Fast Training of Accurate Physics-Informed Neural Networks Without Gradient Descent

ICLR 2026

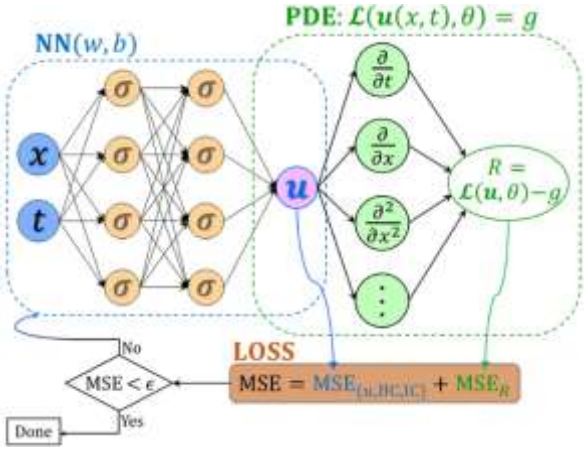
Anonymous authors Paper under double-blind review

BackGround

Partial Differential Equations (PDEs) are fundamental for modeling complex systems across science and engineering, yet remain challenging to solve efficiently. Physics-Informed Neural Networks (PINNs) show promise by incorporating physical constraints into neural network training through PDE residual minimization. However, two critical limitations hinder their performance:

(1) The optimization problem is inherently difficult—high-dimensional, multi-objective, non-convex, and ill-conditioned due to conflicting loss terms from PDE, boundary, and initial conditions;
(2) Time is treated as just another spatial dimension, ignoring the Markovian causality of temporal evolution, which leads to poor capture of high-frequency dynamics and difficulties in long-time integration.

Existing solutions (loss balancing, regularization, temporal decomposition) address symptoms rather than root causes and remain computationally expensive.



Model Construction

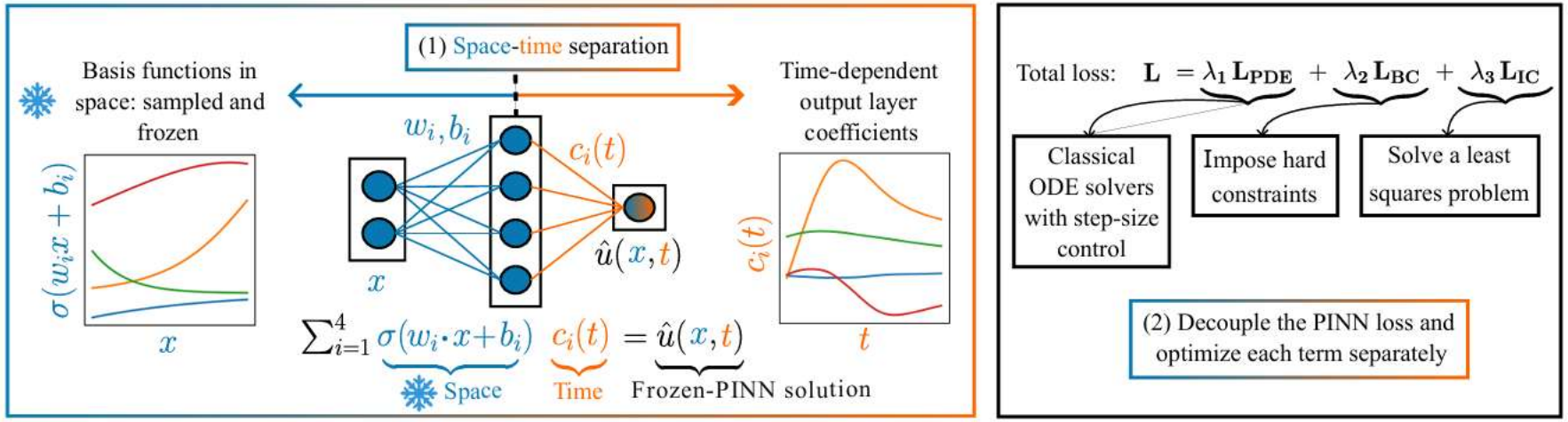


Figure 1: **Core ideas of Frozen-PINNs:** (1) **Space–time separation**: For $x \in \mathbb{R}^d$, spatial bases $\phi_i = \sigma(w_i \cdot x + b_i)$ with $\sigma = \tanh$, $w_i \in \mathbb{R}^d$, $b_i \in \mathbb{R}$ are sampled and frozen (shown for $d = 1$); output layer parameters $c_i(t)$ are evolved via ODEs. Each pair (ϕ_i, c_i) is color-matched. (2) **Loss decoupling**: PDE, boundary, and initial condition losses $L_{\text{PDE}}, L_{\text{BC}}, L_{\text{IC}}$ are optimized independently.

