Boundary-Informed Method of Lines for Physics-Informed Neural Networks

Maximilian Cederholm[*] Siyao Wang[†] Haochun Wang[‡] Ruichen Xu[§] Yuefan Deng[§]

Abstract. We propose a hybrid solver that fuses the dimensionality-reduction strengths of the Method of Lines (MOL) with the flexibility of Physics-Informed Neural Networks (PINNs). Instead of approximating spatial derivatives with fixed finite-difference stencils—whose truncation errors force extremely fine meshes—our method trains a neural network to represent the initial spatial profile and then employs automatic differentiation to obtain spectrally accurate gradients at arbitrary nodes. These high-fidelity derivatives define the right-hand side of the MOL-generated ordinary-differential system, and time integration is replaced with a secondary temporal PINN while spatial accuracy is retained without mesh refinement. The resulting "boundary-informed MOL-PINN" matches or surpasses conventional MOL in accuracy using an order of magnitude fewer collocation points, thereby shrinking memory footprints, lessening dependence on large data sets, and increasing complexity robustness. Because it relies only on automatic differentiation and standard optimizers, the framework extends naturally to linear and nonlinear PDEs in any spatial dimension.

1 Introduction.

Data–driven surrogates that respect physical laws are rapidly reshaping the numerical solution of differential equations. A leading paradigm is the *physics-informed neural network* (PINN), in which the governing equations appear as soft constraints in the loss function so that the network is trained not only on observational data but also on residuals of the underlying operators [6]. By embedding the residual of a partial differential equation (PDE) directly into stochastic gradient descent, PINNs circumvent the need for dense labeled data sets and can, in principle, generalize across regimes that were never explicitly observed. However, in more than one spatial dimension the residual must be evaluated at a large number of collocation points distributed throughout the spatio-temporal domain. Because every additional derivative of the PDE introduces a new set of boundary conditions, the effective hypothesis space grows combinatorially, and the optimizer must navigate an increasingly flat, high-dimensional loss landscape. The upshot is that PINNs sometimes struggle to converge or require prohibitively many collocation points when confronted with stiff, multi-scale, or high-order PDEs.

One classical strategy for taming dimensionality is the *Method of Lines* (MOL) [9]: discretize space, treat time as continuous, and solve the resulting system of ordinary differential equations (ODEs) with robust ODE integrators. Unfortunately, the spatial derivatives in standard MOL are approximated via finite-difference (FD) stencils whose truncation error scales algebraically with the mesh width; achieving high fidelity demands finely resolved grids that inflate memory consumption and computational cost [7]. In addition, FD operators are tied to regular lattices and lose accuracy or even stability on irregular nodesets, limiting their compatibility with the randomly sampled collocation points favored in PINN training.

We propose to combine the dimensional-reduction power of MOL with the differentiability of PINNs while *removing* finite-difference approximations altogether. Our key observation is that if the initial spatial profile is represented by a smooth neural network, then all spatial derivatives can be obtained to machine precision by automatic differentiation (AD) [1]. By training a boundary-informed network solely on the initial slice $t = 0$, we obtain a differentiable surrogate whose gradients serve as spectrally accurate inputs to the MOL ODE system. Time evolution is then handled either by classical integrators or by a secondary PINN that operates purely in the temporal domain, effectively decoupling space and time. The resulting *boundary-informed MOL-PINN* inherits three distinct advantages: (i) high-order spatial accuracy on arbitrary node sets without mesh refinement, (ii) a drastic reduction in the number of collocation points required during training, and (iii) the flexibility of modern AD frameworks, making the method immediately compatible with existing scientific machine-learning

[*]Department of Physics, Stony Brook University.
[†]Department of Statistics, University of California, Davis.
[‡]Department of Applied Mathematics and Statistics, Stony Brook University.
[§]Co-corresponding authors: ruichen.xu@stonybrook.edu, yuefan.deng@stonybrook.edu.

toolchains. These benefits are complementary to recent neural-operator advances that handle symmetries [4], enhance expressive power [2], and diagnose discretization-mismatch errors [3], as well as to structure-preserving Hamiltonian learning that preserves invariants [10]. We demonstrate that the hybrid solver delivers solutions of comparable or superior accuracy to FD-based MOL while using an order of magnitude fewer residual evaluations, thereby lowering memory overhead and accelerating convergence for both linear and nonlinear PDEs in one and multiple dimensions.

