REGRESSIVE DUAL-PINN: FASTER AND MORE RELIABLE TRAINING FOR LONG-HORIZON PDES

Anonymous authors
Paper under double-blind review

ABSTRACT

Physics-informed neural networks (PINNs) use reverse-mode automatic differentiation (AD) to solve partial differential equations (PDEs), and are expected to run fast on advanced AI hardware. In practice, their speed is far below expectations, and reliability over long time horizons is poor, often converging to trivial solutions. Prior fixes, such as finite-difference (FD) schemes and window-based training, either sacrifice accuracy or ignore the relation between successive PINN models. To address these shortcomings, we propose Regressive Dual-PINN (RD-PINN), comprising two components: Dual-FD and regressive initialization. Dual-FD computes exact first-order derivatives via dual numbers and second-order derivatives via FD, significantly reducing pure-FD's approximation error while avoiding AD's costly nested backward passes. Regressive initialization exploits the trajectory of models trained across earlier time windows, using regression to initialize a new window's parameters. Across seven PDE problems and 10 architectures, Dual-FD shows mixed speedups on one-dimensional problems. In 24 paired two-dimensional comparisons, its median speedup is 2.42× and its maximum is 9.68×; peak memory is lower in 21 of the 22 comparisons. Regressive initialization's advantage emerged on five unsteady PDE problems with extended horizons: only RD-PINN consistently sustained correct long-horizon behavior without reference data, while four published window-based and the short-horizon methods degraded as dimensionality and horizon length grew.

1 INTRODUCTION

Consider the two-dimensional Laplace's equation: $\nabla^2\phi = \phi_{xx} + \phi_{yy} = 0$, a classical second-order partial differential equation (PDE) where $\phi(x, y)$ is the solution. To solve it, Physics-Informed Neural Networks (PINNs) use a neural network to approximate the solution and automatic differentiation (AD) to gradually reduce the residual error: $\sum_{(x,y)\in S} (\phi_{xx} + \phi_{yy})^2$, where S is a set of sample points. Although using AD to minimize the residual provides benefits, like enabling the use of advanced AI computing hardware, solving the equations without meshing, and eliminating numerical approximation error in calculating the derivatives, e.g., ϕ_x and ϕ_{xx}, AD computational speed is not appealing, in particular for PDEs with higher-order derivatives. For instance, solving the above Laplace's equation will require PINNs to use AD to compute third derivatives, e.g., $\phi_{xx\theta}$, where θ represents the neural network parameters. As a result, AD produces deeply nested backward passes that each computing process must go through to update the network parameters in each iteration, thus slowing down PINNs' speed. Though some architectural methods and training schemes have been proposed to speed up PINNs, very limited works directly address this problem in AD.

Finite difference (FD) method, which has been extensively employed in numerous solvers, has been reintroduced to PINNs for replacing AD (Sharma & Shankar, 2022; Jiang et al., 2023). Several other methods relying on FD are also proposed (Chiu et al., 2022; Xiang et al., 2025; Mikulski & Uhl, 2026). Finite difference method introduces approximation errors to both first order and second order derivatives. For centered stencils the truncation error is $O(\delta^2)$ at both orders, where δ is the finite-difference step size (App. A.3). However, decreasing the step size also amplifies floating-point round-off, by approximately $1/\delta$ for first derivatives and $1/\delta^2$ for second derivatives. Due to these errors, replacing AD by FD often degrades the accuracy of the solution and sometimes causes convergence difficulties (Chiu et al., 2022; Jiang et al., 2023).

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

Another problem of the classical PINNs is that they cannot reliably solve PDEs with long horizon. They usually minimize the residual term but ignore the initial condition term, causing the convergence to the trivial solution. The problem is that PINNs solve the PDE in different timesteps at the same time. When computing the solution at the ith timestep, the solution at the $i-1$th timestep has not been fully completed. In physical phenomena, e.g., heat diffusion, heat diffuses along the time. The physical events do not happen simultaneously. Classical PINNs do not match physics very well. To address this problem, researchers have proposed window-based methods (Wight & Zhao, 2020; Krishnapriyan et al., 2021; Mattey & Ghosh, 2022) and sampling-based methods (Wight & Zhao, 2020; Daw et al., 2022). The former generally provides more accurate results, but they largely consider only the transition between two adjacent windows, either extending one network over a growing window (Wight & Zhao, 2020), or training a separate network per window using prior predictions as initial conditions (Wight & Zhao, 2020; Krishnapriyan et al., 2021), or retraining a single network while penalizing deviations from the prior solutions (Mattey & Ghosh, 2022). None regresses over the parameter history of multiple completed windows to initialize a new window.

To speed up PINNs and enhance its reliability for solving long horizon PDEs, in this paper, we propose Regressive Dual-PINN (RD-PINN), which comprises two newly developed techniques, Dual-FD and regressive initialization. Dual-FD uses dual number, i.e., $x + a\varepsilon$, where $\varepsilon^2 = 0$ to compute the exact first order derivatives, FD to compute the second order derivatives and AD to update the network parameters. Dual-FD speeds up PINNs significantly because it retains a shallow computational graph, independent of the order of the derivatives of the PDE and reduces approximation error of FD method because its first order derivatives are exact. For enhancing PINNs' reliability to solve long horizon PDEs, we develop regressive initialization, which uses regression over the parameter history of previously completed windows to predict initial network parameters for the next window.

Our contributions are the following.

- Dual-FD combines a forward dual pass for the first derivatives with a single centered difference at second order inside PINN residual training. It retains a shallow computational graph, independent of the order of the derivatives of the PDE, and reduces the round-off amplification of FD from $1/\delta^2$ to $1/\delta$.
- Regressive initialization uses regression over the parameter history of previously completed windows, rather than a single earlier window, to initialize network parameters for the next window. It enhances the reliability of PINNs in solving long horizon PDEs.
- Across seven PDE problems and ten architectures, Dual-FD shows mixed speedups on 1D problems. Across 24 paired 2D comparisons, the median speedup is 2.42× and the maximum is 9.68×, with lower peak memory in 21 of the 22 comparisons. With the horizon extended on five unsteady PDE problems, RD-PINN keeps the long-horizon behavior of the solutions without any data, where four published window-based methods do not.

2 Related Work

PINNs approximate PDE solutions by enforcing governing equations and initial or boundary conditions during training (Raissi et al., 2019). Subsequent frameworks improve representation through modified network units and feature mappings, like QRes (Bu & Karpatne, 2021), PIKAN (Patra et al., 2025), sf-PINN (Wong et al., 2022), IFeF (Aguiar et al., 2026), PIG (Kang et al., 2025), and FPINN (Wu et al., 2025b), or interactions across collocation points, like PINNsFormer (Zhao et al., 2023), PINNMamba (Xu et al., 2025), and ProPINN (Wu et al., 2025a). These frameworks, along with vanilla PINN (StdPINN), form the baselines in our evaluation. Other studies improve training through gradient coordination (Liu et al., 2025; Hwang & Lim, 2024), natural-gradient updates (Schwencke & Furtlehner, 2024), regional residual objectives (Wu et al., 2024), and equality-constrained formulations trained with augmented Lagrangian methods (Basir & Senocak, 2022; Hu et al., 2026). In this study, we focus on residual-differentiation cost across these frameworks and the transfer of learned information over time windows.

Derivative evaluation. Numerical and hybrid differentiation provide alternatives to nested reverse-mode AD. CAN-PINN (Chiu et al., 2022) couples network values and AD derivatives within numerical differentiation formulas; HFD-PINN (Xiang et al., 2025) combines interior finite differences with AD near complex boundaries. These methods differ in their stencils and use of AD, but nei-

ther forms a single AD-FD scheme. On the other hand, DT-PINN (Sharma & Shankar, 2022) and FD-PINN (Jiang et al., 2023) remove AD entirely, forming a single FD-FD scheme with meshless radial-basis-function differences and mesh-based centered differences, respectively, to compute both first- and second-order derivatives. However, FD can introduce step-size-dependent derivative error, requiring calibration to match AD's accuracy (Mikulski & Uhl, 2026). Our Dual-FD instead is a single AD-FD scheme that retains exact first derivative like AD and applies FD only to the second, avoiding first-order error and reducing round-off amplification of second-order error that an FD-FD scheme incurs (Sec. 3.2).