The method: (a) freezes randomly sampled spatial hidden layer parameters to reduce dimensionality, (b) decouples and separately optimizes each loss term, and (c) replaces gradient descent with least squares and adaptive ODE solvers for computing time-dependent output parameters.

Frozen layer parameter initialization

Slide: Hidden Layer Parameter Sampling

Authors' Approach: Two Strategies

Method	Type	Sampling Rule	Best For
ELM	Data-agnostic	W ~ N(0,1), b ~ U(−η,η)	Smooth solutions
SWIM	Data-dependent	Align with collocation pairs	Shock/steep gradients

SWIM Formula

$$w_m = s_1 \frac{x^{(2)} - x^{(1)}}{\|x^{(2)} - x^{(1)}\|}, \quad b_m = -\langle w_m, x^{(1)} \rangle + s_2$$

Sampling Modes

Unsupervised (Default)

- Uniform sampling over all collocation pairs
- No dependence on source term $f(x)$

Supervised

- Weighted sampling with density:

$$\rho(x^{(1)}, x^{(2)}) = \frac{\|f(x^{(2)}) - f(x^{(1)})\|}{\|x^{(2)} - x^{(1)}\|}$$

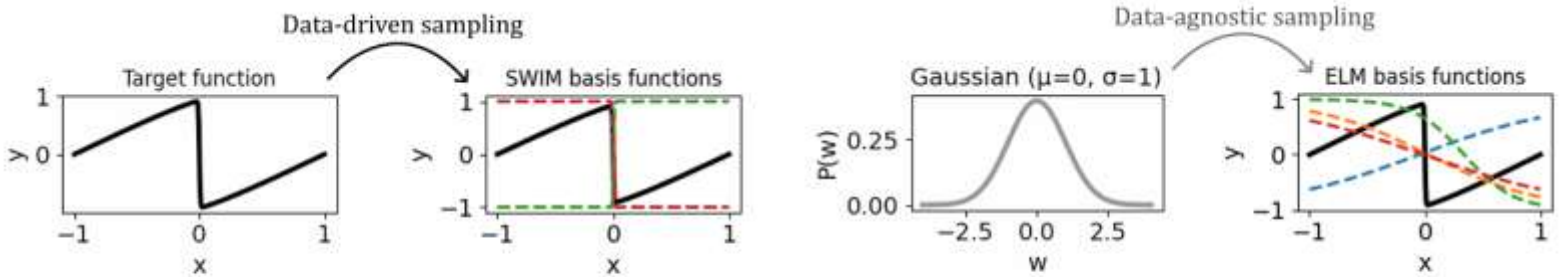


Figure 2: Sampling in Frozen-PINNs: (Left): SWIM (data-driven, places bases with steep gradients near regions with shocks) vs. (Right): ELM (data-agnostic, no control over basis placement).

By inserting the parametrized ansatz

$$\hat{u}(x,t) = C(t)[\Phi(x), \mathbb{1}] = c(t)\sigma(Wx^\top + b) + c_0(t). \tag{1}$$

into the PDE:

$$u_t(x,t) \; + \; \mathcal{L}u(x,t) + \gamma\mathcal{N}(u)(x,t) = f(x), \; x \in \Omega, \; t \in [0,T], \tag{2}$$

$$\mathcal{B}u(x,t) = g(x), \qquad x \in \partial\Omega, \; t \in [0,T], \quad \text{and}, \quad u(x,0) = u_0(x), \quad x \in \Omega,$$

and reformulating it as an ordinary differential equation (ODE) for c(t), the authors compute the time-dependent output-layer parameters c(t) while preserving the inherent causal structure of time-dependent PDEs, thereby enforcing temporal causality by design. The author gets：

$$C_t(t) = R(X, C(t))[\Phi(X), \mathbb{1}]^+, \quad \text{where} \tag{3}$$
$$R(X, C(t)) = -C(t)\mathcal{L}[\Phi(X), \mathbb{1}] - \gamma\mathcal{N}(C(t)[\Phi(X), \mathbb{1}]) + [f(X)]^\top,$$

Method for satisfying frozen nail boundary conditions

They proposed two different strategies to satisfy the boundary conditions of Frozen-PINNs: the first utilizes a boundary-compatible layer, and the second is an augmented reformulation of the ordinary differential equations (ODEs).