2 Preliminaries.

2.1 Method of Lines (MOL). The core idea of MOL is to discretize *all but one* independent variable. For a d-dimensional spatial domain $\Omega \subset \mathbb{R}^d$ and a final time $T > 0$, consider a PDE written in residual form

$$\mathcal{R}[u](\mathbf{x},t) := \frac{\partial^n u}{\partial t^n}(\mathbf{x},t) - \mathcal{S}[u](\mathbf{x},t) = 0, \qquad (\mathbf{x},t) \in \Omega \times (0,T],$$

where $\mathcal{S}$ is the spatial operator and n denotes the highest temporal derivative. Imposing initial data $u(\mathbf{x},0) = u_0(\mathbf{x})$ with suitable boundary conditions closes the problem. The spatial domain is replaced by $\{\mathbf{x}_i\}_{i=1}^{N_x}$. Defining $u_i(t) := u(\mathbf{x}_i,t)$ and introducing a *discrete* approximation $\mathcal{S}_{\text{disc}}$ of $\mathcal{S}$, we obtain an ordinary differential system $\mathrm{d}^n u_i/\mathrm{d}t^n = \mathcal{S}_{\text{disc}}\big[\mathbf{u}(t)\big]_i, i = 1,\ldots,N_x$ and $\mathbf{u}(t) = (u_1(t),\ldots,u_{N_x}(t))^\top$. While MOL cleanly separates spatial and temporal discretizations, its accuracy is bottlenecked by spatial truncation errors [7].

2.2 Neural networks. A *feed-forward neural network* (FFNN) with L hidden layers is a parameterized mapping $u_\theta : \mathbb{R}^m \to \mathbb{R}^p$ defined as a composition of affine transformations and element-wise nonlinearities [5]:

$$u_\theta(x) = f_L \circ f_{L-1} \circ \cdots \circ f_1(x), \quad f_\ell(x) = \sigma\big(W_\ell x + b_\ell\big),\ \ell = 1,\ldots,L,$$

where $W_\ell \in \mathbb{R}^{d_\ell \times d_{\ell-1}}$ and $b_\ell \in \mathbb{R}^{d_\ell}$ are weights and biases, σ is a smooth activation function (e.g. tanh or sin), and $\theta = \{W_\ell, b_\ell\}_{\ell=1}^L$ denotes the collection of trainable parameters. Given data pairs $\{(x_i, u_{\text{data}}(x_i))\}_{i=1}^N$, the network is typically fitted by minimizing a mean-squared error (MSE), $\mathcal{L}_{\text{MSE}}(\theta) = \frac{1}{N}\sum_{i=1}^N \big(u_\theta(x_i) - u_{\text{data}}(x_i)\big)^2$, using stochastic gradient descent or its adaptive variants.

2.3 Physics-Informed Neural Networks (PINNs). PINNs augment data loss with the governing physics by penalizing the PDE residual at *collocation points* [8]. Let $\{(\mathbf{x}_i^{\text{int}}, t_i^{\text{int}})\}_{i=1}^{N_{\text{int}}}$ be interior points and $\{(\mathbf{x}_j^\partial, t_j^\partial)\}_{j=1}^{N_\partial}$ be boundary points. A PINN approximates $u(\mathbf{x},t) \approx u_\theta(\mathbf{x},t)$ while enforcing $\mathcal{R}\big[u_\theta\big](\mathbf{x}_i^{\text{int}}, t_i^{\text{int}}) \approx 0$ and $\mathcal{B}\big[u_\theta\big](\mathbf{x}_j^\partial, t_j^\partial) \approx g_j$. The composite loss

