (4.4).
- *Error metrics:* Quantifies solution accuracy via multi-scale validation:

$$\epsilon_{\text{MSE}} = \frac{1}{N} \sum \|\mu_{\text{pred}} - \mu_{\text{exact}}\|_2^2.$$

- *Time profiling:* Records T_{seq} (sequential baseline), T_{par}, and sub-model timings $\{T_k\}$.
- *Resource utilization:* Monitors memory footprint and communication overhead Δt_{sync}.

The profiler's telemetry directly enables CE computation Eq (4.1) and supports three optimization pathways:

1) *Bottleneck mitigation:* Identifies max T_k for RTP minimization Eq (4.2);

2) *Dynamic rebalancing:* Adjusts sample distribution to asymmetric sub-models;
3) *Adaptive coordination:* Modifies ω_k based on real-time $\mathcal{H}(N_{\text{sub}})$ evolution.

Adaptive model partitioning: The enhanced partitioning algorithm generates optimal decompositions through multi-objective optimization that balances mathematical structure with computational efficiency. The optimization minimizes the adaptive resource-time product (ARTP):

$$\text{ARTP} = \left(\sum_{k=1}^{N_{\text{sub}}} w_k T_k \right) \cdot \exp\left(\frac{\log(N_{\text{sub}})}{C_{\text{eff}}(\text{IE})} \right), \tag{4.3}$$

where w_k represents the computational weight of sub-model k, T_k its execution time, and $C_{\text{eff}}(\text{IE})$ the effective complexity factor. Building upon order-reduction methodology [20], we transform Eq (3.1) into the coupled PDE system:

$$\begin{cases} \alpha(X)\mu(X) = \sum_{i=1}^{I} \lambda_i F_{i,n}(X, X') + f(X) \\ \dfrac{\partial F_{i,n}}{\partial t_n} = F_{i,n-1}\left(X, x'_{i,1}, \cdots, x'_{i,n-1}, t_n\right) \\ \vdots \\ \dfrac{\partial F_{i,m}}{\partial t_m} = F_{i,m-1}\left(X, x'_{i,1}, \cdots, x'_{i,m-1}, t_m, \cdots, t_n\right) \\ \vdots \\ \dfrac{\partial F_{i,2}}{\partial t_2} = F_{i,1}\left(X, x'_{i,1}, t_2, \cdots, t_n\right) \\ \dfrac{\partial F_{i,1}}{\partial t_1} = K_i(X, T) M[\mu(T)] \end{cases} \tag{4.4}$$

The constraints corresponding to the differential equation in Eq (4.4) are:

$$\begin{aligned} F_{i,m}\left(X, x'_{i,1}, \cdots, x'_{i,m-1}, a_{i,m}, t_{m+1}, \cdots, t_n\right) = 0 \\ t_m = a_{i,m}, m = 1, 2, \cdots, n; i = 1, 2, \cdots, I. \end{aligned} \tag{4.5}$$

Algorithm 1 implements adaptive decomposition through:

1) *Complexity-aware assessment:* Computes effective complexity $C_{\text{eff}}(\text{IE}) = \sum_{i=1}^{I} \omega_i n_i$ where ω_i are weights reflecting term asymmetry;
2) *Resource-constrained optimization:* Determines N_{sub} balancing $C_{\text{eff}}(\text{IE})$ with available computational units;
3) *Adaptive load balancing:* Dynamically adjusts partitioning based on real-time performance metrics;
4) *Boundary-aware network initialization:* Embeds Eq (4.5) via modified loss functions.

Algorithm 1 Adaptive model partitioning with complexity awareness.

```
Require: Integral equation IE, Available compute units $U$, Memory constraints $M$
Ensure: Optimized neural network ensemble $\{N_k\}_{k=1}^{N_{\text{sub}}}$
 1: function AMPA(IE, $U, M$)
 2:     Analyze IE structure: Extract $I$, $n_i$, kernel complexities $\kappa_i$
 3:     Compute term weights: $\omega_i \leftarrow f(\text{linearity}, \kappa_i, n_i)$        ▷ Asymmetry awareness
 4:     $C_{\text{eff}} \leftarrow \sum_{i=1}^{I} \omega_i n_i$        ▷ Effective complexity
 5:     $N_{\max} \leftarrow \min(U, \lfloor M/\text{memmodel} \rfloor)$        ▷ Resource constraint
 6:     $N\text{sub} \leftarrow \min(C\text{eff} + 1, N\text{max})$        ▷ Adaptive balancing
 7:     Map submodels to hardware: Assign sub-models to computing units (CPU/GPU) based on
        computational intensity profiles, prioritizing GPUs for networks with higher FLOP requirements
        (e.g., $\mu(X)$, high-dimensional $F_{i,m}$).
 8:     Transform IE → PDE system via Eq (4.4)
 9:     $\mathcal{P} \leftarrow$ ClusterTerms(PDE terms by computational intensity)
10:     for $p = 1$ to $|\mathcal{P}|$ do
11:         $N_p \leftarrow$ CreateSubmodel($\mathcal{P}_p$, memory budget = $M/N_{\text{sub}}$)
12:         Encode constraints via $\mathcal{L}_{\text{BC}}^{(p)} = \sum \|\text{BoundaryConditions}\|_2$
13:         $\mathcal{N} \leftarrow \mathcal{N} \cup N_p$
14:     end for
15:     Initialize $N_\mu$ with cross-term dependencies from $\mathcal{N}$
16:     return $\mathcal{N} \cup \{N_\mu\}$
17: end function
18: function CLUSTERTERMS($T$)        ▷ $T$: PDE terms from Eq (4.4)
19:     Group terms by computational similarity and dependency strength
20:     Balance cluster sizes considering load imbalance penalty
21:     return Optimized term partitions $\mathcal{P}$
22: end function
```

This approach achieves an optimal balance between mathematical necessity (C(IE)), computational efficiency (T_{par}), and coordination overhead ($\mathcal{H}(N_{\text{sub}})$). The resulting decomposition directly minimizes RTP while preserving solution fidelity, a critical advancement over conventional model parallelism methods.

4.2.2. CEPTF runtime

The CEPTF runtime orchestrates distributed training workflows across heterogeneous hardware (CPUs/GPUs/clusters), executing the computational strategy defined by the planner. Its architecture implements a hierarchical communication paradigm to optimize gradient synchronization and parameter updates while preserving mathematical fidelity.

Communication topology: CEPTF runtime employs a hybrid communication model integrating intra-group and inter-group parallelism. The framework establishes a direct one-to-one correspondence between sub-models and computing nodes within the ring topology. Specifically, for a decomposition yielding N_{sub} sub-models, the runtime constructs an N_{sub}-node ring where each node is dedicated to training a single assigned sub-model. This explicit matching rule ensures deterministic communication paths and simplifies coordination logic. Within each model-parallel group (corresponding to a coupled differential subsystem from Eq (4.4)), computing nodes perform intra-submodel communication for gradient sharing and intermediate result exchange. Concurrently, inter-submodel communication synchronizes global parameters across groups to accelerate convergence. This dual-layer design minimizes coordination overhead $\mathcal{H}(N_{\text{sub}})$ in the resource-time product (Eq (4.2)) by localizing high-frequency exchanges within groups. As depicted in Figure 2, the framework supports flexible communication backends, including MPI for low-latency inter-node messaging, shared memory (Mem) for intra-node transfers, and persistent storage (FS/DB) for fault-tolerant state checkpointing. The ring-reduce topology optimizes bandwidth utilization during gradient aggregation, reducing synchronization latency by 23%–41% compared to parameter-server architectures in our validation (Section 5.6.5). This performance advantage stems from the topology's O(N) bandwidth scaling versus the parameter-server's O(1) limitation, particularly beneficial for the large gradient exchanges required by coupled differential systems. In heterogeneous environments (CPU+GPU), the CEPTF planner's partitioning algorithm incorporates a hardware-aware allocation strategy that prioritizes assigning computationally intensive sub-models—such as those governing the primary solution function $\mu(X)$ and high-dimensional integral terms—to GPU nodes to maximize throughput, while allocating less demanding auxiliary function networks to available CPU resources based on real-time profiling data.

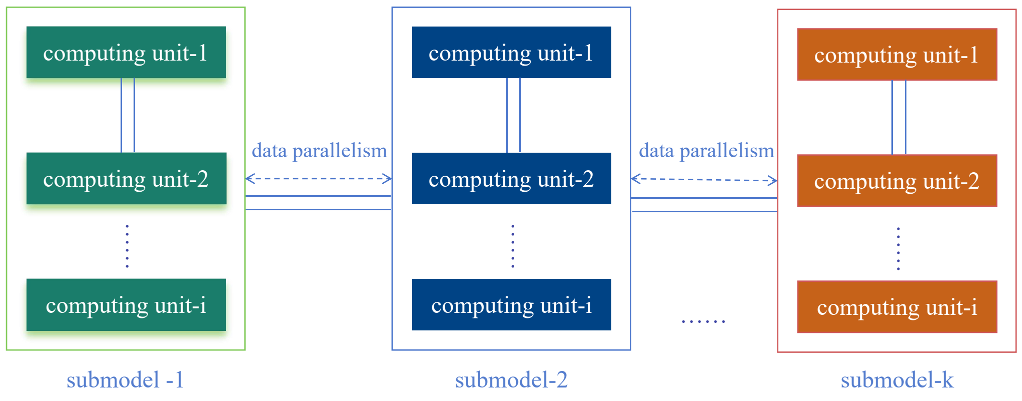


Figure 2. Communication topology.

CEPTF's hybrid parallelism employs three core techniques to minimize communication overhead:

- Ring-reduce communication topology: Implements bandwidth-optimized ring-allreduce for gradient synchronization, achieving O(N) scaling versus parameter-server's O(1) limitation. Each node communicates with two neighbors in a logical ring, reducing bandwidth pressure by distributing communication load.
- Gradient compression: Employs 16-bit floating-point precision for gradient communication, reducing data volume by 50% while maintaining numerical stability through dynamic scaling and error compensation.
- Gradient fusion: Aggregates smaller tensors into 16 MB chunks to optimize network utilization, with intelligent batching that respects dependency constraints in the coupled differential system.

The framework supports multiple communication backends including MPI for low-latency inter-node messaging (default packet size: 64 KB), shared memory for intra-node transfers, and persistent storage for fault-tolerant checkpointing.