Boundary-compliant layer: Some boundary conditions can be enforced through linear mapping $A \in \mathbb{R}^{M_b \times M}$

$$C_t(t) = R(X, C(t))\Phi_A(X)^+, \quad \text{where} \tag{4}$$
$$R(X, C(t)) = -C(t)\mathcal{L}\Phi_A(X) - \gamma\mathcal{N}(C(t)\Phi_A(X)) + [f(X)]^\top.$$

AugmentedODE: This strategy eliminates the need for layers that satisfy the boundary by adding a corrective term that enforces the boundary condition in the ODE. For Dirichlet boundary conditions：

$u(x) = g(x)$, we add $\hat{u}_t(x) = -\kappa(\hat{u}(x) - g(x))$ for $x \in \partial\Omega$ and solve the augmented system: (5)

$$C_t(t) = \underbrace{[R(X, C(t)), -\kappa(C(t)\Phi_A(X_b) - g(X_b)^\top)]}_{\in \mathbb{R}^{1\times(N_c+N_b)}} \underbrace{\Phi_A([X, X_b])^+}_{\in \mathbb{R}^{(N_c+N_b)\times(M_b+1)}},$$

Algorithm

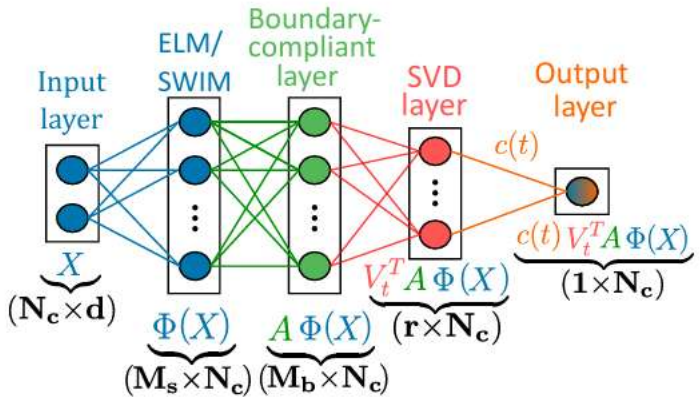


Figure 3: Architecture of Frozen-PINNs trained with a gradient-descent-free training algorithm.

Algorithm 1 Frozen-PINN training algorithm

Input: PDE (Equation (1)), test grid points $X_{\text{test}} \times T_{\text{test}}$
Output: PDE Solution on the test grid points $\hat{u}(X_{\text{test}}, T_{\text{test}})$
Parameters: $N_c, M_s, M_b \in \mathbb{N}, \epsilon_{SVD} \in \mathbb{R}$

1: Sample N_c collocation points: $X \in \mathbb{R}^{N_c \times d}$
2: Construct hidden layer params $\{w_m, b_m\}_{m=1}^{M_s}$ (SWIM/ELM) ▷ Section 2.2
3: Compute hidden layer output $\Phi(X) \in \mathbb{R}^{M_s \times N_c}$
4: Construct *boundary-compliant layer*: $A\Phi(X) \in \mathbb{R}^{M_b \times N_c}$ ▷ Section 2.4
5: Compute truncated SVD: $V_r \Sigma_r U_r^\top = A\Phi(X)$ and *SVD layer* output $V_r^\top A\Phi(X) = A_r\Phi(X)$
6: Compute neural bases: $\Phi_{A_r}(X) := (A_r\Phi(X), 1)^\top \in \mathbb{R}^{(r+1)\times N_c}$
7: Initialize *output-layer* params (least-squares): $C(0) = u(X,0)^\top \Phi_{A_r}(X)^+$
8: Solve ODE for $C(t) \in \mathbb{R}^{1\times(r+1)}$ using Φ_{A_r} ▷ Equation (4)
9: Evaluate $\hat{u}(X_{\text{test}}, T_{\text{test}}) = C(T_{\text{test}})\Phi_{A_r}(X_{\text{test}})$ ▷ Equation (2)