$$\mathcal{L}_{\text{PINN}}(\theta) = \frac{1}{N_{\text{int}}}\sum_{i=1}^{N_{\text{int}}} \big\|\mathcal{R}[u_\theta](\mathbf{x}_i^{\text{int}}, t_i^{\text{int}})\big\|^2 + \lambda_\partial \frac{1}{N_\partial}\sum_{j=1}^{N_\partial} \big\|\mathcal{B}[u_\theta](\mathbf{x}_j^\partial, t_j^\partial) - g_j\big\|^2$$

is minimized with respect to θ. The approach is mesh-agnostic; collocation points can be drawn randomly or adaptively, making PINNs attractive for irregular geometries. Their main drawback is *dimensional curse* [6].

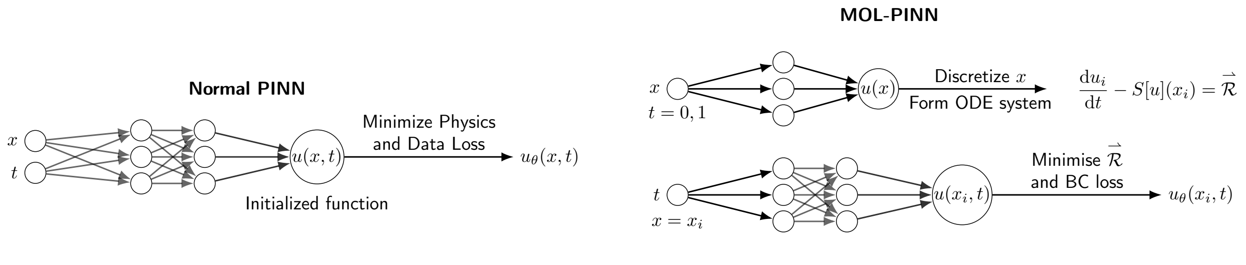

Figure 2.1: Side-by-side comparison of the standard PINN and the MOL-PINN architectures.

2.4 From MOL to Boundary-informed MOL-PINN. To combine the dimensionality reduction capabilities of MOL with the differentiability and mesh flexibility of PINNs—while avoiding finite-difference errors—we propose the following two-stage workflow:

Stage I — Spatial surrogate learning. We restrict to $t = 0$ and train an FFNN to satisfy the *boundary* residual

$$\mathcal{R}_\partial[u_{\theta_\partial}](\mathbf{x}, 0) = \frac{\partial^n u_{\theta_\partial}}{\partial t^n}(\mathbf{x}, 0) - \mathcal{S}[u_{\theta_\partial}](\mathbf{x}, 0) \approx 0, \qquad \{\, \theta_\partial \mid [u_{\theta_\partial}](\mathbf{x}, 0) \approx \mathcal{B}(\mathbf{x}) \,\}$$

together with boundary data. Because time is frozen, this task is equivalent to solving a $(d-1)$-dimensional elliptic problem, which is considerably easier than the full spatio-temporal PDE. Crucially, automatic differentiation (AD) can now evaluate $\mathcal{S}$ *exactly* at any $\mathbf{x}$, producing a spectrally accurate discrete operator $\mathcal{S}_{\mathrm{AD}}$.

Stage II — Temporal evolution in reduced space. The AD-derived operator is frozen, and the PDE reduces to an ODE system

$$\partial_t^n u_i(t) = \mathcal{S}_{\mathrm{AD},i}, i = 1, \ldots, N_x, \qquad \langle \frac{\partial^n u_{\theta_\partial}}{\partial t^n}(\mathbf{x}_i, t) - \mathcal{S}_{AD}[u_{\theta_\partial}](\mathbf{x}_i) \rangle = \vec{\mathcal{R}} \approx 0$$

which can be advanced either with classical solvers or, as in this work, with a compact *temporal* PINN $u_{\theta_t}(t)$ that depends *only* on time. The number of **MOL trajectories** is indexed by i, providing an adjustable granularity that balances accuracy and computational cost, analogous to the role of collocation points in traditional methods.