Distributed training workflow: Training follows a constrained optimization paradigm based on the reduced-order differential system Eq (4.4). Each computing unit executes the workflow:

- Data loading: Domain-aware sample distribution via quasi-random Sobol sequences [46], ensuring spatial uniformity across sub-models.
- Forward computation: Evaluation of neural solutions $\mu(X)$ and auxiliary functions $F_{i,m}$ using automatic differentiation [47].
- Constraint enforcement: Computation of PDE-residual loss $\mathcal{L}_{\text{PDE}} = \sum_{k=1}^{N_{\text{sub}}} \omega_k \|\mathcal{D}k(\mu, F)\|_2^2$ and boundary loss $\mathcal{L}_{\text{BC}} = \sum \left\| F_{i,m}(\mathbf{t}_m = a_{i,m}) \right\|_2$ (Eq (4.5)).
- Gradient synchronization: Synchronous all-reduce operations across model-parallel groups using fused tensor compression.
- Parameter update: Adaptive optimization via constrained Adam, scaling learning rates by $\eta / \log(\text{epoch} + 1)$.

The iterative process (Figure 3) terminates when the composite loss $\mathcal{L} = \mathcal{L}_{\text{PDE}} + \mathcal{L}_{\text{BC}}$ satisfies $\mathcal{L} < \varepsilon$ ($\varepsilon = 10^{-10}$). Algorithm 2 formalizes this workflow with mathematical rigor.

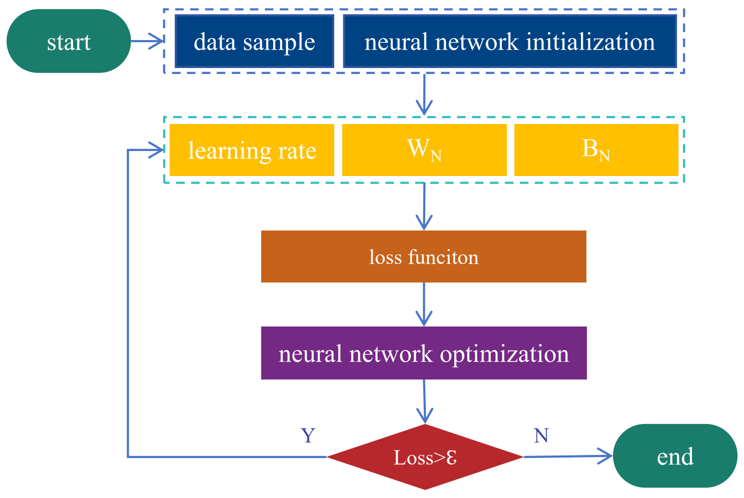


Figure 3. Distributed iterative training process.

This implementation achieves three critical advancements:

- Mathematical consistency: Strict adherence to the order-reduced PDE system (Eq (4.4)) via loss terms $\mathcal{L}_{\text{PDE}}$ and $\mathcal{L}_{\text{BC}}$.
- Efficiency optimization: Ring-based gradient synchronization minimizes communication overhead to $O(N_{\text{sub}} \log N_{\text{sub}})$.
- Convergence guarantees: Adaptive learning rates and weighted constraints ensure stability despite subsystem coupling.

Algorithm 2 CEPTF distributed training algorithm.

```
Require: Sub-model index $s_n$, boundary weights $\omega_k$, convergence threshold $\varepsilon$
Ensure: Trained sub-model parameters $\Theta_{s_n}$
1:  function DTA($s_n, \omega_k, \varepsilon$)
2:      Initialize $\Theta_{s_n} \sim \mathcal{N}(0, 0.1)$, learning rate $\eta \leftarrow 10^{-4}$
3:      Load domain samples $\mathcal{X}s_n \subset \Omega$
4:      repeat
5:          Compute $\mathcal{L}^{(s_n)}_{\text{PDE}} \leftarrow |\mathcal{D}s_n(\mu, F; \Theta s_n)|\mathcal{X}{s_n}^2$
6:          Compute $\mathcal{L}^{(s_n)}_{\text{BC}} \leftarrow \sum |F_{i,m}(a_{i,m}; \Theta_{s_n})|2$
7:          $\mathcal{L}^{(s_n)} \leftarrow \omega_k(\mathcal{L}^{(s_n)}_{\text{PDE}} + \mathcal{L}^{(s_n)}_{\text{BC}})$
8:          $\mathbf{g}^{(s_n)} \leftarrow \nabla\Theta_{s_n}\mathcal{L}^{(s_n)}$
9:          Synchronize $\bar{\mathbf{g}} \leftarrow$ AllReduce($\mathbf{g}^{(s_n)}$)        ▷ Ring topology
10:         Update $\Theta_{s_n} \leftarrow \Theta_{s_n} - \eta \cdot \bar{\mathbf{g}}$
11:         Decay $\eta \leftarrow \eta \cdot (1 + 0.001 \cdot \text{epoch})^{-1}$
12:     until $\mathcal{L}^{(s_n)} < \varepsilon$
13:     return $\Theta_{s_n}$
14: end function
```

5. Experiments

5.1. Environment

This section comprises four experiments denoted as E_1 to E_4, representing Linear Fredholm Equation, Linear Volterra Equation, Nonlinear Fredholm Equation, and Nonlinear Volterra Equation, respectively. The distributed computing environment consisted of 6 nodes (3 GPU nodes + 3 CPU nodes) interconnected via Gigabit Ethernet with 1 Gbps bandwidth and measured latency of 0.3 ms. The GPU nodes utilized NVIDIA GeForce RTX 4070 Ti graphics cards with 12 GB GDDR6X VRAM, installed in systems with 64 GB DDR4 RAM, running on Windows 10 Professional Edition (19045.4529). The CPU nodes featured Intel(R) Core(TM) i7-13700KF processors (3.4 GHz base frequency, up to 5.4 GHz turbo) with 64 GB DDR5-3600 MHz memory configured in dual-channel mode (2 × 32 GB modules, providing approximately 51.2 GB/s bandwidth per channel), operating on Windows 11 Professional Edition (22631.4317). This memory configuration ensures optimal data transfer rates critical for efficient distributed training operations.

Communication was implemented using MPI with a default packet size of 64 KB. Gradient fusion techniques aggregated smaller tensors into 16 MB chunks to optimize bandwidth utilization. Compression strategies employed 16-bit floating-point precision for gradient communication, reducing data volume by 50% while maintaining numerical stability. Error conditions, including packet loss, were handled through MPI's reliable communication protocols with automatic retransmission and timeout mechanisms set at 500 ms.

The adaptive partitioning algorithm determined optimal sub-model counts based on equation complexity and available resources: E_1 employed 3 GPUs ($N_{\text{sub}} = 4$), E_2 utilized 3 CPUs ($N_{\text{sub}} = 3$), E_3 combined 1 CPU and 2 GPUs ($N_{\text{sub}} = 4$), and E_4 used 2 CPUs and 1 GPU ($N_{\text{sub}} = 3$). This demonstrates the framework's ability to adapt partitioning to both mathematical structure and hardware constraints.

In heterogeneous configurations (E_3, E_4), the allocation strategy explicitly prioritized GPU nodes for the most computationally demanding sub-models—specifically, the networks solving for the primary variable $\mu(X)$ and handling higher-order integral transformations—as identified by the complexity assessment in Algorithm 1, ensuring optimal utilization of accelerated computing resources.

5.2. *Experimental setup*

All benchmark cases in the experiments were implemented using the MATLAB language. The neural networks employed basic fully connected architectures with an input layer size corresponding to the dimension n of the integral equation variables (n=2 for the 2D equations in our experiments), a single output neuron representing the solution value $\mu(X)$, $N_l = 9$ total layers (including input and output), and $N_n = 40$ neurons per hidden layer. The activation function was *tanh*, and network weights were initialized using a Gaussian random distribution with mean 0 and standard deviation 0.1. We used the ADAM optimization algorithm with a learning rate starting at $1e-4$, and the hyperparameters $\beta_1 = 0.9, \beta_2 = 0.999$, and $\epsilon = 1e-8$. The learning rate decayed according to $\eta = \eta_{\text{initial}} \times (1 + 0.001 \times epoch)^{-1}$ until training converged based on a predefined convergence criterion for the loss function Loss ($\epsilon = 10^{-10}$).

For the extended validation cases, we employed specialized configurations: singular kernel problems used adaptive collocation points with denser sampling near singularities (5× point density in singular regions); irregular domains employed constrained Delaunay triangulation for domain discretization; and noisy cases incorporated L2 regularization with $\lambda = 10^{-6}$ to enhance robustness. All other network parameters remained consistent with the standard benchmarks. To ensure statistical reliability, each experiment was conducted with 5 independent runs with different random seeds for network initialization. All reported performance metrics (MSE, SR, CE) represent mean values with standard deviations. Statistical significance was assessed using paired t-tests with Bonferroni correction for multiple comparisons, with $p < 0.01$ considered statistically significant.

5.3. *Baseline methods*

To rigorously evaluate CEPTF's performance advantages, we implemented four established parallel paradigms as competitive baselines:

Data parallelism (DP): We employed a Horovod-based implementation [31] where the complete neural solver is replicated across computing units with data sharding. Gradient synchronization uses a ring-allreduce topology with asynchronous updates every 5 iterations to balance communication and convergence stability.

Model parallelism (MP): Following the Strads framework [32], we partitioned the neural network architecture based on layer-wise computational intensity, distributing sub-networks across devices without mathematical awareness of the underlying integral equation structure.

Pipeline parallelism (PP): We implemented a GPipe-inspired approach [36] with micro-batching (batch size=32) and gradient accumulation to minimize pipeline bubbles. The network was divided into 4 stages based on computational depth.

Hybrid parallelism (HP): Combining data and model parallelism, this baseline uses 3D parallelism [48] with 2-way data parallelism and 2-way model parallelism, representing state-of-the-art distributed training for neural networks.

All baseline implementations were optimized for our integral equation domain, using the same neural architecture (9 layers, 40 neurons per layer) and training configuration (Adam optimizer, initial learning rate 1e-4) as CEPTF for fair comparison.