PDE benchmark	Method	Training time (s)	Normalized training time	Relative L^2 error
Advection ($\beta = 40$)	PINN (Adam)	-	-	Fail for β=40
	SWIM	-	-	Fail for β=40
	ELM	-	-	Fail for β=40
	Causal PINN	357.63	533	2.90e0 ± 1.2e0
	PINN (L-BFGS)	30.5	45.5	6.92e-1 ± 2.96e-2
Krishnapriyan et al. (2021)	PINN (seq2seq, L-BFGS)	-	-	2.41e-1
Krishnapriyan et al. (2021)	PINN (Curriculum training, L-BFGS)	-	-	5.33e-2
	Frozen-PINN-elm (our) ❄	**0.67**	1	**4.19e-3 ± 2.97e-3**
	Frozen-PINN-swim (our) ❄	0.7	1	8.42e-9 ± 1.12e-8
	Mesh-based method (IGA)	**0.07**	0.1	**1.17e-10**
Euler-Bernoulli (classical)	PINN (Adam)	4209.82	84196	3.95e-2 ± 1.79e-2
Kapoor et al. (2023)	PINN (L-BFGS)	2303.71	46074	4.21e-3 ± 9.56e-4
	Frozen-PINN-elm (our) ❄	**0.05**	1	**2.82e-4 ± 2.15e-4**
	Mesh-based method (IGA)	**0.94**	0.13	4.21e-7
	Frozen-PINN-elm (our) ❄	6.90	1	**9.33e-9 ± 4.36e-9**
Euler-Bernoulli (Winkler)	PINN (L-BFGS)	1858^+	37160^+	5.33e+0
Kapoor et al. (2024b)	Adaptive PINN	3807.89	76140	5.32e+0
Kapoor et al. (2024b)	Self-adaptive PINN	4042.57	80840	5.15e+0
Kapoor et al. (2024b)	Wavelet PINN	4764.25	95280	4.38e+0
Kapoor et al. (2024b)	Causal PINN	1873^+	37460^+	3.00e-2
	Frozen-PINN-elm (our) ❄	**0.05**	1	**1.41e-2 ± 4.19e-3**
	Frozen-PINN-swim (our) ❄	2.41	1	1.42e-7 ± 1.20e-7
	Mesh-based method (IGA)	**1.08**	0.44	**2.70e-8**
Wave Hao et al. (2024)	PINN (FBPINN)	3090^+	5517.9^+	5.91e-1 ± 4.74e-2
Hao et al. (2024)	PINN (L-BFGS)	350^+	625^+	5.88e-1 ± 9.63e-2
Hao et al. (2024)	PINN (gPINN)	775^+	1383.9^+	5.56e-1 ± 1.67e-2
Hao et al. (2024)	PINN (NTK)	840^+	1500^+	9.79e-2 ± 7.72e-3
	Frozen-PINN-elm (our) ❄	**0.56**	1	**1.81e-6 ± 1.01e-6**
Burgers	Causal PINN	1531.79	2945.75	1.60e-2 ± 8.97e-3
	PINN (L-BFGS)	275.2	529.2	3.88e-3 ± 2.61e-3
Kiyani et al. (2025)	PINN (BFGS with trust region)	24^+	46.1^+	1.1e-3
	Frozen-PINN-swim (our) ❄	**0.52**	1	1.00e-3 ± 1.13e-3
Chen et al. (2024b)	PINN (residual-based attention)	-	-	8.22e-4 ± 2.33e-4
McClenny & Braga-Neto (2023)	Self-adaptive PINN	-	-	4.80e-4 ± 1e-4
Chen et al. (2024b)	PINN (balanced residual decay rate)	-	-	1.38e-4 ± 0.85e-4
Kiyani et al. (2025)	PINN (RAdam + BFGS)	1070	203^+	6e-6
Urbán et al. (2025)	PINN (SSBroyden)	-	-	2.9e-6 ± 0.4e-6
	Frozen-PINN-swim (our) ❄	**5.25**	1	2.27e-7 ± 6.89e-8
	Mesh-based method (IGA)	76.32	14.5	1.12e-7
Kiyani et al. (2025)	**PINN (Adam + SSBroyden)**	2812^+	535^+	**1.62e-8**
Nonlinear diffusion	PINN (Adam)	81.36	145.2	2.09e-2 ± 3.14e-3
	PINN (L-BFGS)	255.9	456.9	1.22e-2 ± 2.38e-4
	Mesh-based method (FEM)	2.71	4.83	2.68e-3
	Frozen-PINN-elm (our) ❄	**0.56**	1	**2.60e-3 ± 1.61e-3**
	Frozen-PINN-swim (our) ❄	423	-	**2.06e-6 ± 1.99e-6**
5-d Reaction diffusion	PINN (Adam)	171.43	3428.6	3.40e-1 ± 1.79e-2
	PINN (L-BFGS)	183.38	3667.6	3.33e-2 ± 1.54e-2
Zang et al. (2020)	Weak Adversarial Network	-	-	2.8e-2
	Frozen-PINN-swim (our) ❄	**0.05**	1	**1.07e-2 ± 4.52e-4**
	Frozen-PINN-swim (our) ❄	**12.43**	-	**9.99e-5 ± 6.21e-9**
10-d heat	PINN (Adam)	1002.49	3037.8	1.68e-1 ± 3.21e-2
Wang & Dong (2024)	PINN (L-BFGS)	189.6	574.5	6.06e-4 ± 1.00e-4
benchmark extended to $d = 10$	**Frozen-PINN-elm (our)** ❄	**0.33**	1	**4.35e-4 ± 5.91e-5**
	Frozen-PINN-elm (our) ❄	168.6	-	**2.28e-5 ± 2.1e-5**
100-d heat He et al. (2023)	PINN (Adam)	141^+	1084.6^+	0.60e-2
He et al. (2023)	PINN (no stacked-backpropagation)	49.8^+	383.1^+	0.63e-2
	PINN (Adam+L-BFGS)	26.25^+	201.9^+	4.98e-3 ± 2.96e-4
	Frozen-PINN-elm (our) ❄	**0.13**	1	**4.12e-4 ± 1.70e-5**