Forward derivative propagation offers another route. SPINN (Cho et al., 2023) combines a separable architecture with forward-mode AD. DOF (Li et al., 2024) directly propagates differential-operator values without constructing the full Hessian, while STDE (Shi et al., 2024) uses random Taylor jets to estimate high-dimensional and high-order operators. Recent explicit kernels propagate only the derivative states required by a particular residual (Cao et al., 2026). In contrast, our Dual-FD combines dual-number forward differentiation for first derivatives with a centered difference for second derivatives, avoiding nested reverse-mode passes without imposing a separable architecture, while retaining reverse-mode parameter gradients (Sec. 3.2).

Temporal training. Sequential PINN training passes the terminal prediction of one interval to the next as an initial condition (Krishnapriyan et al., 2021). Causal weighting prioritizes earlier residuals and can be combined with time marching (Wang et al., 2024). To preserve learned history, bc-PINN (Mattey & Ghosh, 2022) retrains a single network while penalizing deviations from earlier solutions, whereas extrapolation-driven E-DNN (Wang et al., 2025) uses time-dependent parameter corrections that preserve previously learned intervals. Evolutional deep neural networks instead obtain parameter velocities from the PDE and integrate them in physical time (Du & Zaki, 2021). Unlike existing solutions, our framework uses frozen networks from completed windows to supply parameter history for regressive initialization of each new window's network, as well as historical predictions at newly sampled points (Sec. 3.3). Combined with Dual-FD, our framework addresses both residual-evaluation cost and initialization across successive windows.

3 METHOD

3.1 THE SLOWNESS OF AUTOMATIC DIFFERENTIATION

In this subsection, we take a closer look how PINNs use AD to solve PDEs. Consider the Kovasznay flow, a steady-state solution of the incompressible Navier–Stokes equations (NSE), involving two independent variables (x, y) and three output variables u, v, p:

$$u_x + v_y = 0\ , \quad uu_x + vu_y = -p_x + \tfrac{1}{Re}(u_{xx} + u_{yy})\ , \quad uv_x + vv_y = -p_y + \tfrac{1}{Re}(v_{xx} + v_{yy})\ , \tag{1}$$

where x, y represents a spatial coordinate, u, v, p represent the horizontal velocity, vertical velocity, and pressure, respectively, and Re refers to Reynolds number. A standard PINN uses a single neural network taking two input variables (x, y) with three output variables u, v, p to approximate u, v, p. Note that u, v and p are functions of (x, y) and the network parameters, θ.

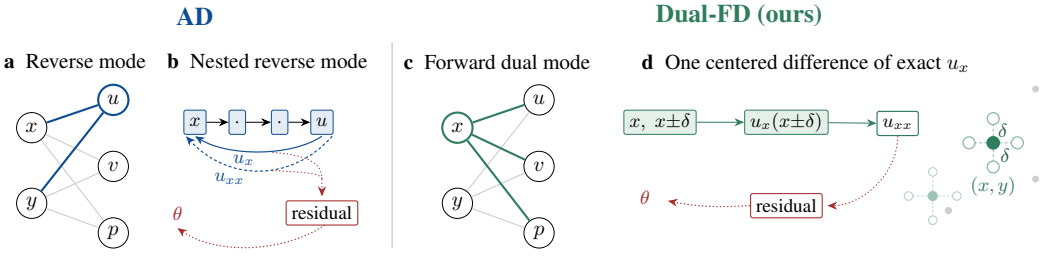


Figure 1: Derivative evaluation for the Kovasznay residual. Dual-FD avoids nested coordinate back-propagation while retaining parameter gradients.

To compute the first-order derivatives, u_x, u_y, v_x, v_y, p_x and p_y required by the Kovasznay problem, AD performs one forward pass and three backpropagations illustrated in Fig. 1a. More clearly, u_x and u_y are obtained the backpropagation starting from u; v_x and v_y are obtained from the backpropa-

gation starting from v, and p_x and p_y are obtained from the backpropagation starting from p. To compute the second order derivatives, u_{xx}, u_{yy}, v_{xx} and v_{yy}, AD carries out additional four backpropagations. Starting from u_x, AD obtains u_{xx} and u_{xy} through a nested graph-differentiation process that is illustrated in Fig. 1b. Even though the Kovasznay problem does not involve u_{xy}, AD computes it as a side product. Similar for u_y, v_x and v_y, AD will provide one side product for each of them. After computing all the first order and second-order derivatives, PINN uses AD again to update the network parameters, which requires computing $u_{x\theta}, u_{y\theta}, v_{x\theta}, v_{y\theta}, p_{x\theta}, p_{y\theta}, u_{xx\theta}, u_{yy\theta}, v_{xx\theta}$, and $v_{yy\theta}$. Although Kovasznay is governed by second-order PDEs, PINN in fact computes third order derivatives, which will produce very deep computational graphs, significantly slowing down PINN speed. To address this issue, our RD-PINN exploits dual numbers and finite difference to retain a shallow computational graph.

3.2 DUAL-FD

To prevent using the deep computational graph as AD and reduce the approximation error of FD method, we propose Dual-FD, which exploits dual number to compute the first-order derivatives, FD to compute the second-order derivatives and AD for parameters' updates.

A dual number can be represented as $a + b\varepsilon$, where $a, b \in \mathbb{R}$ and $\varepsilon^2 = 0$. It is similar to complex number, $a + bi$, but where $i^2 = -1$. Because of the property $\varepsilon^2 = 0$, dual number provides a straightforward approach to compute first-order derivatives (Griewank & Walther, 2008; Baydin et al., 2018). Given $f : \mathbb{D} \to \mathbb{D}$, where $\mathbb{D}$ represents the space of dual number, we have the following equality:

$$f(a + b\varepsilon) = f(a) + f'(a)\, b\, \varepsilon \,. \tag{2}$$

By removing $b\varepsilon$ from the output of the dual part, we can obtain the first-order derivatives. For the sake of clear presentation, we let the neural network approximating the PDE's solution be a function $\psi : \mathbb{D}^m \to \mathbb{D}^n$. For the Kovasznay problem, the input dimension, m, is two, i.e., (x, y) and output dimension, n, is three, i.e., (u, v, p). In many cases, PINN is applied to problems with small input dimensions. To compute all the first-order derivatives of u, v and p of the Kovasznay problem, the proposed Dual-FD uses two forward passes, as shown in Fig. 1c; unlike in Fig. 1a, no backpropagation is needed. To compute (u_x, v_x, p_x) and (u_y, v_y, p_y), Dual-FD respectively takes $(x + b\varepsilon, y)$ and $(x, y + b\varepsilon)$ as inputs. We would like to highlight that the first-order derivatives are exact, unlike FD methods that would introduce approximation error.

To avoid providing deep computational graphs, Dual-FD uses central difference to compute the required second-order derivatives:

$$u_{xx}(x, y) = \frac{u_x(x + \delta, y) - u_x(x - \delta, y)}{2\delta} + O(\delta^2) \,, \tag{3}$$

where δ is the finite-difference step size illustrated in Fig. 1d by a floating stencil (open circles) over each sample point (solid circles). The order here is set by the stencil and not by the exactness of u_x, with route-dependent round-off behavior (App. A.3). After computing all the second-order derivatives, Dual-FD uses AD to compute the gradient of the parameters and update them.

For the Kovasznay problem, FD introduces approximation errors to all ten derivatives terms, but Dual-FD only introduces approximation errors in the four second-order derivatives. Thus, it is expected that Dual-FD is faster than AD and reduces accuracy loss of FD (App. A.4).

3.3 REGRESSIVE INITIALIZATION OVER FROZEN TIME WINDOWS

Dual-FD is proposed to replace AD in evaluating the residual derivatives and speed up PINNs. However, it alone cannot solve PDE problems with long horizon. As discussed in Sec. 1, existing window-based methods do not regress over the parameter history of multiple completed windows. Like these methods, we had to bootstrap with too little history to regress at first: the first window is trained from scratch, the second and third copy the previous window's model, and the fourth uses regression with fixed placeholder coefficients (Eq. 4). From the fifth window onward, we instead fit the increment trend across all completed windows below, exploiting the intrinsic relationship between models in adjacent windows, which are governed by the same PDE even though approximating solutions at different times.

To make use of the sequence of models and enhance PINNs performance for long horizon problem, in this paper, we propose regressive initialization. We divide the total time T into n windows. For the first window, we train a PINN model from scratch. For the second window, we copy the trained model from the first step and use Dual-FD to train it. The same process is repeated up to the third window. From the fourth window onward, regression is used to initialize the PINN model.