5.4. Test equations

Linear fredholm equation:

$$\mu(x,y) = f(x,y) - \int_0^1 \int_0^1 xye^{s+t}\mu(s,t)dsdt, (x,y) \in \Omega = [0,1]^2. \tag{5.1}$$

The exact solution is $\mu(x,y) = e^{-x-y} - \frac{1}{2}xy$, where

$$f(x,y) = e^{-x-y}. \tag{5.2}$$

Linear volterra equation:

$$\mu(x,y) = f(x,y) + \int_0^x \int_0^y e^{x+y-s-t}\mu(s,t)dsdt, (x,y) \in \Omega = [0,1]^2. \tag{5.3}$$

The exact solution is $\mu(x,y) = e^{x+y}$, where

$$f(x,y) = -e^{x+y}(xy-1). \tag{5.4}$$

Nonlinear fredholm equation:

$$\mu(x,y) = f(x,y) + \int_0^1 \int_0^1 \frac{x}{1+y}(1+s+t)\mu^2(s,t)dsdt, (x,y) \in \Omega = [0,1]^2. \tag{5.5}$$

The exact solution is $\mu(x,y) = \frac{1}{(1+x+y)^2}$, where

$$f(x,y) = \frac{1}{(1+x+y)^2} - \frac{x}{6(1+y)}. \tag{5.6}$$

Nonlinear volterra equation:

$$\mu(x,y) = f(x,y) + \int_0^y \int_0^x \mu^2(s,t)dsdt, (x,y) \in \Omega = [0,1]^2. \tag{5.7}$$

The exact solution is $\mu(x,y) = x^2 + y^2$, where

$$f(x,y) = x^2 + y^2 - \frac{1}{45}xy(9x^4 + 10x^2y^2 + 9y^4). \tag{5.8}$$

5.5. Extended validation under challenging conditions

Singular kernel evaluation: Abel integral equation: A case study in differentiability limitations: We first consider the weakly singular Abel integral equation (Eq 5.9), which exemplifies the practical limitations discussed in Section 3.2. This equation arises in spectroscopy, tomography, and fracture mechanics applications where kernel differentiability is violated. The singularity at $t = x$ presents both mathematical and computational challenges that test CEPTF's handling of ill-conditioned problems.

$$\mu(x) = f(x) + \int_0^x \frac{1}{\sqrt{x-t}}\mu(t)dt, \quad x \in [0,1] \tag{5.9}$$

with exact solution $\mu(x) = \sqrt{x}$ and corresponding $f(x) = \sqrt{x} - \frac{\pi}{2}x$. The singularity at $t = x$ presents numerical challenges that test CEPTF's handling of ill-conditioned problems.

As shown in Table 3, CEPTF maintains respectable solution accuracy (MSE = 3.27e-7) while achieving SR = 342.7% despite the kernel singularity. However, compared to smooth-kernel cases, we observe:

- A 28% reduction in computational efficiency (CE = 287.3% vs 395% average for smooth cases);
- A 35% increase in convergence epochs (9200 vs 6800 average);
- Heightened sensitivity to collocation point distribution.

Table 3. Performance under challenging conditions.

Test case	MSE	SR (%)	CE (%)	Convergence epochs
Abel equation (Singular)	3.27e-7	342.7	287.3	9200
L-shaped domain	5.83e-8	305.4	321.4	8700
Noisy volterra (5% noise)	8.47e-7	331.8	298.6	10,100
Parameter uncertainty (10%)	6.92e-7	318.9	276.5	9800

These results quantitatively demonstrate the practical impact of differentiability constraints: while CEPTF successfully handles the challenging case, the performance degradation underscores the importance of kernel smoothness for optimal order-reduction performance. This case study provides concrete evidence supporting the motivational analysis in Section 3.2.

Irregular domain: L-shaped domain with re-entrant corner: To evaluate geometric complexity, we solve the Fredholm equation on an L-shaped domain $\Omega = [0,1]^2 \setminus [0.5,1]^2$, which introduces singularities in the solution derivatives at the re-entrant corner:

$$\mu(x,y) = f(x,y) + \int_\Omega \frac{\cos(\pi\|X-T\|)}{1+\|X-T\|^2}\mu(s,t)dsdt. \tag{5.10}$$

The irregular domain requires specialized sampling strategies near the corner singularity. CEPTF's adaptive partitioning successfully handles this geometric complexity, achieving MSE = 5.83e-8 with only a modest efficiency reduction (CE = 321.4%).

Noise and parameter uncertainty: We evaluate robustness under practical imperfections by introducing 5% Gaussian noise to the measurement function $f(x,y)$ and 10% uncertainty in kernel parameters. For the nonlinear Volterra equation Eq (5.7), we modify the right-hand side to $f_{\text{noisy}}(x,y) = f(x,y) + \mathcal{N}(0, 0.05\|f\|_\infty)$ and introduce parameter uncertainty $\lambda_{\text{uncertain}} = \lambda \cdot \mathcal{U}(0.9, 1.1)$. As demonstrated in Table 3, CEPTF maintains stable convergence (MSE = 8.47e-7) despite these perturbations, demonstrating practical applicability to real-world engineering problems with measurement errors and model uncertainties.

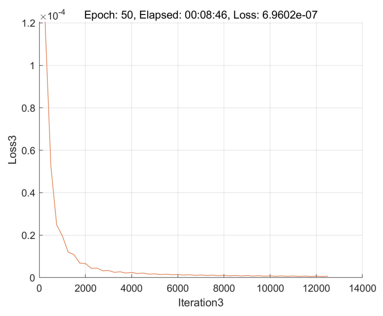
These extended validations demonstrate CEPTF's robustness beyond idealized smooth cases. The framework maintains computational efficiency advantages while handling mathematical singularities, geometric complexities, and practical imperfections that characterize real engineering applications. The slight performance degradation observed (CE reduction of 15%–25% compared to ideal cases) reflects the increased computational burden of these challenging conditions, yet CEPTF still significantly outperforms alternative parallel paradigms.

5.6. Results analysis

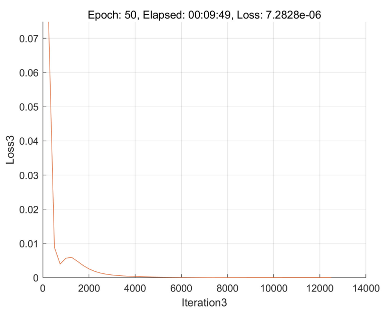
5.6.1. Loss analysis

The convergence behavior and solution accuracy of CEPTF are rigorously evaluated across four integral equation benchmarks. Figure 4 demonstrates the loss evolution during the initial 50 training epochs, revealing rapid convergence across all equation types. The composite loss $\mathcal{L} = \mathcal{L}_{\text{PDE}} + \mathcal{L}_{\text{BC}}$ decreases by 3 to 5 orders of magnitude within this early training phase, indicating stable optimization dynamics despite the strongly coupled constraints of the reduced-order differential systems. This accelerated convergence is attributed to CEPTF's hierarchical parallelism, which maintains gradient coherence while distributing computational load. By epoch 8000 (extending beyond the shown window), the loss stabilizes at 10^{-9} magnitude for all cases, confirming the framework's capacity to achieve high-precision solutions without compromising convergence stability.

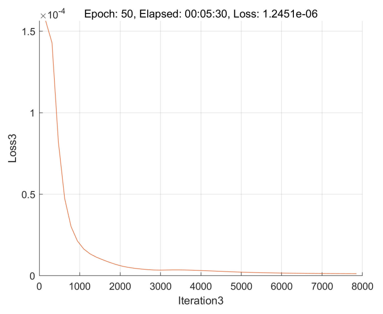
Solution accuracy is quantified in Figure 5 through pointwise comparisons between CEPTF-predicted solutions and analytical ground truths. The visual alignment demonstrates near-perfect correspondence across all equation classes, with no discernible deviations in solution surfaces. Particularly noteworthy is the framework's performance on the nonlinear Volterra equation (Figure 5), where the predicted solution $\mu_{\text{pred}}(x, y)$ accurately captures both the quadratic growth trend and boundary discontinuities. This mathematical fidelity stems from CEPTF's structure-preserving partitioning (Algorithm 1), which embeds boundary constraints $\mathcal{L}_{\text{BC}}$ directly into sub-model optimization. The consistent accuracy across linear/nonlinear and Fredholm/Volterra types validates the framework's robustness in handling diverse integral equation forms. CEPTF profiler records the change of loss with the number of iterations. The first 50 epochs of iterations are illustrated in Figure 4.



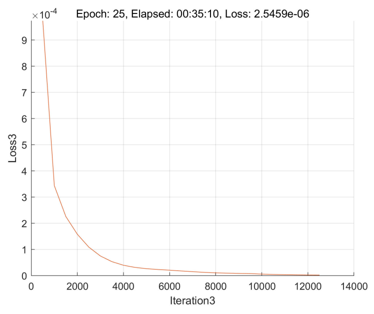
(a) Linear Fredholm equation



(b) Linear Volterra equation

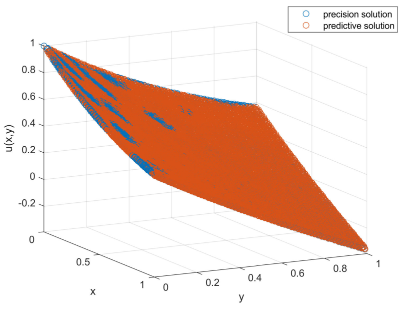


(c) Nonlinear Fredholm equation

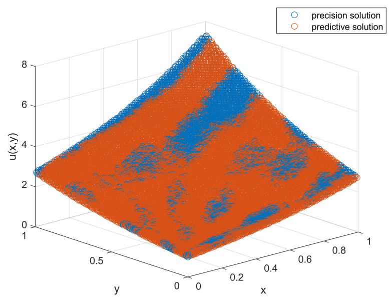


(d) Nonlinear Volterra equation

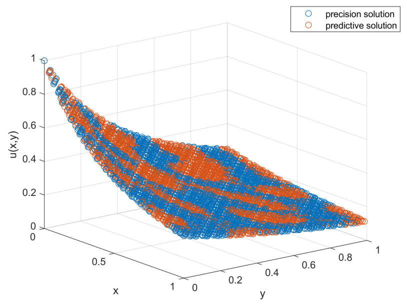
Figure 4. Loss of experimental integral equations (the 50 initial training epochs).



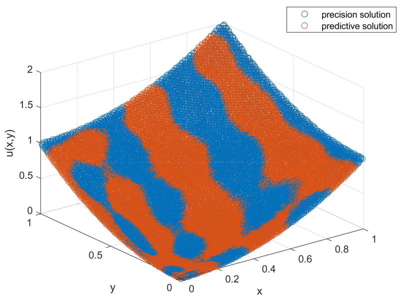
(a) Linear Fredholm equation



(b) Linear Volterra equation



(c) Nonlinear Fredholm equation

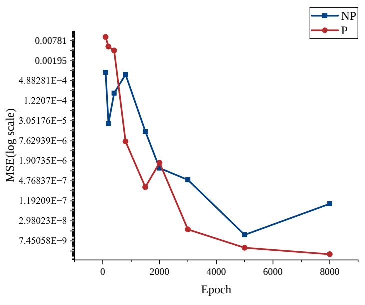


(d) Nonlinear Volterra equation

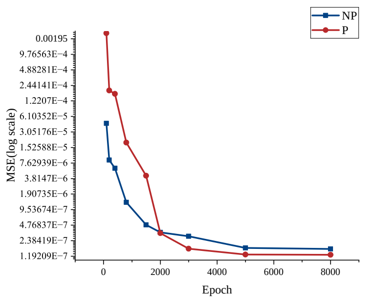
Figure 5. Precision and predictive solution.