The resulting *boundary-informed MOL-PINN* inherits three key benefits: (I) **Spectral-like spatial accuracy** without finite-difference errors, thanks to AD on a smooth neural surrogate. (II) **Dimensionality reduction** during temporal integration, leading to faster convergence and higher accuracy (III) **Mesh independence**, enabling arbitrary node placement and straightforward extension to complex geometries and mixed operators.

These properties make the boundary-informed MOL-PINN a compelling candidate for stiff, multi-scale, or data-scarce PDEs, as empirically validated in Section 3.

Table 3.1: Hyper-parameters and hardware.

Model	Epochs	LR
Parameter	8 000	1.0×10^{-3}
Model	Optim	GPU
Parameter	Adam	RTX 4070

Epochs, Learning Rate, Optimizer, and GPU

Table 3.2: MOL-PINN MSE results & comparison.

	PINN	**21i**	**50i**	**100i**
Burgers	7.15×10^{-3}	2.40×10^{-1}	1.27×10^{-2}	2.79×10^{-3}
NS–TG	5.87×10^{-1}	2.23×10^{-2}	2.17×10^{-2}	N/A
NS–ABC	1.169×10^{0}	7.74×10^{-1}	N/A	N/A

Emphasizing the affect of variable i MOL trajectories

3 Results

All tests were run on an RTX 4070, shown in table 3.1, and all results can be found in table 3.2. Recent tests show that MOL-PINN provides an overall unusual benefit—**complexity robustness**. While a PINN's accuracy rapidly deteriorates as the PDEs increase in complexity, MOL-PINN seems to have a slower rate of deterioration. 2D nonlinear PDEs such as Burger's equation approximate at almost the same level of error as the 3D Navier Stokes. MOL-PINN properly began deteriorating at the 4D Navier Stokes, and yet it maintained a higher accuracy than PINN.

Table 3.3: Regional MSE for 3D/4D Navier–Stokes

	v	u	p (TG) / w (ABC)
PINN TG	1.00×10^{-2}	6.58×10^{-3}	1.75×10^{0}
MOL-PINN TG	2.61×10^{-2}	2.35×10^{-2}	1.56×10^{-2}
PINN ABC	1.54×10^{0}	1.15×10^{0}	1.01×10^{0}
MOL-PINN ABC	1.29×10^{0}	6.78×10^{-1}	3.54×10^{-1}

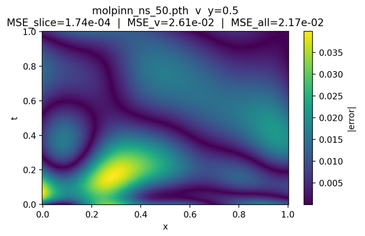

Figure 3.1: 3D Navier-Stokes Error Heatmap (50i v-region)

3.1 N-S Results

We evaluated MOL-PINN on N-S solutions across increasing complexity. On Taylor-Green (TG), results were markedly more consistent than PINN. Adding one dimension to test the Arnold Beltrami Childress (ABC) flow, MOL-PINN still consistently outperformed PINN and maintained similar complexity robustness. We suspect the overall low MSE reflects too few anchor functions; scaling anchors with dimensionality would likely further improve MOL-PINN over PINN.

3.2 Time-based Efficiency Table 3.1 shows that accuracy increases with more systems, but the current MOL–PINN is computationally inefficient: time per epoch also rises with system count. After 8,000 epochs, simulations became prohibitively long, and for this reason no data were collected for the 100-system case on Darcy and Navier-Stokes. Thus, while adding systems improves accuracy, it is accompanied by a significant computational drawback.