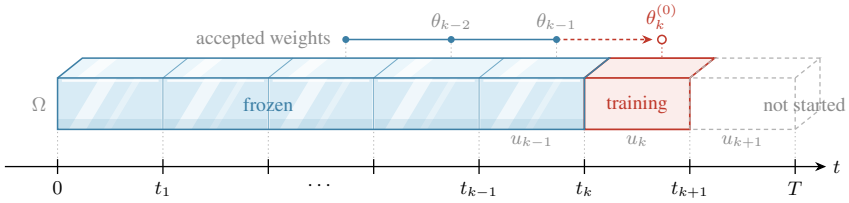


Figure 2: Time-window progression with regressive initialization. Frozen networks supply past solution values and parameter increments for the next initialization.

Let θ_k denote the parameters obtained after training window k, and let $d_k = \theta_k - \theta_{k-1}$ for $k \geq 2$. We approximate the parameter increment and initialize the next model as:

$$d_{k+1} \approx a_k d_k + b_k d_{k-1} \,, \qquad \theta_{k+1}^{(0)} = \theta_k + a_k d_k + b_k d_{k-1} \,, \tag{4}$$

where $a_k, b_k \in \mathbb{R}$. We use model parameters obtained from the previous windows and the least-squares method to determine a_k and b_k:

$$(a_k, b_k) = \arg\min_{a,b} \sum_{j=3}^{k-1} \| a d_j + b d_{j-1} - d_{j+1} \|_2^2 \,. \tag{5}$$

The fourth window uses $a = 1$, $b = 0$; from the fifth window onward, a complete historical increment triplet is available for fitting. The regression estimates only two scalars. We set the partial derivatives of the objective in Eq. 5 with respect to a and b to zero, obtaining a 2×2 linear system determined by five sums of inner products between increments. When nonsingular, this system has the closed-form solution in Eq. 14 (App. A.5). The sums are updated incrementally, so regression requires only the current parameters, the two latest increments, and five scalars. To train the initialized model in window $k+1$, we sample points at the final time of window k and evaluate the trained model θ_k at these points. Its outputs provide the initial condition for the new window.

4 Experiments

In this section, we first compare AD with Dual-FD in terms of speed. We evaluate them on ten PINN methods, including StdPINN (Raissi et al., 2019), QRes (Bu & Karpatne, 2021), sf-PINN (Wong et al., 2022), PIKAN (Patra et al., 2025), PINNsFormer (Zhao et al., 2023), PINNMamba (Xu et al., 2025), ProPINN (Wu et al., 2025a), FPINN (Wu et al., 2025b), PIG (Kang et al., 2025) and IFeF (Aguiar et al., 2026), and seven PDE problems listed in Table 1, where two of them are first order and five of them are second order. Their formulas are given in App. B.2. These problems are employed in previous PINN studies (Raissi et al., 2019; Krishnapriyan et al., 2021). We measure the wall-clock time until the relative L_2 error first falls below a predefined threshold (App. B.2). For each evaluation, we perform three runs with different random seeds.

To understand where Dual-FD obtains the speedup, we analyse the time taken in the forward pass, in the computation of first- and second-order derivatives, and in backpropagation with parameter updates in one iteration. In this analysis, we also include FD+FD and AD+FD, where FD+FD uses FD to compute the first- and second-order derivatives and AD+FD uses AD to compute the first-order derivatives and FD to compute the second-order derivatives. In addition to speed, we also compare AD and Dual-FD on memory consumption and the accuracy of their solutions (App. C.4).

To evaluate the proposed RD-PINN for solving long-horizon PDEs, we compare it with temporal training methods, including Seq-to-seq (Krishnapriyan et al., 2021), Causal weighting (Wang et al., 2024) and bc-PINN (Mattey & Ghosh, 2022) on five PDE problems involving temporal derivatives. These problems are also employed in previous temporal training studies (Penwarden et al., 2023).

4.1 SPEED COMPARISON WITH MATCHED ACCURACY

We report the AD/Dual-FD time-to-threshold ratio, with values above one indicating faster convergence for Dual-FD. Table 1 summarizes results across seven PDE problems and ten PINN methods, where n_2 denotes the number of second-order derivative terms in each equation. The three two-dimensional problems are Kovasznay, Burgers 2D, and Taylor–Green. Results are not accepted if, even after threshold relaxation, failure to converge leaves large relative L_2 errors and the predicted fields fail to recover the physical features. If neither method produces an acceptable result, the comparison is marked n/c. If AD can reach the threshold but Dual-FD cannot, it is denoted as AD; vice versa, it is denoted as Dual. Each problem–architecture comparison in Table 1 uses three random seeds. If both operators fail to reach the original threshold for a seed, they are compared at a relaxed common threshold, subject to the physical-field checks in App. B.2. A star (*) marks a comparison in which at least one of the three seeds uses a relaxed threshold; it does not indicate fewer seeds. In total, eleven results are obtained from the relaxed thresholds and 37 results from the original thresholds. Every entry of Table 1 that carries a number enters the statistics.

Table 1: Speed comparison at matched accuracy: AD/Dual-FD time-to-threshold ratios from three random seeds. *: at least one seed uses a relaxed threshold. AD, Dual and n/c are defined in Sec. 4.1.

	n_2	StdPINN	QRes	sf-PINN	PIKAN	PINNsF.	PMamba	ProPINN	FPINN	PIG	IFeF	Med.
Reaction	0	0.94*	1.18	*n/c*	*n/c*	1.15	1.10*	0.98*	1.27*	*AD*	1.60	1.15
Convection	0	*Dual*	*n/c*	*n/c*	*n/c*	*AD*	*n/c*	0.97*	*n/c*	*n/c*	*n/c*	0.97
Burgers 1D	1	0.86	1.52	0.91	0.64	1.29	1.70	1.07	1.01	2.08	12.92*	1.18
Schrödinger	2	1.99	2.21*	1.50*	*n/c*	5.03*	*AD*	2.23*	2.03*	9.37	*n/c*	2.21
Kovasznay	4	1.81	3.20	1.84	1.41	3.29	4.31*	1.94	2.08	9.68	1.53*	2.01
Burgers 2D	4	1.16	2.53*	1.08	*n/c*	*n/c*	3.39*	3.03*	1.91	*n/c*	*n/c*	2.22
Taylor–Green	4	2.22	2.78*	1.81	*n/c*	3.26*	3.93*	2.72*	2.31*	*n/c*	3.40*	2.75

The row-wise medians in Table 1 indicate that Dual-FD is faster than AD in general for all problems, except for Convection, for which there is only one valid ratio, 0.97. The speedup tends to be more pronounced for problems with more second-order derivative terms (App. C.2). Across the three two-dimensional problems, Dual-FD is faster in all 24 comparisons. The median speedup among these 24 comparisons is $2.42\times$, with a range of 1.08–$9.68\times$. The median speedups for Kovasznay, Burgers 2D and Taylor–Green are $2.01\times$, $\mathbf{2.22}\times$ and $2.75\times$, respectively. Restricting to the 16 two-dimensional comparisons that pair at the original thresholds, Dual-FD's median speedup is $2.01\times$ over the same range of 1.08–$9.68\times$. These results show that when the problems are more complex, for example two-dimensional and second order, Dual-FD tends to perform better.

To study how network width and depth affect Dual-FD's speedup, we evaluate StdPINN on Kovasznay. All other settings follow Table 1 (Fig. 3b). Across widths 16 to 256, speedup ranges from $1.75\times$ to $2.41\times$, with no monotonic trend. Across the tested depths, speedup rises from $1.90\times$ at 5 layers to $2.48\times$ at 17 layers and then flattens, reaching $2.47\times$ at 25 layers.

We also compare peak GPU memory usage between Dual-FD and AD (Table 11 in App. C.4). Among the 22 two-dimensional comparisons, Dual-FD uses less peak memory in 21, with Dual-FD/AD ratios of 0.17–0.84 and a median of 0.62. The exception is PIKAN on Kovasznay, at 2.18. Across all 47 comparisons of the matrix measured this way, the median is 0.65. The two remaining two-dimensional comparisons are ProPINN on Burgers 2D and Taylor–Green, whose AD run does not fit on the 20 GB card when the whole residual batch is evaluated at once and is therefore run on 512 collocation points at a time while the Dual-FD run takes the batch whole. Their peaks are not measured at the same batch size and are left out, and their speed comparisons remain in Table 1.