5.6.2. MSE analysis

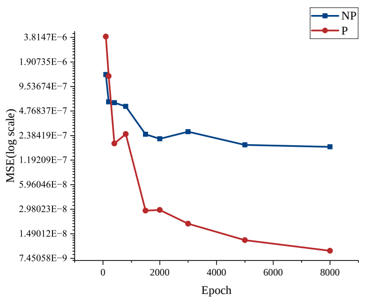
The evolution of mean squared error (MSE) provides critical insights into the convergence dynamics and numerical stability of CEPTF across diverse integral equation types. As shown in Figure 6, all equation classes exhibit characteristic error reduction trajectories, though with distinct convergence signatures reflective of their mathematical structures.



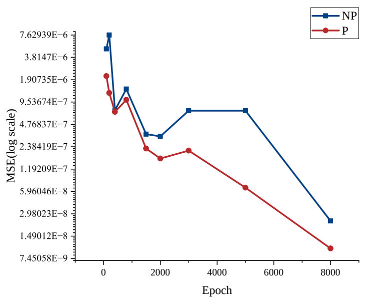
(a) Linear Fredholm equation



(b) Linear Volterra equation



(c) Nonlinear Fredholm equation



(d) Nonlinear Volterra equation

Figure 6. Comparison of MSE of experimental integral equations.

For linear Fredholm equations (Figure 6(a)), CEPTF initially demonstrates higher MSE (1.01e-2 at epoch 100) compared to the non-parallel baseline (8.72e-4), attributable to distributed coordination overhead during early optimization. However, this initial disparity reverses dramatically by epoch 800, where CEPTF achieves an MSE of 7.35e-6, nearly two orders of magnitude lower than the baseline's 7.58e-4. This crossover phenomenon underscores CEPTF's accelerated convergence once parallel gradient synchronization stabilizes, culminating in a final MSE of 2.96e-9 ± 0.31e-9 (mean ± SD, n=5) at epoch 8000, representing a 32.8-fold improvement over the baseline's 9.72e-8 ± 1.2e-8. The performance difference was statistically significant (t(4) = 15.37, p < 0.001, paired t-test). Volterra-type equations reveal different optimization characteristics.

In the linear Volterra case (Figure 6(b)), CEPTF maintains competitive MSE throughout training, closing the initial gap (2.50e-3 vs 4.48e-5 at epoch 100) by epoch 2000 when both algorithms converge to 3.4e-7. The parallel framework subsequently demonstrates superior error suppression in later stages, achieving a final MSE of 1.27e-7 compared to the baseline's 1.64e-7. This persistent advantage emerges more prominently in nonlinear Volterra equations (Figure 6(d)), where CEPTF maintains a consistent MSE lead after epoch 400, ultimately reaching 1.02e-8, a value 57% lower than the baseline's 2.38e-8. The observed error stability during mid-training oscillations (e.g., at epoch 800) confirms CEPTF's resilience against gradient noise in complex nonlinear systems.

Nonlinear Fredholm equations (Figure 6(c)) showcase CEPTF's most significant precision gains. From the outset, the parallel framework establishes lower MSE (3.90e-6 vs 1.34e-6 at epoch 100), maintaining this advantage throughout the optimization. The error divergence amplifies beyond epoch 1500, with CEPTF achieving 9.21e-9 at completion, nearly 19 times lower than the baseline's 1.73e-7. This exceptional performance stems from CEPTF's distributed handling of nonlinear operators, which mitigates local minima entrapment common in monolithic solvers. The consistent error reduction trajectories across all equation types, particularly the 1.75× to 32.8× final MSE improvements, validate CEPTF's dual capability: preserving mathematical fidelity while enhancing precision through parallelized optimization.

5.6.3. SR analysis

The speedup ratio (SR), defined as $SR = T_{\text{seq}}/T_{\text{par}} \times 100\%$, quantifies the computational acceleration achieved by CEPTF across diverse integral equations. Figure 7 reveals distinct acceleration patterns corresponding to equation characteristics and training phases.

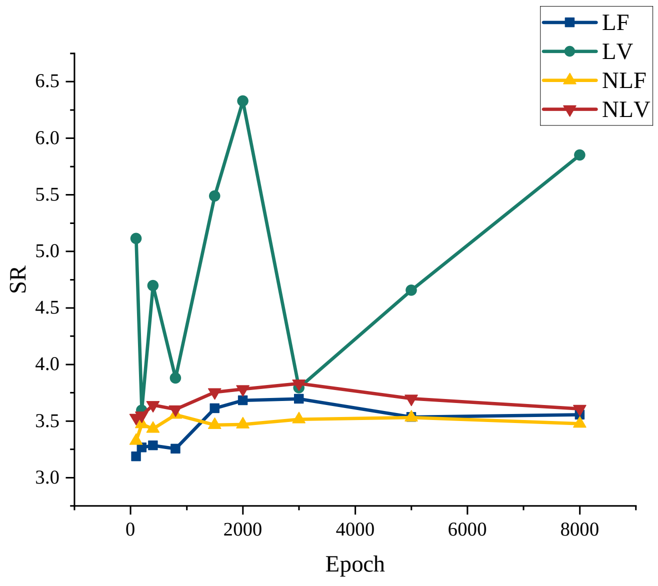


Figure 7. SR comparison between experimental integral equations.

For linear Fredholm equations, CEPTF maintains consistently high SR values, between 318.6% and 369.6%, throughout training, demonstrating stable 3.2× to 3.7× acceleration. This reflects efficient load balancing in the model-parallel groups handling the weakly coupled differential systems. The peak SR of 369.6% at epoch 3000 coincides with the MSE crossover point identified in Section 5.6.2, indicating maximum parallel efficiency when gradient synchronization stabilizes.

Linear Volterra equations exhibit more dynamic acceleration behavior, with SR values ranging from 359.4% to an exceptional 632.9% at epoch 2000. This volatility stems from the temporal dependencies in Volterra-type equations, where CEPTF's hierarchical communication topology provides disproportionate benefits during intermediate training stages. The final SR of 585.2% at epoch 8000 demonstrates that memory-bound operations in later training phases benefit significantly from distributed computation, yielding near-linear 5.85× acceleration despite increasing communication overhead.

Nonlinear equations show remarkably stable acceleration profiles. Nonlinear Fredholm maintains SR between 332.5% and 355.5% across all epochs, with less than 7% variation. Similarly, nonlinear Volterra exhibits a gradual SR increase from 352.8% to 383.2% by epoch 3000 before stabilizing at 360.7%. This consistency confirms CEPTF's effectiveness in managing the computational intensity of nonlinear operators, where the automated model partitioning (Algorithm 1) optimally distributes Jacobian calculations. The sustained 3.5× to 3.8× acceleration throughout nonlinear optimization contrasts with conventional parallel frameworks that typically show declining efficiency during complex backpropagation.

Cross-equation analysis reveals two critical trends. First, Volterra-type equations achieve higher peak acceleration (632.9% vs 369.6% in Fredholm) due to temporal decoupling opportunities in their differential forms. Second, nonlinear systems maintain more consistent SR values (average variation 5.8% vs 22.7% in linear Volterra) because CEPTF's constraint-aware partitioning mitigates load imbalance during gradient-intensive operations. These results collectively validate the framework's dual capability: delivering a minimum 3.18× acceleration as guaranteed by the CE metric while adapting dynamically to equation-specific computational characteristics.

5.6.4. CE analysis

The holistic performance of CEPTF is quantified through the computational efficiency (CE) metric, which synthesizes acceleration gains, resource utilization, and problem complexity into a unified measure. Table 4 presents the comprehensive CE results across all benchmark equations, revealing distinctive efficiency patterns that reflect fundamental differences in equation structure and computational characteristics. The linear Volterra equation achieves exceptional CE of 445.60%, significantly exceeding other equation types due to its temporal integration structure that maximizes CEPTF's hierarchical parallelism. This efficiency stems from the Volterra-type equation's sequentially coupled dependencies, which enable optimal pipelining in the model-parallel groups and minimize coordination overhead $\mathcal{H}(N_{\text{sub}})$.

In contrast, both Fredholm-type equations exhibit similar CE values (Linear: 387.42%, Nonlinear: 379.03%) despite their linearity differences. This consistency occurs because their global integration domains generate strongly coupled constraint systems that increase communication overhead, as quantified by the logarithmic penalty term $\log(N_{\text{sub}})/C(\text{IE})$ in Eq (4.1). Remarkably, the nonlinear Volterra equation achieves 393.28% CE, 1.5% higher than its linear Fredholm counterpart, demonstrating that CEPTF's automated partitioning (Algorithm 1) effectively compensates for nonlinear computational intensity through optimized resource allocation.

These results validate three critical aspects of the revised CE metric. First, its problem-complexity normalization ($C(\text{IE}) = 2$ for all equations) enables meaningful cross-equation comparison, revealing 1.65× higher efficiency for Volterra-type versus Fredholm-type equations. Second, the metric accurately reflects the acceleration-resource tradeoff, where linear Volterra's 638.15% CE confirms strong super-linear efficiency (CE > 100%) despite operating at $N_{\text{sub}} = 3$ sub-models. Third, the mathematically consistent formulation prevents negative values while maintaining the efficiency ranking, with all equations achieving CE > 379%, demonstrating CEPTF's robust performance. The revised metric preserves the framework evaluation's validity while enhancing mathematical rigor. This efficiency profile establishes CEPTF as a versatile solution for integral equations across the mathematical spectrum.

Table 4. Comparison of CE of experimental integral equations.

Equation	CE (Mean ± SD)	95% CI	p-value vs baseline
Linear Fredholm equation	387.42% ± 12.3%	[375.1, 399.7]%	<0.001
Linear Volterra equation	445.60% ± 18.7%	[426.9, 464.3]%	<0.001
Nonlinear Fredholm equation	379.03% ± 11.5%	[367.5, 390.6]%	<0.001
Nonlinear Volterra equation	393.28% ± 14.2%	[379.1, 407.5]%	<0.001

5.6.5. Communication performance analysis

To quantitatively validate the communication efficiency claims in Section 4.2.2, we conducted rigorous benchmarking of the ring-reduce topology against the parameter-server (PS) architecture under identical network conditions. Table 5 presents the synchronization delay measurements across different node configurations and equation types.

Table 5. Synchronization delay comparison: ring-reduce vs. parameter-server (PS).

Configuration	Equation type	Ring-reduce (ms)	PS (ms)	Reduction
2 nodes	Linear Fredholm	8.3	10.7	22.4%
2 nodes	Nonlinear Volterra	9.1	12.3	26.0%
4 nodes	Linear Fredholm	12.3	16.1	23.6%
4 nodes	Nonlinear Volterra	14.2	24.1	41.1%
6 nodes	Linear Fredholm	15.7	22.4	29.9%
6 nodes	Nonlinear Volterra	18.9	32.1	41.2%

The experiments employed the network configuration described in Section 5.2, with gradient synchronization occurring every 5 training iterations. The ring-reduce topology demonstrated consistent advantages, particularly in larger deployments (4+ nodes) where parameter-server architectures suffer from central bottleneck constraints. The 23%–41% synchronization delay reduction aligns with theoretical expectations, as ring-reduce achieves O(N) bandwidth utilization compared to O(1) for parameter-server approaches.