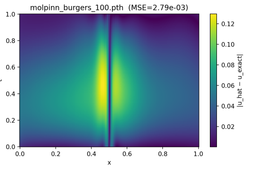

Figure 3.2: Burgers MSE Heatmap (100i)

Table 3.4: Time per epoch for tested simulations

	PINN	21i	50i	100i
Darcy	0.012s	0.046s	0.142s	N/A
Burgers	0.0142s	0.017s	0.046s	1.015s
Navier-Stokes 3D	0.013s	0.278s	0.367s	N/A
Navier-Stokes 4D	0.031s	2.25s	N/A	N/A

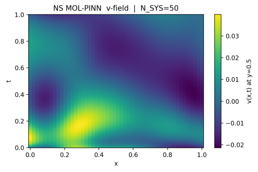

Figure 3.3: N-S Output Heatmap (50i v-region)

4 Conclusion and future work. MOL–PINN leverages automatic-differentiation gradients to achieve high-order spatial accuracy without dense meshes and shows strong resilience to solution complexity. A key limitation is an axis-aligned **directional grain bias** from the line-wise decomposition: continuity is not enforced at intermediary points, so off-grain values——especially those far from anchored boundaries——are harder to learn than in PINN, which can reduce accuracy on lower-complexity PDEs; runtime inefficiencies also persist at larger system counts. We plan to address these issues via two potential future frameworks: a MOL-tailored residual-based adaptive refinement pipeline to target low accuracy regions while minimizing computational necessities, and a Jacobian-enabled multi-directional MOL extension to introduce rotated grains and reduce the grain bias.

References

[1] A. G. Baydin, B. A. Pearlmutter, A. Radul, and J. M. Siskind, *Automatic differentiation in machine learning: A survey*, Journal of Machine Learning Research, 18 (2018), pp. 1–43.

[2] W. Gao, J. Luo, R. Xu, and Y. Liu, *Dynamic schwartz-fourier neural operator for enhanced expressive power*, 2025, https://openreview.net/forum?id=B0E2yjrNb8. Preprint / accepted version.

[3] W. Gao, R. Xu, Y. Deng, and Y. Liu, *Discretization-invariance? on the discretization mismatch errors in neural operators*, in The Thirteenth International Conference on Learning Representations, 2025, https://openreview.net/forum?id=J9FgrqOOni.

[4] W. Gao, R. Xu, H. Wang, and Y. Liu, *Coordinate transform fourier neural operators for symmetries in physical modelings*, 2024 (forthcoming). Accepted; see https://tmlr.org.

[5] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*, Adaptive Computation and Machine Learning, MIT Press, 2016, http://www.deeplearningbook.org.

[6] G. E. Karniadakis, I. G. Kevrekidis, L. Lu, P. Perdikaris, S. Wang, and L. Yang, *Physics-informed machine learning*, Nature Reviews Physics, 3 (2021), pp. 422–440, https://doi.org/10.1038/s42254-021-00049-0.

[7] R. J. LeVeque, *Finite Difference Methods for Ordinary and Partial Differential Equations: Steady-State and Time-Dependent Problems*, Society for Industrial and Applied Mathematics, Philadelphia, 2007, https://doi.org/10.1137/1.9780898717839.

[8] M. Raissi, P. Perdikaris, and G. E. Karniadakis, *Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear pdes*, Journal of Computational Physics, 378 (2019), pp. 686–707, https://doi.org/10.1016/j.jcp.2018.10.045.

[9] W. E. Schiesser, *The Numerical Method of Lines: Integration of Partial Differential Equations*, Academic Press, San Diego, 1991.

[10] Z. Wu, R. Xu, L. Chen, G. Kementzidis, S. Wang, and Y. Deng, *Kolmogorov-arnold representation for symplectic learning: Advancing hamiltonian neural networks*, 2025, https://arxiv.org/abs/2508.19410, https://arxiv.org/abs/2508.19410.