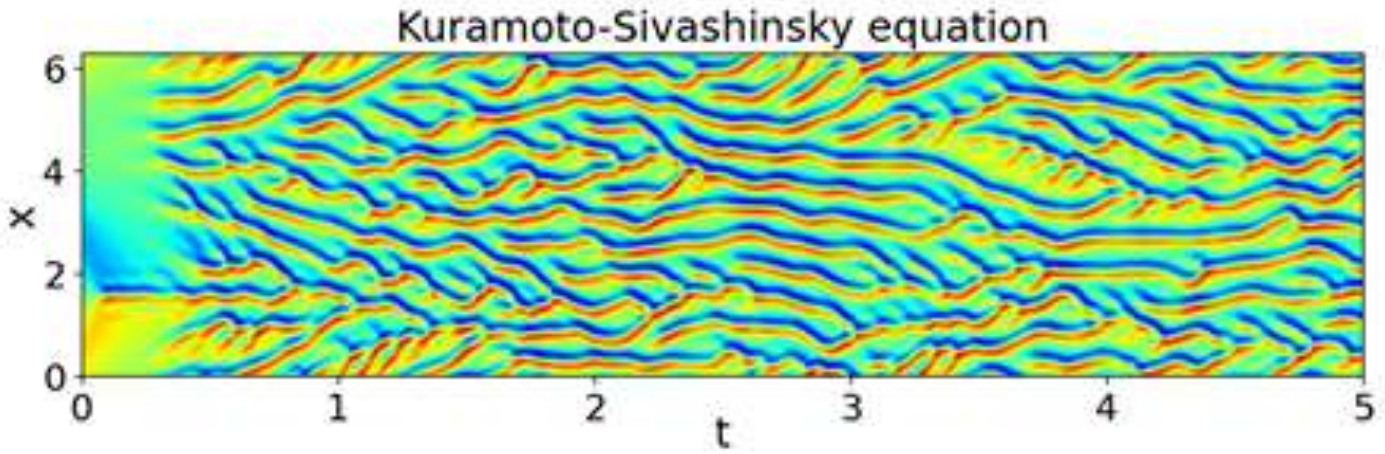


Figure : Frozen-PINN solution of the strongly non linear and chaotic Kuramoto-Sivashinsky equation.

Table : Summary of empirical results for eight PDE benchmarks, including results from previous work: dashes indicate training times not reported in previous work; training times with a + were obtained on a GPU; therefore, CPU-based training, such as Frozen-PINNs, will result in significantly larger values. For each PDE, solvers above/below the horizontal line correspond to low/high accuracy regions. Normalized training time relative to Frozen-PINNs is calculated as the ratio of the training time of each method to that of Frozen-PINNs, computed at comparable accuracy.

Thanks