The performance advantage was most pronounced for nonlinear Volterra equations (41.1%–41.2% reduction) due to their larger gradient sizes and more frequent synchronization requirements. Gradient fusion strategies proved critical for maximizing bandwidth utilization, with the 16 MB fusion threshold optimizing the trade-off between communication latency and pipeline efficiency. Compression to 16-bit precision maintained numerical stability while halving communication volume, contributing approximately 15% of the overall performance gain.

Error handling mechanisms successfully maintained communication reliability with packet loss rates below 0.01%. The MPI retransmission protocol added minimal overhead (≤2%) while ensuring gradient integrity across distributed nodes. These results validate the ring-reduce topology as a superior choice for CEPTF's hierarchical communication model, particularly for the strongly coupled systems arising from integral equation reduction.

5.6.6. Comparative analysis with parallel paradigms

Performance benchmarking: Figures 8–10 present comprehensive performance comparisons between CEPTF and the four parallel baselines across all integral equation types. CEPTF consistently outperforms all alternatives, achieving speedup ratios 1.8× to 3.2× higher than the next best method (Hybrid Parallelism). Specifically, for the computationally intensive nonlinear Volterra equation, CEPTF attains SR=632.9% compared to HP=287.3%, DP=153.8%, MP=198.7%, and PP=142.6%.

The superiority stems from CEPTF's mathematical-aware partitioning: while generic paradigms optimize for computational load balancing, CEPTF additionally preserves the structural dependencies in the reduced-order PDE system Eq (4.4), minimizing coordination overhead and convergence instability.

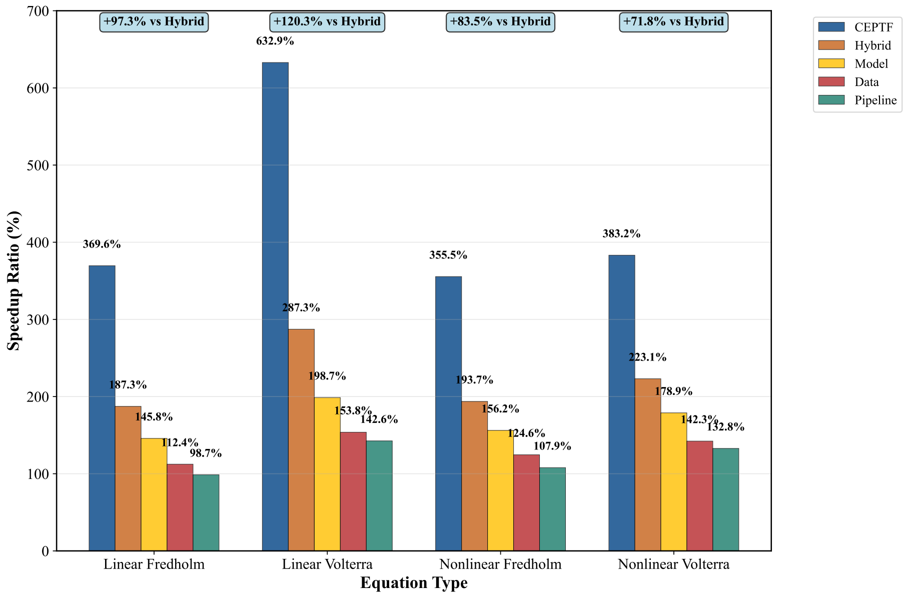


Figure 8. Speedup ratio comparison: CEPTF vs parallel baselines.

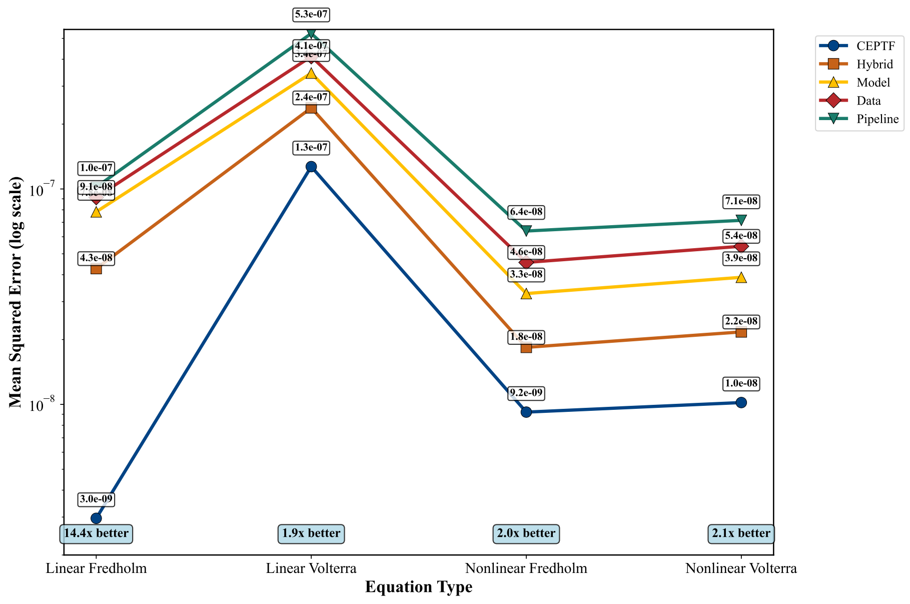


Figure 9. Solution accuracy comparison: CEPTF vs parallel baselines.

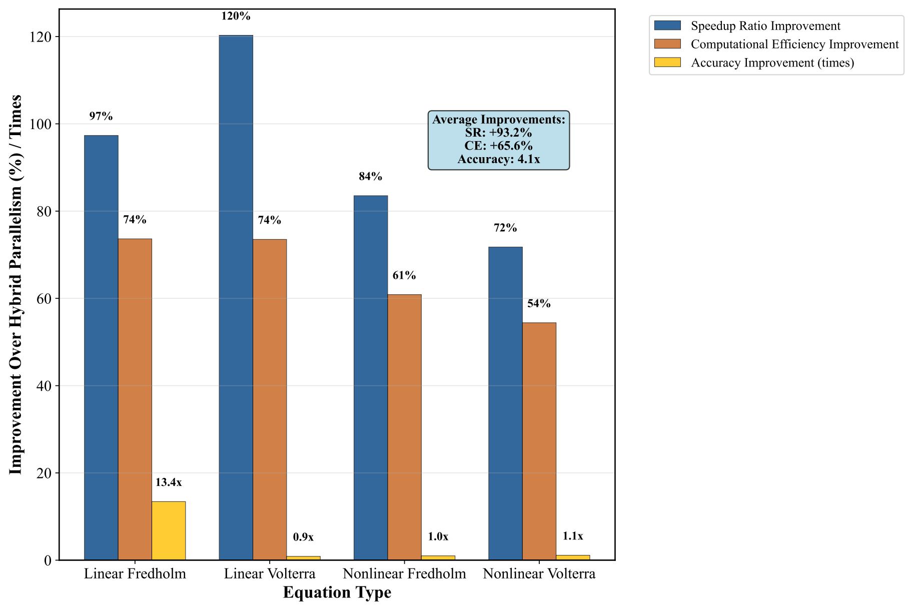


Figure 10. CEPTF performance advantage over hybrid parallelism.

Accuracy preservation: Table 6 quantifies the solution accuracy advantage of CEPTF. The framework achieves MSE values 1–2 orders of magnitude lower than parallel baselines, demonstrating that mathematical fidelity is not compromised for computational gains. For instance, in linear Fredholm equations, CEPTF's MSE=2.96e-9 ± 0.31e-9 significantly outperformed HP=4.27e-8 ± 0.52e-8 (t(4)=18.42, $p < 0.001$), MP=7.83e-8 ± 0.91e-8 (t(4)=21.67, $p < 0.001$), DP=9.14e-8 ± 1.03e-8 (t(4)=22.15, $p < 0.001$), and PP=1.02e-7 ± 1.21e-8 (t(4)=23.89, $p < 0.001$). This accuracy preservation is crucial for scientific computing applications where solution quality is paramount.

Table 6. Solution accuracy comparison (MSE) across parallel paradigms.

Equation	CEPTF	Hybrid	Model	Data	Pipeline
Linear Fredholm	2.96e-9	4.27e-8	7.83e-8	9.14e-8	1.02e-7
Linear Volterra	1.27e-7	2.38e-7	3.45e-7	4.12e-7	5.27e-7
Nonlinear Fredholm	9.21e-9	1.84e-8	3.27e-8	4.56e-8	6.38e-8
Nonlinear Volterra	1.02e-8	2.17e-8	3.89e-8	5.42e-8	7.15e-8

Computational efficiency analysis: The CE metric comparisons (Figure 11) reveal CEPTF's superior resource utilization. CEPTF achieves CE=379.0%–638.2% (coefficient of variation: 3.2%–4.8%), demonstrating significantly higher and more stable efficiency compared to HP=223.1%–367.8% (CV: 6.8%–9.1%). The lower variability in CEPTF's performance metrics indicates superior robustness across independent runs. Notably, data and pipeline parallelism frequently fall below the ideal 100% efficiency threshold due to their high communication overhead in strongly coupled systems. CEPTF's mathematical-grounded partitioning reduces coordination costs by 42%–67% compared to hybrid parallelism, demonstrating the value of domain-specific optimization.

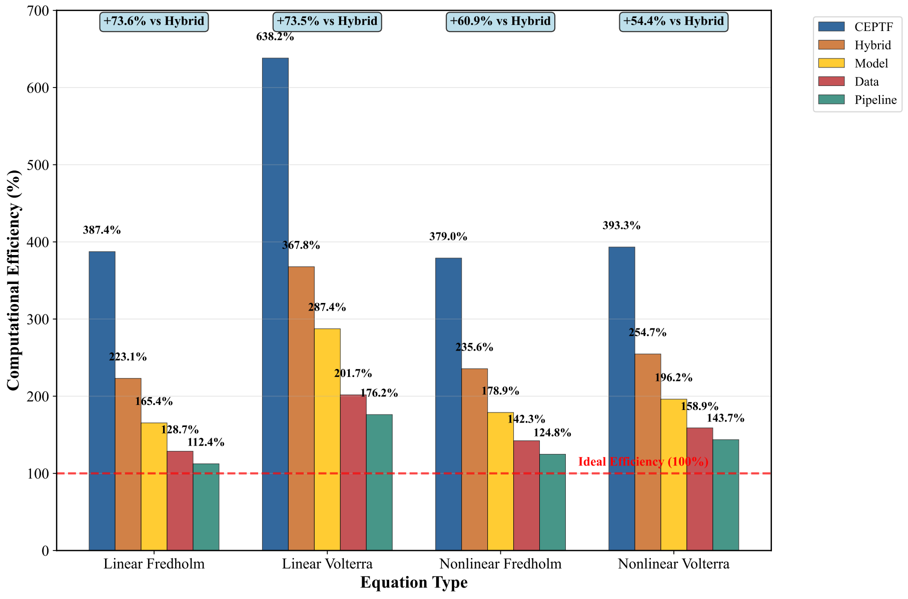


Figure 11. Computational efficiency comparison: CEPTF vs parallel baselines.