Finally, reaching the same error threshold does not imply equal final solution errors (Fig. 3a). The accuracy summary covers 49 paired comparisons (147 seed pairs). Statistics use final checkpoints and each comparison's median of three seed-wise ratios. Fig. 3a plots the median final error over three seeds for AD and Dual-FD separately. The median Dual-FD/AD ratio is 1.07 over the 49 comparisons and 1.13 over the 38 comparisons paired at the original thresholds; 46 of the 49 agree within an order of magnitude, and Dual-FD has the lower error in 18 comparisons. The largest discrepancies include sf-PINN and QRes on Schrödinger, where the reported Dual-FD errors are about an order of magnitude higher than AD. Both comparisons use a relaxed threshold for at least

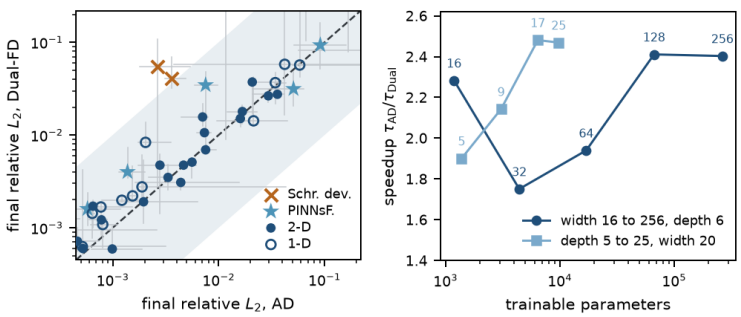


Figure 3: (a) Final L_2: three-seed medians and ranges; shading: Dual-FD/AD ratios of 0.1–10. (b) Kovasznay StdPINN scans: width, three-seed medians; depth, one seed per size.

one seed, underscoring the distinction between faster threshold attainment and comparable solution accuracy. Using the best checkpoints instead of the final checkpoints gives median error ratios of 1.05 over all 49 comparisons and 1.08 over the 38 comparisons at the original thresholds. Taking the ratio of the two medians instead of the median seed-wise ratio gives 1.35 and 1.36. Comparisons without a common reached threshold are retained in Table 1 but excluded from these paired statistics. Physical-field comparisons complement these statistics (App. D.5).

These experiments assess efficiency and accuracy with one network covering the solution domain. Sec. 4.3 extends the time interval to test whether correct dynamics persist over long horizons.

4.2 Where the per-step speedup comes from

The previous experiments have shown that Dual-FD performs better than AD in general, in particular on more complex problems. In this experiment, we investigate where Dual-FD gains the speed. We separately record the time taken in the forward pass, the first-order derivatives, the second-order derivatives and the parameter updates in each iteration. In addition to AD+AD and Dual-FD, for problems with second-order derivatives, we also compare with AD+FD and FD+FD. Due to space constraints, Fig. 4 shows results for five selected problems evaluated on StdPINN at width 32, where one is first order and the others have one to four second-order terms. The results for the other problems employed in Table 1 can be found in App. C.1.

Compared with AD+AD, which uses AD to compute both first- and second-order derivatives, Fig. 4 shows that Dual-FD's per-iteration advantage depends on the order and the number of second-order terms in the equations. For Convection, which has no second-order terms, Dual-FD takes slightly longer than AD. On Burgers 1D with one second-order term, total times are similar, but Dual-FD remains slightly slower. On Schrödinger and the two two-dimensional problems with multiple second-order terms, Dual-FD has a much lower total computation cost than AD+AD. Fig. 4 reveals that the time savings come from the computation of the second-order derivatives and the backpropagation for parameter updates, where the saving is the largest. When AD+AD performs the backpropagation, it must traverse a high-order computation graph. Dual-FD traverses a much simpler graph. As a result, it gains a significant speedup.

Compared with AD+FD, Dual-FD's speedup is still significant. When computing the backpropagation for parameter updates, AD+FD's computation graph is one order lower than that of AD+AD. However, each second-order derivative requires first-order derivatives evaluated at two shifted inputs. As a result, when performing backpropagation, it traverses more recorded graphs and requires more computation time. Although FD+FD is the fastest per iteration, a step time alone does not decide time-to-threshold, because it says nothing about the accuracy a route attains or the iterations it needs. On a tested subset, the median final L_2 against AD is $6.6\times$ for FD+FD and $1.14\times$ for Dual-FD, and of the seven comparisons where AD reaches a threshold FD+FD reaches none in three, while Dual-FD reaches one in all seven (App. C.3).

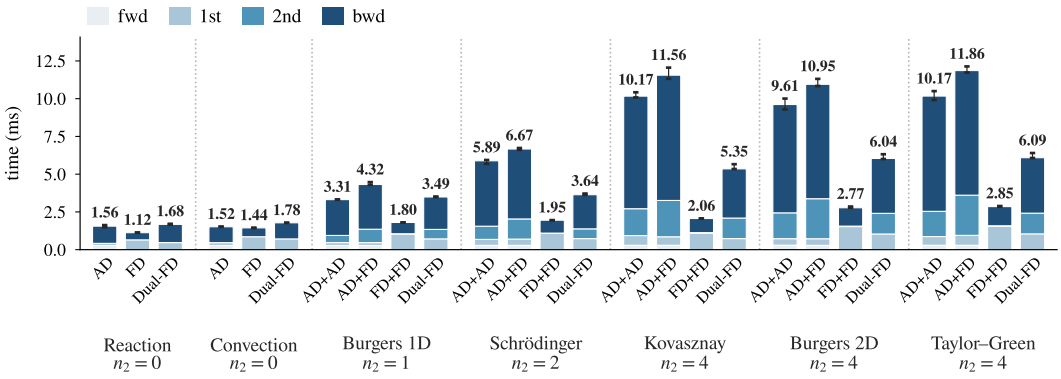


Figure 4: Per-step timing for a width-32 StdPINN, medians over 250 measured steps with interquartile ranges.

4.3 RD-PINN TIME-WINDOW SOLVING OVER LONG HORIZONS

Finally, we compare RD-PINN with existing temporal training methods and whole-horizon networks across five time-dependent PDE problems (Table 2; App. E.2). We assess its ability to maintain field accuracy and physical fluctuations over extended time intervals, without reference training data. Relative L_2 error is normalized by the norm of the full reference field. When the background dominates local disturbances, losing those disturbances can therefore leave a small relative error. We additionally report a physical index Φ, the predicted-to-reference fluctuation-magnitude ratio. For Burgers 2D and the cylinder, we record velocity at a fixed location and define $\Phi = 100\% \times A_{\mathrm{pred}}/A_{\mathrm{ref}}$, where A is the fluctuation amplitude of the velocity signal. For the other problems, A is the spatial standard deviation of each snapshot, which removes the uniform background, and the ratio $A_{\mathrm{pred}}/A_{\mathrm{ref}}$ is averaged over the second half of the horizon. A value of $\Phi = 100\%$ indicates matching magnitudes, while higher or lower values indicate overestimation or underestimation. We assess solution quality using both Φ and L_2.

We perform paired AD and Dual-FD tests across eight frameworks (Table 2; App. E.2). Dual-FD reduces runtime in all 40 problem–framework combinations, with an arithmetic mean speedup of approximately $2.30\times$. Across the 20 comparisons for the four pairable temporal training frameworks, including RD-PINN, the mean is approximately $2.25\times$. EDNN advances network parameters using a time derivative obtained by least-squares projection. It has no corresponding AD/Dual-FD substitution in residual training and is therefore excluded from the paired speedup statistics. These results show that Dual-FD reduces computational cost under the tested training settings.

Together, field error and the physical index show RD-PINN's main advantage on the three two-dimensional problems, where its fluctuation magnitudes are closer to the reference than those of the compared temporal methods. On Taylor–Green, whose vortices decay over time, RD-PINN achieves the lowest mean field error of 4.0% with $\Phi = 99.0\%$, preserving both field accuracy and spatial variation. Burgers 2D involves local transport and diffusion over a large background, so whole-field error can mask underestimated disturbances. Although Seq-to-seq has a slightly lower mean error than RD-PINN (1.9% versus 2.4%), its physical index is only 63.1%, compared with RD-PINN's 95.9%. The cylinder requires sustained wake oscillations to develop from an analytic initial state. For the cylinder, RD-PINN adopts the residual setting of PECANN (Hu & Senocak, 2025). Fig. 5(a) shows an approximately symmetric low-speed wake at $t = 10$, followed by pronounced asymmetry at $t = 30$ and 50. Fig. 5(b) records the transverse velocity at a fixed location: initially weak fluctuations grow into alternating positive and negative oscillations, consistent with the loss of wake symmetry. The late-time prediction and phase-aligned DNS signal show similar oscillatory patterns. Together with the lowest mean field error of 14.2% and $\Phi = 79.7\%$, these observations show that RD-PINN recovers substantial unsteady wake fluctuations without reference training data. On dissipative Burgers 1D, its mean error of 2.0% and $\Phi = 101.4\%$ indicate low error and a fluctuation magnitude close to the reference. On dispersive Schrödinger, RD-PINN also recovers a spatial fluctuation magnitude close to the reference ($\Phi = 101.5\%$).