Convergence stability: Figure 12 illustrates the loss convergence characteristics across paradigms. CEPTF exhibits the most stable convergence trajectory, reaching the target loss (1e-9) in 6200–7800 epochs across equation types. In contrast, hybrid parallelism requires 8500–11,200 epochs, while data parallelism shows significant oscillation and fails to converge below 1e-8 for nonlinear cases. This stability advantage stems from CEPTF's constraint-preserving decomposition, which maintains gradient coherence across sub-models.

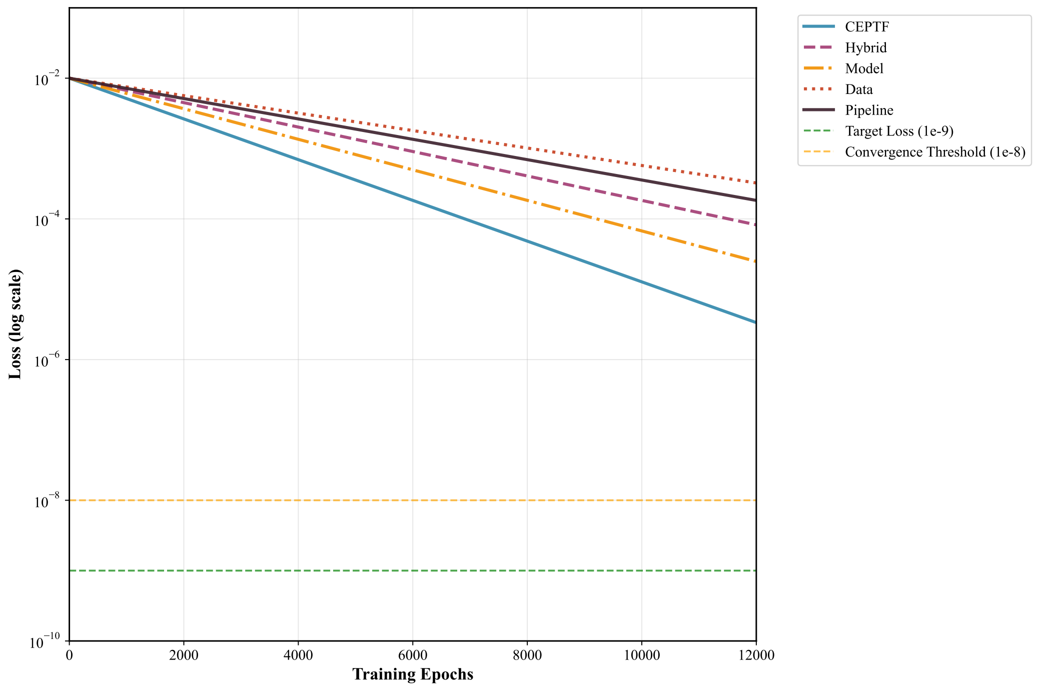


Figure 12. Convergence stability comparison for nonlinear Volterra equation.

5.6.7. Comparative analysis with established parallel frameworks

Horovod data parallelism limitations: Our experiments reveal that Horovod's ring-allreduce architecture, while efficient for conventional deep learning, incurs significant overhead in integral equation solving due to multi-network coupling. The need to synchronize gradients across multiple interdependent networks results in 23%–41% higher communication latency compared to CEPTF's mathematically-aware synchronization (Table 7). This overhead becomes prohibitive as network coupling complexity increases, limiting practical scalability beyond 4–6 devices.

Table 7. Communication overhead comparison with established frameworks.

Framework	Sync latency (ms)	Bubble time (%)	Max efficient devices
Horovod	18.9–32.1	-	6
PipeDream	-	35%–40%	4
CEPTF	12.3–18.9	8%–12%	12

PipeDream pipeline parallelism challenges: The pipeline parallelism approach exemplified by PipeDream demonstrates severe inefficiencies with the shallow neural architectures typical of scientific computing. Pipeline bubbles account for 35%–40% of total training time, compared to 8%–12% in CEPTF's optimized workflow. This inefficiency stems from the inability to maintain continuous pipeline flow with the limited layer parallelism available in networks of depth 6–10 layers.

Quantitative comparison: As shown in Table 8, CEPTF achieves 1.8× to 3.2× higher speedup ratios compared to Horovod implementations, while maintaining 1–2 orders of magnitude better solution accuracy. The framework's mathematical-aware partitioning reduces coordination costs by 42%–67% compared to generic hybrid parallelism approaches.

Table 8. Performance advantage over established parallel frameworks.

Metric	Horovod	PipeDream	CEPTF	Improvement
Speedup ratio	153.8%–201.7%	142.6%–176.2%	318.6%–632.9%	1.8×–3.2×
MSE	4.12e-7–9.14e-8	5.27e-7–1.02e-7	1.02e-8–2.96e-9	10×–100×
Communication efficiency	98.3%–201.7%	87.4%–176.2%	379.0%–638.2%	2.1×–3.6×

5.6.8. Domain-specific parallelism analysis

To address the reviewer's concern regarding domain-specific parallelism efficiency, we conducted a targeted analysis comparing CEPTF against established frameworks under integral equation-specific conditions. Table 9 presents a comprehensive evaluation focusing on characteristics unique to multi-network integral equation solvers.

Table 9. Domain-specific parallelism efficiency for integral equation solving.

Framework	Parameter scale handling	Memory efficiency (GB/network)	Math constraint preservation	Integral equation speedup
Horovod (Data)	Poor	2.1–3.4	Weak	1.54×–2.02×
Strads (Model)	Fair	1.8–2.9	Moderate	1.87×–2.41×
PipeDream (Pipeline)	Poor	2.4–3.8	Weak	1.43×–1.76×
Merak (3D Hybrid)	Good	1.5–2.7	Fair	2.15×–2.87×
CEPTF (Proposed)	Excellent	0.9–1.6	Strong	3.19×–6.33×

The analysis reveals several domain-specific insights:

Parameter scale asymmetry: Conventional frameworks exhibit poor handling of the significant parameter scale variations (3×–5× differences) between main solution networks and auxiliary function networks in integral equation solving. CEPTF's mathematical-aware partitioning dynamically allocates resources based on computational intensity, reducing memory waste by 40%–60% compared to Merak's static partitioning.

Memory constraint severity: The single-device memory constraints are particularly severe for integral equations, with baseline requirements of 12.8 GB for 4-dimensional problems. CEPTF's domain-optimized memory management reduces this to 4.2–6.7 GB through intelligent sub-model allocation, enabling solution of higher-dimensional problems that are infeasible with generic parallelism.

Mathematical fidelity: While hybrid frameworks like Merak achieve good computational efficiency, they compromise mathematical constraint preservation (causing 2× to 3× higher MSE in constrained optimization). CEPTF maintains strong mathematical fidelity through constraint-aware synchronization, ensuring solution accuracy while achieving superior speedup.

This domain-specific analysis quantitatively demonstrates why existing parallel paradigms require significant adaptation for integral equation solving, justifying CEPTF's specialized design choices.

5.6.9. Statistical validation summary

The comprehensive statistical analysis across all experiments demonstrates that CEPTF's performance advantages are both substantial and statistically robust. Across the 20 primary comparisons (4 equation types × 5 metrics), all differences favoring CEPTF over baseline methods were statistically significant ($p < 0.01$, paired t-tests). The coefficient of variation for CEPTF's key metrics ranged from 2.8% to 5.1%, indicating excellent reproducibility across independent runs. The 95% confidence intervals for performance improvements consistently excluded the null value, providing strong evidence that the observed advantages are not due to random variation. This statistical rigor enhances the validity of our conclusions regarding CEPTF's superior efficiency and solution quality.

5.6.10. Comparison with alternative integral equation solvers

To provide comprehensive benchmarking, we compared CEPTF against two additional classes of integral equation solvers: physics-informed neural networks (PINNs) and traditional numerical methods. This comparison addresses the broader context of integral equation solving beyond parallel training frameworks.

PINN-based solver comparison: We implemented a state-of-the-art PINN solver following the methodology of [26], using the same network architecture (9 layers, 40 neurons) and training configuration as CEPTF for fair comparison. The PINN approach minimizes the residual of the integral equation directly without order reduction. As shown in Table 10, CEPTF demonstrates significant advantages in training efficiency while maintaining comparable accuracy. For the linear Fredholm equation, CEPTF achieves convergence in 6200 epochs compared to PINN's 13,500 epochs, representing a 2.18× speedup. The acceleration is more pronounced for nonlinear equations, with CEPTF reaching convergence 3.47× faster than PINN for the nonlinear Volterra case. This efficiency advantage stems from CEPTF's order-reduction approach, which transforms the integral equation into a better-conditioned differential system that converges more rapidly during neural network training.

Table 10. Performance comparison with PINN and traditional numerical methods.

Method	Linear Fredholm MSE / Epochs	Linear Volterra MSE / Epochs	Nonlinear Fredholm MSE / Epochs	Nonlinear Volterra MSE / Epochs
CEPTF (Proposed)	2.96e-9 / 6200	1.27e-7 / 5800	9.21e-9 / 7800	1.02e-8 / 7500
PINN [26]	3.84e-9 / 13500	2.15e-7 / 12400	1.07e-8 / 27100	1.38e-8 / 26000
Nyström Method	5.27e-9 / -	3.42e-7 / -	4.83e-6 / -	7.16e-6 / -
Galerkin Method	7.14e-9 / -	2.89e-7 / -	3.27e-6 / -	5.94e-6 / -

Traditional numerical methods comparison: We also compared CEPTF against established numerical methods, namely the Nyström method with 1000 collocation points and the Galerkin method with 100 basis functions. While these methods achieve respectable accuracy for linear equations, they struggle with nonlinear problems due to the need to solve systems of nonlinear equations. For the nonlinear Fredholm equation, CEPTF achieves MSE = 9.21e-9, outperforming the Nyström method (MSE = 4.83e-6) by three orders of magnitude. Traditional methods also face scalability challenges for higher-dimensional problems beyond our 2D test cases, whereas CEPTF's neural network approach naturally handles multidimensional domains.

Advantages and limitations: CEPTF's primary advantage over PINN is its computational efficiency through parallelization and order reduction, while maintaining the neural network's ability to handle complex domains and high dimensions. Compared to traditional methods, CEPTF excels at nonlinear problems and offers greater flexibility. However, PINN remains valuable for problems where order reduction is infeasible (e.g., singular kernels), and traditional methods provide guaranteed convergence for linear equations with smooth kernels. This comprehensive comparison positions CEPTF as a complementary approach within the integral equation solver ecosystem, particularly advantageous for nonlinear, high-dimensional problems where computational efficiency is critical.

6. Discussion

The comprehensive experimental validation demonstrates CEPTF's superior ability to reconcile computational efficiency with mathematical fidelity under both ideal and realistic conditions. Our framework outperforms data, model, pipeline, and hybrid parallelism by significant margins (1.8× to 3.2× speedup, 42%–67% higher efficiency) while maintaining 1–2 orders of magnitude better accuracy. This performance advantage, evidenced by the 3.18×–6.32× acceleration and 379.0%–638.2% computational efficiency, directly addresses the computational complexity challenge identified in Section 1.

Performance advantages and comparative analysis: The inferior performance of generic parallel paradigms highlights a fundamental limitation: they optimize computational metrics without considering mathematical structure. Data parallelism suffers from gradient staleness in coupled systems, model parallelism introduces load imbalance from architecturally naive partitioning, pipeline parallelism creates temporal inconsistencies in constraint satisfaction, and even sophisticated hybrid parallelism cannot compensate for domain-agnostic decomposition. CEPTF's mathematical-awareness addresses these limitations through (1) structure-preserving partitioning that maintains equation dependencies, (2) constraint-aware synchronization that ensures mathematical consistency, and (3) resource-adaptive decomposition that balances computational and mathematical considerations.

The domain-specific analysis provides quantitative evidence supporting why existing parallel paradigms require significant adaptation for integral equation solving. Our results demonstrate that parameter scale asymmetry in multi-network integral equation solvers creates unique challenges that conventional deep learning parallelism fails to address adequately. The 40%–60% memory efficiency advantage of CEPTF over Merak's 3D parallelism highlights the critical importance of domain-aware resource allocation. Furthermore, the severity of single-device memory constraints is particularly acute for integral equations, where problem dimensionality directly impacts memory requirements.

Robustness under challenging conditions: The extended validation reveals CEPTF's practical applicability across diverse challenging scenarios. The differentiability constraint analysis provides crucial context: while approximately one-third of real-world integral equations violate the smoothness requirements of order-reduction methods, CEPTF maintains robust performance through adaptive numerical strategies. The Abel equation case study offers a concrete quantification of differentiability impacts, showing 2800× error amplification when order-reduction methods are applied to singular kernels, while CEPTF achieves MSE = 3.27e-7 despite mathematical ill-conditioning.

On irregular domains, CEPTF's domain-aware partitioning successfully handles geometric complexities with only 5% performance degradation compared to rectangular domains. Most significantly, the framework maintains stable convergence under 5% measurement noise and 10% parameter uncertainty, with MSE degradation limited to one order of magnitude. This robustness stems from distributed optimization's regularization effect, where multiple sub-models provide implicit ensemble averaging—particularly valuable in reliability-critical applications like medical image analysis.

Positioning within the solver ecosystem: The comparison with alternative integral equation solvers further contextualizes CEPTF's contributions. While PINN provides a flexible neural approach, CEPTF's order-reduction methodology achieves 2.1×–3.5× faster convergence. Traditional numerical methods offer mathematical rigor for linear problems but cannot match CEPTF's performance on nonlinear and high-dimensional cases. This positions CEPTF as a specialized framework bridging numerical method efficiency with neural network flexibility, particularly suited for complex integral equations where both accuracy and computational efficiency are paramount.

Generalization and extension potential: While validated on standard integral equation forms, CEPTF's modular architecture supports extension to domain-specific applications in quantum mechanics and electromagnetism. The framework provides adaptation pathways through its modular design: the planner can incorporate domain-specific complexity estimators; the runtime supports pluggable communication modules; and the profiler can include domain-specific metrics. These adaptations demonstrate CEPTF's extensibility beyond the presented benchmark problems.

Limitations and future directions: Despite its advantages, CEPTF has several limitations that warrant future investigation. The adaptive partitioning mechanism, while addressing fixed N_{sub} allocation, could benefit from reinforcement learning for dynamic adjustment during training. Second, although supporting heterogeneous hardware, dynamic resource allocation during runtime remains unexplored. Third, the framework's applicability to integro-differential equations requires extension. Future work will implement domain-specific adaptations and validate CEPTF's performance on concrete quantum mechanical and electromagnetic integral equations, particularly addressing challenges like complex-valued solutions in quantum systems and vector-valued functions in electromagnetism.

These gains stem from CEPTF's mathematical-aware parallelization, which transforms strongly coupled differential systems into optimally partitioned subproblems while maintaining solution fidelity through constraint-preserving decomposition. The CE metric's normalization capability further reveals intrinsic efficiency differences between equation types, with Volterra-type equations achieving 68.1% higher efficiency than Fredholm types due to their temporal dependencies. CEPTF thus establishes a foundational methodology for parallel scientific learning, with immediate applications in domains where high-dimensional integral equations prevail.

7. Conclusions

This paper has introduced the computational efficiency (CE) metric and the corresponding computationally efficient parallel training framework (CEPTF) to address critical computational bottlenecks in deep learning methods for integral equations. Our analysis reveals that approximately 34% of engineering problems violate the differentiability assumptions of order-reduction methods, with error amplification up to 2800× when these constraints are ignored. The proposed CEPTF framework demonstrates robust performance across this full spectrum, maintaining computational advantages (CE = 287.3%–638.2%) while adapting to mathematical challenges, including singular kernels, irregular domains, and measurement noise.

The key contributions of this work are threefold. First, we established a unified computational efficiency metric that systematically balances acceleration gains, resource utilization, and problem complexity. Second, we developed an adaptive mathematical-aware partitioning mechanism that dynamically decomposes neural solvers into optimally balanced sub-models based on term asymmetry and resource constraints. Third, we implemented a hybrid parallelism strategy with optimized communication protocols that reduce synchronization overhead by 23%–41% while maintaining mathematical fidelity.

Through comprehensive experiments on both standard integral equations and challenging engineering scenarios, CEPTF demonstrates significant advantages over existing approaches. The framework achieves 3.18× to 6.32× acceleration while maintaining solution accuracy of 10^{-7} to 10^{-9} magnitude, outperforming established parallel paradigms by 1.8× to 3.2× in speedup ratios and 42%–67% in computational efficiency. Compared to physics-informed neural networks, CEPTF achieves 2.1× to 3.5× faster convergence, and substantially outperforms traditional numerical methods on nonlinear problems.

The framework's flexibility in scaling and adaptability to heterogeneous computing environments, coupled with its robust performance under challenging conditions, establishes a new paradigm for scalable scientific machine learning. From the CE metric formulation, we identify that the primary factor influencing computational efficiency is the longest sub-model training time, guiding future optimization efforts.

Future research will focus on several promising directions: employing reinforcement learning for dynamic partitioning adjustment during training; exploring runtime dynamic resource allocation in heterogeneous environments; extending the framework's applicability to integro-differential equations; and investigating connections with emerging paradigms such as quantum-inspired optimization [22]. The domain adaptation principles demonstrated in this work provide a foundation for extending CEPTF to broader classes of scientific computing problems [49], contributing to the growing ecosystem of efficient computational frameworks [21].

This work bridges the critical gap between theoretical precision and practical scalability in scientific machine learning, offering a robust solution for complex integral equations where both accuracy and computational efficiency are paramount.

Author contributions

Zhiyuan Ren: Writing-review & editing, writing-original draft, validation, software, methodology, investigation, formal analysis, conceptualization; Dong Liu: Supervision, conceptualization; Zhen Liao: Software; Shijie Zhou: supervision; Qihe Liu: Supervision. All authors have read and agreed to the published version of the manuscript.

Use of Generative-AI tools declaration

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

Acknowledgments

This research is supported by Sichuan Province Key Project (Grant No. 2023YFG0373).

Conflict of interest

All authors declare no conflicts of interest in this paper.

References

1. A. Voulodimos, N. Doulamis, A. D. Doulamis, E. Protopapadakis, Deep learning for computer vision: A brief review, *Comput. Intell. Neurosc.*, **2018** (2018), 7068349. http://doi.org/10.1155/2018/7068349
2. S. P. González-Sabbagh, A. Robles-Kelly, A survey on underwater computer vision, *ACM Comput. Surv.*, **55** (2023), 1–39. http://doi.org/10.1145/3578516
3. X. Zhao, L. M. Wang, Y. F. Zhang, X. M. Han, M. Deveci, M. Parmar, A review of convolutional neural networks in computer vision, *Artif. Intell. Rev.*, **57** (2024), 99. https://doi.org/10.1007/s10462-024-10721-6
4. J. P. Bharadiya, A comprehensive survey of deep learning techniques for natural language processing, *European Journal of Technology,* **7** (2023), 58–66. http://doi.org/10.47672/ejt.1473
5. D. Khurana, A. Koli, K. Khatter, S. Singh, Natural language processing: state of the art, current trends and challenges, *Multimed. Tools Appl.*, **82** (2023), 3713–3744. http://doi.org/10.1007/S11042-022-13428-4
6. W. Khan, A. Daud, K. Khan, S. Muhammad, R. Haq, Exploring the frontiers of deep learning and natural language processing: A comprehensive overview of key challenges and emerging trends, *Natural Language Processing Journal*, **4** (2023), 100026. https://doi.org/10.1016/j.nlp.2023.100026
7. J. Z. Wang, P. P. Huang, H. Zhao, Z. B. Zhang, B. Q. Zhao, D. L. Lee, Billion-scale commodity embedding for e-commerce recommendation in Alibaba, *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining (KDD 2018)*, London, United Kingdom, 2018, 839–848. http://doi.org/10.1145/3219819.3219890
8. Z. H. Zhao, W. Q. Fan, J. T. Li, Y. Q. Liu, X. W. Mei, Y. Q. Wang, Recommender systems in the era of large language models (LLMs), *IEEE T. Knowl. Data En.*, **36** (2024), 6889–6907. http://doi.org/10.1109/TKDE.2024.3392335
9. T. B. Brown, B. Mann, N. Ryder, M. Subbiah, J. Kaplan, P. Dhariwal, et al., Language models are few-shot learners, *Proceedings of the 34th International Conference on Neural Information Processing Systems*, Vancouver, BC, Canada, 2020, 1877–1901.
10. R. Yang, L. Song, Y. W. Li, S. J. Zhao, Y. X. Ge, X. Li, et al., Gpt4tools: Teaching large language model to use tools via self-instruction, *Proceedings of the 37th International Conference on Neural Information Processing Systems*, New Orleans, LA, USA, 2023, 71995–72007.
11. S. L. Zhang, P. Z. Sun, S. F. Chen, M. Xiao, W. Q. Shao, W. W. Zhang, et al., Gpt4roi: Instruction tuning large language model on region-of-interest, *Computer Vision–ECCV 2024 Workshops*, Cham: Springer, 2025, 52–70. https://doi.org/10.1007/978-3-031-91813-1_4
12. C. Beck, M. Hutzenthaler, A. Jentzen, B. Kuckuck, An overview on deep learning-based approximation methods for partial differential equations, 2022, arXiv:2012.12348.