Table 2: Long-horizon solution accuracy, physical-feature recovery and runtime speedup. Paired entries are ordered Dual-FD / AD. Whole-horizon results are in Table 16 (App. E.2). EDNN has no AD/Dual-FD substitution; its computational cost is reported as runtime without a paired speedup.

Framework / scheme	Metric	Burgers 1D (ν=0.1) $T = 2$	Burgers 2D $T = 8$	Taylor–Green $T = 20$	Schrödinger $T = 1.5$	Cylinder $T = 50$
Seq-to-seq	Mean L_2 (%)	4.8 / 3.9	**1.9** / 1.9	77.9 / 81.9	6.6 / 4.5	36.1 / 37.1
(Krishnapriyan et al., 2021)	Final L_2 (%)	11.0 / 8.0	1.9 / 1.6	97.0 / 103.8	17.8 / 12.0	36.2 / 36.7
	Wall clock (s)	245 / 305	1015 / 2161	**422** / 823	239 / 420	**660** / 1305
	Speedup S	1.25	2.14	1.96	1.76	1.98
	Physics Φ	100.8 / 100.1	63.1 / 58.7	48.0 / 47.2	100.6 / 100.3	2.7 / 1.5
Causal weighting	Mean L_2 (%)	3.6 / 4.2	2.5 / 2.1	97.4 / 91.9	**4.4** / 4.5	36.2 / 37.1
(Wang et al., 2024)	Final L_2 (%)	7.1 / 8.7	2.9 / 2.0	109.7 / 107.3	**6.8** / 9.3	35.6 / 37.6
	Wall clock (s)	941 / 1943	10663 / 37441	1669 / 6432	3232 / 8019	2516 / 10783
	Speedup S	2.07	**3.52**	**3.85**	2.48	**4.29**
	Physics Φ	**99.7** / 100.3	46.2 / 77.2	45.8 / 46.6	**100.0** / 99.9	2.6 / 0.7
bc-PINN	Mean L_2 (%)	9.9 / 6.8	3.1 / 2.7	114.7 / 154.9	8.2 / 7.8	70.3 / 68.9
(Mattey & Ghosh, 2022)	Final L_2 (%)	29.8 / 18.7	2.7 / 1.9	180.9 / 254.1	19.8 / 17.5	70.8 / 70.4
	Wall clock (s)	2189 / 2478	2413 / 2654	1561 / 1621	1860 / 2318	3118 / 3554
	Speedup S	1.13	1.10	1.04	1.25	1.14
	Physics Φ	104.1 / 106.5	12.1 / 6.3	45.6 / 43.2	99.2 / 99.4	0.2 / 0.2
EDNN*	Mean L_2 (%)	3.6	1.9×10^{7}	110.3	137.1	71.3
(Du & Zaki, 2021)	Final L_2 (%)	8.5	9.9×10^{5}	132.3	137.1	109.9
	Wall clock (s)	1855	23717	6581	5889	6195
	Physics Φ	103.5	0.0	73.2	147.8	19.9
RD-PINN	Mean L_2 (%)	**2.0** / 2.3	2.4 / 2.7	**4.0** / 4.4	32.4 / 31.1	**14.2** / 14.5
	Final L_2 (%)	**1.7** / 1.9	**1.6** / 2.0	**7.1** / 8.1	60.3 / 58.7	**16.9** / 18.6
	Wall clock (s)	**214** / 502	**192** / 639	1434 / 3401	**211** / 549	3271 / 10745
	Speedup S	**2.35**	3.33	2.37	**2.61**	3.29
	Physics Φ	101.4 / 102.0	**95.9** / 94.1	**99.0** / 99.0	101.5 / 101.9	**79.7** / 93.4

Bold marks the best value per problem and metric, using Dual-FD for paired methods: lowest errors and runtime, highest S, and Φ closest to 100%. Mean/final L_2 are time-averaged/final relative errors (%). Burgers 2D uses $q = u$ at $(1, 1)$; the cylinder uses $q = v$ at $(2, 0)$ over the final ten time units, with the DNS signal phase-aligned and sampled at the same times. For paired AD/Dual-FD methods, wall clock and S are three-seed means of runtime (s) and paired AD/Dual-FD runtime ratios, respectively. *EDNN marches its parameters by least squares instead of training a residual, so no operator swap applies. Full metric definitions and capacity rules: App. E.

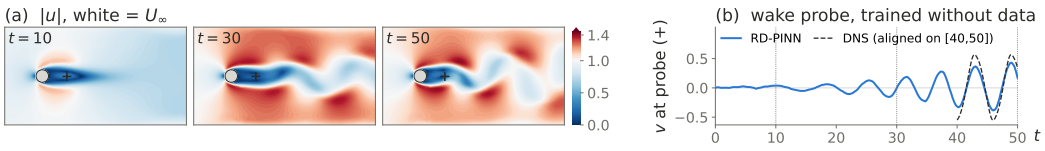


Figure 5: RD-PINN cylinder wake at $Re = 100$ without reference training data. (a) Speed magnitude at $t = 10$, 30 and 50. (b) Transverse velocity at $(2, 0)$, showing the emergence of wake oscillations; the dashed curve is the phase-aligned reference.

RD-PINN improves physical-fluctuation recovery on the tested two-dimensional problems, while Dual-FD further reduces the runtime of temporal training.

5 Conclusion

In this paper, we present RD-PINN, combining Dual-FD with regressive initialization over frozen time windows. Dual-FD computes exact first derivatives using dual numbers and approximates second derivatives using finite differences, avoiding nested coordinate backpropagation. It reduces time to threshold in all tested two-dimensional configurations where both operators reach a common error threshold, and reduces peak memory in 21 of the 22 two-dimensional tests. For long-horizon problems, RD-PINN improves physical-fluctuation recovery on the tested two-dimensional problems without reference training data, while Dual-FD further reduces runtime within temporal training frameworks.

AI USE STATEMENT

Generative AI tools were used to aid and polish the writing of this paper. We have reviewed all AI-assisted text and take responsibility for the final content.

REPRODUCIBILITY STATEMENT

Code and configuration files will be released upon publication. The experimental settings are given in Apps. B and E.1, and the detailed training records, including model capacities, reached thresholds and wall-clock times, are given in App. D. The dual propagation rules and the validation of the operator against autograd are given in Apps. A.1 and A.2.

REFERENCES

Miguel Aguiar, Karl H Johansson, Matthieu Barreau, et al. Iterative training of physics-informed neural networks with fourier-enhanced features. In *International Conference on Learning Representations*, volume 2026, pp. 111164–111176, 2026.

Shamsulhaq Basir and Inanc Senocak. Physics and equality constrained artificial neural networks: Application to forward and inverse problems with multi-fidelity data fusion. *Journal of Computational Physics*, 463:111301, 2022.

Atılım Güneş Baydin, Barak A. Pearlmutter, Alexey Andreyevich Radul, and Jeffrey Mark Siskind. Automatic differentiation in machine learning: a survey. *Journal of Machine Learning Research*, 18(153):1–43, 2018.

Jesse Bettencourt, Matthew J Johnson, and David Duvenaud. Taylor-mode automatic differentiation for higher-order derivatives in JAX. In *Program transformations for ML workshop at NeurIPS 2019*, 2019.

Jie Bu and Anuj Karpatne. Quadratic residual networks: A new class of neural networks for solving forward and inverse problems in physics involving pdes. In *Proceedings of the 2021 SIAM International Conference on Data Mining (SDM)*, pp. 675–683. SIAM, 2021.

Wenbo Cao, Zhe Lu, and Weiwei Zhang. Verified residual-specific explicit derivative kernels for physics-informed learning and discretized PDE adjoints. *arXiv preprint arXiv:2606.29702*, 2026.