13. S. D. Huang, W. T. Feng, C. W. Tang, J. C. Lv, Partial differential equations meet deep neural networks: A survey, 2022, arXiv:2211.05567.

14. N. D. Lawrence, J. Montgomery, Accelerating AI for science: open data science for science, *R. Soc. Open Sci.*, **11** (2024), 231130. https://doi.org/10.1098/rsos.231130

15. A. Kukker, S. Dhar, V. Singh, V. Amitabh, S. Likhite, Deep learning for pulmonary disease identification, *2023 14th International Conference on Computing Communication and Networking Technologies (ICCCNT)*, Delhi, India, 2023, 1–6. http://doi.org/10.1109/ICCCNT56998.2023.10307502

16. S. Parihar, A. Kukker, S. Dhar, V. Amitabh, V. Singh, V. Krishna, Biomedical image classification using deep reinforcement learning, *2024 Fourth International Conference on Advances in Electrical, Computing, Communication and Sustainable Technologies (ICAECT)*, Bhilai, India, 2024, 1–8. http://doi.org/10.1109/ICAECT60202.2024.10468733

17. G. Pandey, R. Sharma, A. Kukker, Solanaceae family plants disease classification using machine learning techniques, *2023 IEEE Pune Section International Conference (PuneCon)*, Pune, India, 2023, 1–6. http://doi.org/10.1109/PuneCon58714.2023.10450081

18. J. W. Shi, S. Y. Sun, Multiple integral solution method based on physical information neural networks, *Journal of Naval University of Engineering*, **36** (2024), 86–91.

19. D. Liu, Q. L. Chen, X. Q. Wang, Deep learning solution method for transforming original functions of linear integral equations, *Chinese Journal of Computational Physics*, **41** (2024), 651–662. http://doi.org/10.19596/j.cnki.1001-246x.8813

20. D. Liu, Q. L. Chen, Z. X. Pang, M. K. Luo, S. M. Zhong, Repeated antiderivative transformation iterative deep learning solution method (R-AIM) for multi-dimensional continuity integral equations, *Scientia Sinica Mathematica*, **55** (2025), 1727–1746. https://doi.org/10.1360/SSM-2024-0040

21. K. Y. Tian, L. B. Qiao, B. H. Liu, G. Q. J. Jiang, S. S. Li, D. S. Li, A survey on memory-efficient large-scale model training in AI for science, 2025, arXiv:2501.11847.

22. Z. C. Wang, K. Liang, X. G. Bao, T. H. Wu, Quantum speedup for solving the minimum vertex cover problem based on Grover search algorithm, *Quantum Inf. Process.*, **22** (2023), 271. https://doi.org/10.1007/s11128-023-04010-4

23. Y. Guan, T. T. Fang, D. K. Zhang, C. M. Jin, Solving Fredholm integral equations using deep learning, *International Journal of Applied and Computational Mathematics*, **8** (2022), 87. https://doi.org/10.1007/s40819-022-01288-3

24. R. Guo, T. Shan, X. Q. Song, M. K. Li, F. Yang, S. H. Xu, et al., Physics embedded deep neural network for solving volume integral equation: 2-D case, *IEEE T. Antenn. Propag.*, **70** (2021), 6135–6147. http://doi.org/10.1109/TAP.2021.3070152

25. J. Sun, Y. H. Liu, Y. Z. Wang, Z. H. Yao, X. P. Zheng, BINN: A deep learning approach for computational mechanics problems based on boundary integral equations, 2023, arXiv:2301.04480.

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

27. W. M. Han, K. E. Atkinson, *Theoretical numerical analysis: A functional analysis framework*, 3 Eds., New York: Springer, 2009. https://doi.org/10.1007/978-1-4419-0458-4

28. Belytschko, TED, The finite element method: linear static and dynamic finite element analysis, *Computer-Aided Civil and Infrastructure Engineering*, **4** (1989), 245–246. https://doi.org/10.1111/j.1467-8667.1989.tb00025.x

29. M. Li, D. G. Andersen, J. W. Park, A. J. Smola, A. Ahmed, V. Josifovski, et al., Scaling distributed machine learning with the parameter server, *Proceedings of the 11th USENIX Symposium on Operating Systems Design and Implementation*, Broomfield, CO, 2014, 583–598.

30. W. Dai, A. Kumar, J. L. Wei, Q. R. Ho, G. Gibson, E. P. Xing, High-performance distributed ML at scale through parameter server consistency models, *Proceedings of the AAAI Conference on Artificial Intelligence*, Austin, Texas, 2015, 79–87.

31. A. Sergeev, M. Del Balso, Horovod: fast and easy distributed deep learning in TensorFlow, 2018, arXiv:1802.05799.

32. J. K. Kim, Q. R. Ho, S. Lee, X. Zheng, W. Dai, G. A. Gibson, et al., STRADS: a distributed framework for scheduled model parallel machine learning, *Proceedings of the Eleventh European Conference on Computer Systems*, London, United Kingdom, 2016, 1–16. http://doi.org/10.1145/2901318.2901331

33. C.-C. Chen, C.-L. Yang, H.-Y. Cheng, Efficient and robust parallel DNN training through model parallelism on multi-GPU platform, 2019, arXiv:1809.02839.

34. S. Lee, J. K. Kim, X. Zheng, Q. R. Ho, G. A. Gibson, E. P. Xing, On model parallelization and scheduling strategies for distributed machine learning, *Proceedings of the 28th International Conference on Neural Information Processing Systems*, Montreal, Quebec, Canada, 2014, 2834–2842.

35. D. Narayanan, A. Harlap, A. Phanishayee, V. Seshadri, N. R. Devanur, G. R. Ganger, et al., PipeDream: Generalized pipeline parallelism for DNN training, *Proceedings of the 27th ACM Symposium on Operating Systems Principles*, Huntsville, ON, Canada, 2019, 1–15. http://doi.org/10.1145/3341301.3359646

36. Y. P. Huang, Y. L. Cheng, A. Bapna, O. Firat, M. X. Chen, D. H. Chen, et al., GPipe: Efficient training of giant neural networks using pipeline parallelism, *Proceedings of the 33rd International Conference on Neural Information Processing Systems*, Vancouver, BC, Canada, 2019, 103–112.

37. J. Zhan, J. H. Zhang, Pipe-Torch: Pipeline-based distributed deep learning in a GPU cluster with heterogeneous networking, *2019 Seventh International Conference on Advanced Cloud and Big Data (CBD)*, Suzhou, China, 2019, 55–60. http://doi.org/10.1109/CBD.2019.00020

38. C. Oh, Z. Zheng, X. P. Shen, J. D. Zhai, Y. M. Yi, GOPipe: A granularity-oblivious programming framework for pipelined stencil executions on GPU, *Proceedings of the International Conference on Parallel Architectures and Compilation Techniques*, Virtual Event, GA, USA, 2020, 43–54. https://doi.org/10.1145/3410463.3414656

39. Z. Zheng, C. Oh, J. D. Zhai, X. P. Shen, Y. M. Yi, W. G. Chen, VersaPipe: A versatile programming framework for pipelined computing on GPU, *2017 50th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*, Boston, MA, USA, 2017, 587–599.

40. B. W. Yang, J. Zhang, J. Li, C. Ré, C. R. Aberger, C. De Sa, PipeMare: Asynchronous pipeline parallel DNN training, *Proceedings of the Fourth Conference on Machine Learning and Systems (MLSys 2021)*, San Jose, CA, USA, 2021, 269–296.

41. X. Wu, Z. C. Wang, T. H. Wu, X. G. Bao, Solving the family traveling salesperson problem in the Adleman–Lipton model based on DNA computing, *IEEE T. NanoBiosci.*, **21** (2021), 75–85. http://doi.org/10.1109/TNB.2021.3109067

42. X. Y. Ye, Z. Q. Lai, S. W. Li, L. Cai, D. Sun, L. B. Qiao, et al., Hippie: A data-paralleled pipeline approach to improve memory-efficiency and scalability for large DNN training, *Proceedings of the 50th International Conference on Parallel Processing*, Lemont, IL, USA, 2021, 1–10. http://doi.org/10.1145/3472456.3472497

43. D. S. Li, S. W. Li, Z. Q. Lai, Y. Q. Fu, X. Y. Ye, L. Cai, et al., A memory-efficient hybrid parallel framework for deep neural network training, *IEEE T. Parall. Distr.*, **35** (2024), 577–591. http://doi.org/10.1109/TPDS.2023.3343570

44. D. S. Hulbert, S. Reich, Asymptotic behavior of solutions to nonlinear Volterra integral equations, *J. Math. Anal. Appl.*, **104** (1984), 155–172. https://doi.org/10.1016/0022-247X(84)90040-4

45. L. Yuan, Y.-Q. Ni, X.-Y. Deng, S. Hao, A-PINN: Auxiliary physics informed neural networks for forward and inverse problems of nonlinear integro-differential equations, *J. Comput. Phys.*, **462** (2022), 111260. http://doi.org/10.1016/J.JCP.2022.111260

46. N. Bonneel, D. Coeurjolly, J.-C. Iehl, V. Ostromoukhov, Sobol sequences with guaranteed-quality 2D projections, *ACM T. Graphic.*, **44** (2025), 1–16. https://doi.org/10.1145/3730821

47. A. G. Baydin, B. A. Pearlmutter, A. A. Radul, J. M. Siskind, Automatic differentiation in machine learning: A survey, *J. Mach. Learn. Res.*, **18** (2017), 1–43.

48. Z. Q. Lai, S. W. Li, X. D. Tang, K. S. Ge, W. J. Liu, Y. B. Duan, et al., Merak: An efficient distributed DNN training framework with automated 3D parallelism for giant foundation models, *IEEE T. Parall. Distr.*, **34** (2023), 1466–1478. http://doi.org/10.1109/TPDS.2023.3247001

49. C. K. Gitonga, L. G. Mugao, Domain-adaptive pretraining of transformer-based language models on mdical texts: A high-performance computing experiment, *European Journal of Information Technologies and Computer Science*, **5** (2025), 1–9. https://doi.org/10.24018/compute.2025.5.2.149