Pao-Hsiung Chiu, Jian Cheng Wong, Chinchun Ooi, My Ha Dao, and Yew-Soon Ong. Can-pinn: A fast physics-informed neural network based on coupled-automatic–numerical differentiation method. *Computer Methods in Applied Mechanics and Engineering*, 395:114909, 2022.

Junwoo Cho, Seungtae Nam, Hyunmo Yang, Seok-Bae Yun, Youngjoon Hong, and Eunbyung Park. Separable physics-informed neural networks. volume 36, pp. 23761–23788, 2023.

Felix Dangel, Tim Siebert, Marius Zeinhofer, and Andrea Walther. Collapsing Taylor mode automatic differentiation. volume 38, pp. 147127–147166, 2026.

Arka Daw, Jie Bu, Sifan Wang, Paris Perdikaris, and Anuj Karpatne. Mitigating propagation failures in physics-informed neural networks using retain-resample-release (R3) sampling. 2022.

Yifan Du and Tamer A Zaki. Evolutional deep neural network. *Physical Review E*, 104(4):045303, 2021.

Jeffrey Fike and Juan Alonso. The development of hyper-dual numbers for exact second-derivative calculations. In *49th AIAA aerospace sciences meeting including the new horizons forum and aerospace exposition*, pp. 886, 2011.

Andreas Griewank and Andrea Walther. *Evaluating derivatives: principles and techniques of algorithmic differentiation*. SIAM, 2008.

Qifeng Hu and Inanc Senocak. Unsupervised simulation of incompressible flows with physics-and equality-constrained artificial neural networks. *arXiv preprint arXiv:2511.18820*, 2025.

Qifeng Hu, Shamsulhaq Basir, and Inanc Senocak. Conditionally adaptive augmented Lagrangian method for physics-informed learning of forward and inverse problems. *Computer Methods in Applied Mechanics and Engineering*, 456:118953, 2026.

Youngsik Hwang and Dong-Young Lim. Dual cone gradient descent for training physics-informed neural networks. volume 37, pp. 98563–98595, 2024.

Qinghua Jiang, Chang Shu, Lailai Zhu, Liming Yang, Yangyang Liu, and Zhilang Zhang. Applications of finite difference-based physics-informed neural networks to steady incompressible isothermal and thermal flows. *International Journal for Numerical Methods in Fluids*, 95(10): 1565–1597, 2023.

Namgyu Kang, Jaemin Oh, Youngjoon Hong, and Eunbyung Park. Pig: Physics-informed gaussians as adaptive parametric mesh representations. In *International Conference on Learning Representations*, volume 2025, pp. 73022–73046, 2025.

Aditi Krishnapriyan, Amir Gholami, Shandian Zhe, Robert Kirby, and Michael Mahoney. Characterizing possible failure modes in physics-informed neural networks. volume 34, pp. 26548–26560, 2021.

Ruichen Li, Chuwei Wang, Haotian Ye, Di He, and Liwei Wang. Dof: Accelerating high-order differential operators with forward propagation. *arXiv preprint arXiv:2402.09730*, 2024.

Qiang Liu, Mengyu Chu, and Nils Thuerey. Config: Towards conflict-free training of physics informed neural networks. In *International Conference on Learning Representations*, volume 2025, pp. 59531–59566, 2025.

Revanth Mattey and Susanta Ghosh. A novel sequential method to train physics informed neural networks for Allen Cahn and Cahn Hilliard equations. *Computer Methods in Applied Mechanics and Engineering*, 390:114474, 2022.

Maciej J Mikulski and Tadeusz Uhl. Derivative computation in PINNs: Automatic differentiation, finite differences and beyond. *arXiv preprint arXiv:2608.11020*, 2026.

Subhajit Patra, Sonali Panda, Bikram Keshari Parida, Mahima Arya, Kurt Jacobs, Denys I Bondar, and Abhijit Sen. Physics informed Kolmogorov-Arnold neural networks for dynamical analysis via efficient-KAN and wav-KAN. *Journal of Machine Learning Research*, 26(233):1–39, 2025.

Barak A Pearlmutter. Fast exact multiplication by the Hessian. *Neural computation*, 6(1):147–160, 1994.

Michael Penwarden, Ameya D Jagtap, Shandian Zhe, George Em Karniadakis, and Robert M Kirby. A unified scalable framework for causal sweeping strategies for physics-informed neural networks (PINNs) and their temporal decompositions. *Journal of Computational Physics*, 493:112464, 2023.

Nasim Rahaman, Aristide Baratin, Devansh Arpit, Felix Draxler, Min Lin, Fred Hamprecht, Yoshua Bengio, and Aaron Courville. On the spectral bias of neural networks. In *International conference on machine learning*, pp. 5301–5310. PMLR, 2019.

Maziar Raissi, Paris Perdikaris, and George E Karniadakis. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *Journal of Computational physics*, 378:686–707, 2019.

Franz M Rohrhofer, Stefan Posch, Clemens Gößnitzer, and Bernhard C Geiger. On the role of fixed points of dynamical systems in training physics-informed neural networks. *arXiv preprint arXiv:2203.13648*, 2022.

Nilo Schwencke and Cyril Furtlehner. Anagram: A natural gradient relative to adapted model for efficient pinns learning. 2024.

Ramansh Sharma and Varun Shankar. Accelerated training of physics-informed neural networks (PINNs) using meshless discretizations. *Advances in neural information processing systems*, 35: 1034–1046, 2022.

Zekun Shi, Zheyuan Hu, Min Lin, and Kenji Kawaguchi. Stochastic Taylor derivative estimator: Efficient amortization for arbitrary differential operators. volume 37, pp. 122316–122353, 2024.

Sifan Wang, Xinling Yu, and Paris Perdikaris. When and why pinns fail to train: A neural tangent kernel perspective. *Journal of Computational Physics*, 449:110768, 2022.

Sifan Wang, Shyam Sankaran, and Paris Perdikaris. Respecting causality for training physics-informed neural networks. *Computer Methods in Applied Mechanics and Engineering*, 421: 116813, 2024.

Yong Wang, Yanzhong Yao, and Zhiming Gao. An extrapolation-driven network architecture for physics-informed deep learning. *Neural Networks*, 183:106998, 2025.

Colby L Wight and Jia Zhao. Solving Allen–Cahn and Cahn–Hilliard equations using the adaptive physics informed neural networks. *arXiv preprint arXiv:2007.04542*, 2020.

Jian Cheng Wong, Chin Chun Ooi, Abhishek Gupta, and Yew-Soon Ong. Learning in sinusoidal spaces with physics-informed neural networks. *IEEE Transactions on Artificial Intelligence*, 5 (3):985–1000, 2022.

Haixu Wu, Huakun Luo, Yuezhou Ma, Jianmin Wang, and Mingsheng Long. Ropinn: Region optimized physics-informed neural networks. volume 37, pp. 110494–110532, 2024.

Haixu Wu, Yuezhou Ma, Hang Zhou, Huikun Weng, Jianmin Wang, and Mingsheng Long. Propinn: Demystifying propagation failures in physics-informed neural networks. *arXiv preprint arXiv:2502.00803*, 2025a.

Wenyuan Wu, Siyuan Duan, Yuan Sun, Yang Yu, Dong Liu, and Dezhong Peng. Deep fuzzy physics-informed neural networks for forward and inverse PDE problems. *Neural Networks*, 181:106750, 2025b.

Zixue Xiang, Wei Peng, Wen Yao, and Weien Zhou. Hybrid finite-difference physics-informed neural networks partial differential equation solver for complex geometries. *Journal of Thermophysics and Heat Transfer*, 39(4):686–703, 2025.

Chenhui Xu, Dancheng Liu, Yuting Hu, Jiajie Li, Ruiyang Qin, Qingxiao Zheng, and Jinjun Xiong. Sub-sequential physics-informed learning with state space model. 2025.

Zhiyuan Zhao, Xueying Ding, and B Aditya Prakash. Pinnsformer: A transformer-based framework for physics-informed neural networks. 2023.

A THE DUAL OPERATOR

A.1 DUAL PROPAGATION RULES FOR NETWORK PRIMITIVES

Each primitive maps the input value–tangent pair $(h, \dot{h})$ to the corresponding output pair. The output tangent is the exact derivative in the seeded direction.

Table 3: Dual propagation rules for the network primitives.

Primitive	**Rule** $(h, \dot{h}) \mapsto$
Linear	$(Wh + b,\ W\dot{h})$
Tanh	$(\tanh h,\ \dot{h}\,(1 - \tanh^2 h))$
Sine / cosine	$(\sin h,\ \dot{h}\cos h)$ / $(\cos h,\ -\dot{h}\sin h)$
Elementwise product	$(h_1 h_2,\ \dot{h}_1 h_2 + h_1 \dot{h}_2)$
Softmax / attention	composition of the rules above (exp, sum, product, quotient)
State-space scan	the linear and product rules applied stepwise along the scan

The quotient and exponential rules follow the same pattern: (h_1/h_2) carries $(\dot{h}_1 h_2 - h_1 \dot{h}_2)/h_2^2$, and e^h carries $\dot{h}\, e^h$. Every benchmark architecture is a composition of these primitives. The released wrapper registers exactly one rule for each primitive.

A.2 VALIDATION OF THE OPERATOR AGAINST AUTOGRAD

Table 4 validates the exact-1 operator against float64 autograd on the seven benchmark problems. We use a width-32 StdPINN, 4096 uniformly sampled points and the step $\delta = 0.005$ used throughout the paper. First-derivative entries report the largest relative L_2 error across all output components and input directions of the base-seeded tangent. These errors are at machine precision in float64 and match the round-off of float32 autograd in float32.

Residual entries report the relative difference from the full float64 autograd residual. The float64 column isolates centered-difference truncation: it is zero for residuals without second derivatives and reaches at most 3.7×10^{-6} on Schrödinger. The float32 residual differences and float32 autograd controls are reported separately.

Table 4: Exact-1 operator against float64 autograd on the seven benchmark problems. Passes: dual passes per residual evaluation. First derivatives: largest relative L_2 error of the base-seeded tangents over all outputs and directions. Residual: relative difference of the assembled residual. The float32 autograd columns are the round-off control.

		First derivatives			Residual		
Problem	**Passes**	Dual f64	Dual f32	AD f32	Dual f64	Dual f32	AD f32
Convection	2	2.5×10^{-15}	9.6×10^{-7}	9.9×10^{-7}	1.8×10^{-16}	3.3×10^{-7}	3.3×10^{-7}
Reaction	1	2.5×10^{-15}	9.6×10^{-7}	9.9×10^{-7}	0	2.3×10^{-7}	2.3×10^{-7}
Burgers 1D	4	2.5×10^{-15}	9.6×10^{-7}	9.9×10^{-7}	2.6×10^{-8}	8.5×10^{-8}	7.6×10^{-10}
Schrödinger	4	5.2×10^{-15}	2.0×10^{-6}	2.0×10^{-6}	3.7×10^{-6}	3.2×10^{-6}	9.6×10^{-8}
Kovasznay	6	4.0×10^{-15}	1.6×10^{-6}	1.5×10^{-6}	4.5×10^{-9}	2.1×10^{-8}	2.1×10^{-8}
Burgers 2D	7	3.9×10^{-15}	1.4×10^{-6}	1.5×10^{-6}	7.3×10^{-9}	6.2×10^{-8}	6.2×10^{-8}
Taylor–Green	7	3.3×10^{-15}	1.2×10^{-6}	1.2×10^{-6}	3.9×10^{-9}	4.7×10^{-8}	4.7×10^{-8}

A.3 Truncation and round-off of the two second-derivative routes

A centered first difference,

$$u_x \approx \frac{u(x+\delta) - u(x-\delta)}{2\delta}, \tag{6}$$

has total error $O(\delta^2) + O(\varepsilon_m/\delta)$. A centered second difference,

$$u_{xx} \approx \frac{u(x+\delta) - 2u(x) + u(x-\delta)}{\delta^2}, \tag{7}$$

has error $O(\delta^2) + O(\varepsilon_m/\delta^2)$, where ε_m denotes the relative error of one evaluation of u, approximately 10^{-7} in float32.

Dual numbers propagate the first derivative exactly, without a truncation term or $1/\delta$ amplification. Taking a centered difference of these first derivatives,

$$u_{xx} \approx \frac{u_x(x+\delta) - u_x(x-\delta)}{2\delta}, \tag{8}$$

gives error $O(\delta^2) + O(\varepsilon_m/\delta)$. Both second-derivative routes therefore have the same truncation order, although their leading constants can differ. Their round-off amplification differs by one power of δ.

Balancing truncation and round-off gives optimal step sizes of order $\varepsilon_m^{1/4}$ for pure finite differences and $\varepsilon_m^{1/3}$ for Dual-FD. The corresponding error floors are of order $\varepsilon_m^{1/2}$ and $\varepsilon_m^{2/3}$. In float32, these are 3×10^{-4} and 2×10^{-5}; the measured network error floors are 1.4×10^{-3} and 1.3×10^{-4}, respectively.

Hyper-dual numbers compute an exact second derivative in one pass by carrying second-order components through the forward graph (Fike & Alonso, 2011). Our operator keeps the graph as shallow as the network and accepts $O(\delta^2)$ truncation at second order, bounded by the round-off analysis above.

A.4 The cost of nested AD

The nesting mechanism. Consider the one-dimensional heat equation $u_t = u_{xx}$, the simplest PDE with a second derivative in its residual. For a one-hidden-layer network,

$$u_\theta(x,t) \;=\; \textstyle\sum_j a_j\, \sigma\big(z_j\big), \qquad z_j = w_j x + v_j t + b_j, \qquad \sigma = \tanh\,. \tag{9}$$

The two derivatives required by the residual can be written explicitly as

$$\partial_t u_\theta = \textstyle\sum_j a_j v_j\, \sigma'(z_j), \qquad \partial_x^2 u_\theta = \textstyle\sum_j a_j w_j^2\, \sigma''(z_j). \tag{10}$$

AD differentiates the program that evaluates u_θ, rather than using these formulas directly. Computing a second derivative therefore requires differentiating the program that produced the first derivative:

1. The forward pass evaluates $u_\theta(x,t)$ and records the computation graph G_0, including z_j, $\sigma(z_j)$ and the output sum. These intermediates are retained for the backward sweep.
2. The first backward sweep computes $\partial_x u_\theta = \sum_j a_j w_j \, \sigma'(z_j)$ and $\partial_t u_\theta$. It evaluates $\sigma'(z_j)$ and two products per hidden unit, forming a program comparable in size to the forward pass. To permit further differentiation, its backward operations are recorded as a new graph G_1 using `create_graph=True` in `PyTorch`.
3. The second backward sweep differentiates through G_1 to obtain u_{xx}, reaching σ'' through the recorded σ' nodes. Both G_0 and G_1 remain in memory. The entire structure is discarded and rebuilt at the next training step.

The closed forms in Eq. 10 follow from the toy network's single layer. For a deep network, derivatives are computed through the three steps above rather than written out explicitly.

Forward mode requires no recorded backward sweep. Seeding (x,t) with tangent $(1,0)$ produces $\sum_j a_j w_j \sigma'(z_j) = \partial_x u_\theta$ in one pass, as in Eq. 10. The same first-order propagation cannot reach σ''; our operator approximates this derivative instead (Sec. 3.2).

The mode analysis in Sec. 3.2 distinguishes parameter training from nested input differentiation. Parameter training has very large n and $m = 1$, for which reverse mode is provably the appropriate choice. The gradient costs at most a small constant times the function: $\mathrm{OPS}(\nabla f) \leq \omega \, \mathrm{OPS}(f)$, with $\omega \in [3,4]$ (Griewank & Walther, 2008, §3.3 and §4.6).

Nested input differentiation has a different cost structure. Repeated first-order AD scales exponentially with derivative order k in time and memory (Shi et al., 2024, §3.2); repeated adjoints are treated in Griewank & Walther (2008, Ch. 5). A backward pass over N operations forms a program of approximately cN operations, with $c > 1$, because each operation contributes its adjoint operations. Recording successive passes therefore multiplies the differentiated program at each order,

$$\mathrm{size}(G_k) \;\approx\; c \cdot \mathrm{size}(G_{k-1}) \;\approx\; c^k \, \mathrm{OPS}(u_\theta), \tag{11}$$

so the time and memory costs of order-k differentiation inherit the factor c^k. Each level differentiates the redundant structure inherited from the previous level, producing "combinatorial amounts of redundant work" (Bettencourt et al., 2019). This observation motivates Taylor-mode and stochastic higher-order operators (Dangel et al., 2026; Shi et al., 2024).

Memory costs also differ between modes. Reverse mode stores the evaluation trace for its backward pass, and nested reverse mode additionally stores the trace of that backward pass. Forward accumulation instead carries a tangent alongside each value without storing a trace (Griewank & Walther, 2008, §4.5 and §4.6). A Hessian-vector product computed by forward differentiation of the gradient costs approximately as much as one gradient (Pearlmutter, 1994). Thus, reverse-over-reverse is the software-default route, rather than the route prescribed by this analysis for directional second derivatives.

For the second-order residuals studied here, $k = 2$ is small, but the overhead remains structural. Two resident graphs and two recorded backward passes are rebuilt at each step, with saved-tensor memory proportional to network size times collocation batch.

These graph differences also determine the cost of the final parameter backpropagation. Under nested AD, the loss contains u_{xx} produced by a backward pass through another recorded backward pass. Backpropagation to θ traverses both passes and reaches σ''' at the activations. Under Dual, derivatives enter the loss as forward-propagated tangents, such as $u_x = \sigma'(z)\,\dot{z}$ at the last layer. Parameter backpropagation differentiates σ' once, reaching σ''. Its graph follows the forward network, widened by the tangent channel but no deeper than the network itself.

Nested AD also computes derivatives that some residuals do not need. A backward pass returns derivatives along every input direction: a pass seeded on u_x returns u_{xy} and u_{xt} alongside u_{xx}. The unused mixed derivatives are discarded. A dual pass carries one direction and computes no derivatives outside that direction.

The cost in practice. In our `PyTorch` instrumentation, each recorded backward sweep increases the differentiated graph's node count by a factor of approximately 1.6, independently of network width. This is the measured constant c in Eq. 11. Its width independence locates the growth in derivative order rather than model size. Under the matched protocol of Sec. 4, two AD runs exhaust a 16 GB GPU while the corresponding forward runs complete (Sec. 4.1). Sec. 2 reviews existing responses to this cost; the analysis here explains its computational source.

A.5 The recorded training step

Dual-FD avoids nested reverse-mode graphs for input derivatives while retaining the graph required for parameter gradients. With fixed tensor shapes and control flow, each training step is the same composition of primitive operations,

$$\theta \leftarrow \theta - \eta \, \nabla_\theta \, \mathcal{L}\big(P_m \circ \cdots \circ P_1(x;\theta)\big), \tag{12}$$

where $P_1, \ldots, P_m$ comprise tangent-carrying forward passes and loss assembly. Their sequence remains fixed across steps; only tensor values change. The program can therefore be captured once and replayed throughout training. This describes our implementation's compilation strategy. Rebuilding an AD graph alone does not establish that every nested-AD implementation is incompatible with graph capture.

The solver also reuses network objects, tensor shapes, derivative passes and the optimizer sequence across windows, allowing reuse of the compiled training step. Only the current network and its two frozen predecessors remain active.

Solving the regression fit. The regression in Eq. 5 uses increments from completed windows and estimates only two unknown scalar coefficients, a and b. Define the prediction residual as $r_j = a\, d_j + b\, d_{j-1} - d_{j+1}$. Setting the partial derivatives of $\sum_j \|r_j\|_2^2$ to zero gives $\sum_j d_j^\top r_j = 0$ and $\sum_j d_{j-1}^\top r_j = 0$. Expanding these conditions yields

$$\begin{pmatrix} \sum_j d_j^\top d_j & \sum_j d_j^\top d_{j-1} \\ \sum_j d_j^\top d_{j-1} & \sum_j d_{j-1}^\top d_{j-1} \end{pmatrix} \begin{pmatrix} a_k \\ b_k \end{pmatrix} = \begin{pmatrix} \sum_j d_j^\top d_{j+1} \\ \sum_j d_{j-1}^\top d_{j+1} \end{pmatrix}, \tag{13}$$

where all sums run from $j = 3$ to $k-1$. The symmetric matrix has three independent entries, and the right-hand side has two. These five sums of inner products collect contributions from all parameter components and completed triplets. Denoting the diagonal entries of the matrix in Eq. 13 by S_1 (top left) and S_3 (bottom right), its off-diagonal entry by S_2, and the right-hand-side entries by S_4 (top) and S_5 (bottom), the system has the closed-form solution

$$a_k = \frac{S_3 S_4 - S_2 S_5}{S_1 S_3 - S_2^2}, \qquad b_k = \frac{S_1 S_5 - S_2 S_4}{S_1 S_3 - S_2^2}, \tag{14}$$

whenever $S_1 S_3 - S_2^2 > 0$, so no auxiliary prediction network is trained. We added 10^{-8} to S_1 and S_3 in every window to ensure that the denominator stays positive.

Each completed window adds one triplet, allowing the sums to be updated without retaining the full parameter history. The regression stores only the two most recent increments and the accumulated sums, so its storage does not grow with the number of windows.

B Experimental setup

B.1 Hyperparameters and architecture details

The ten architectures are StdPINN, QRes (Bu & Karpatne, 2021), sf-PINN (Wong et al., 2022), PIKAN (Patra et al., 2025), PINNsFormer (Zhao et al., 2023), PINNMamba (Xu et al., 2025), ProPINN (Wu et al., 2025a), FPINN (Wu et al., 2025b), PIG (Kang et al., 2025) and IFeF (Aguiar et al., 2026). Each follows its official reference configuration as closely as possible. AD and Dual-FD share the same capacity within each pair; capacity sources are recorded in App. D.1. The Dual-FD step is $\delta = 5 \times 10^{-3}$. Stage-timing and end-to-end experiments use different AD pass organizations and compilation settings, as specified in App. C.1.

Training recipe and its architecture-specific exceptions. Paired runs share the optimizer, budget, collocation set and loss weights; only the derivative operator differs. The generic recipe fixes 10^4 collocation points per run and gives equal weights to the residual, initial and boundary terms. It uses L-BFGS with strong-Wolfe line search and at most 20 inner iterations per outer step.

Five architectures instead follow their published schedules. StdPINN, QRes and FPINN use 10^4 Adam steps at learning rate 10^{-3} before L-BFGS. PIKAN uses Adam only, and IFeF follows the two-level procedure of its reference implementation.

Sampling and loss adaptations also follow the reference implementations. PINNsFormer and PINN-Mamba use the official tensor-product grids instead of random point clouds. PINNMamba additionally retains the official sequence-consistency term at its published weight. PIG resamples its point cloud at every outer step, as in its release.

Compilation settings are uniform within each operator. Dual-FD uses `torch.compile`, except on PIKAN, whose kernels do not compile. AD and FD run eagerly; the nested reverse graph does not compile in this version of PyTorch. Chunked residual backpropagation is used only when a single recorded graph exceeds GPU memory. This applies throughout PINNsFormer and PINNMamba, and to AD in ProPINN configurations with three-dimensional inputs. Chunking changes the memory schedule but not the loss.

Each run ends when it reaches its budget or the validation error plateaus. Validation uses relative L_2 error against the audited reference, evaluated every 50 L-BFGS steps or 1000 Adam steps. A plateau is declared when ten consecutive evaluations show less than one percent relative improvement. This criterion monitors reference-relative error rather than training loss, with identical parameters for both operators.

B.2 Reference-solution precision audit

Reference precision and the acceptability of a learned physical field are assessed separately. A timing threshold is admitted only when $\varepsilon / E_{\mathrm{ref}} \geq 10$, requiring the threshold to be at least ten times the estimated reference error. The available reference audits are summarized below.

Kovasznay and Taylor–Green have analytic solutions. Their float32 evaluation discrepancy is approximately 2×10^{-7}, an evaluation artefact rather than reference error, so no planned threshold is affected. Convection and reaction use the analytic reference solutions from the architecture papers that introduced these problems.

For Burgers 1D, a spatial and temporal Richardson study estimates the reference error at 1.2×10^{-10}. For Burgers 2D, three-grid Richardson extrapolation gives 4.3×10^{-4}, admitting thresholds of 10^{-2} and 5×10^{-3}. The 10^{-3} threshold does not meet the criterion. Schrödinger was cross-validated against a split-step Fourier solver, giving a likely reference error of 2×10^{-7} and a conservative bound of 2×10^{-5}. This admits thresholds down to 2×10^{-4} for the headline results.

The cylinder comparison in Sec. 4.3 uses a fixed reference dataset whose pointwise error cannot be quantified from the published solver settings. All cylinder comparisons therefore use that dataset, rather than the continuous solution, as the reference.

Acceptance of individual seeds. Each seed is assessed using both relative L_2 error and the predicted physical field. Field inspection checks the reference's spatial structure and, where applicable, its temporal evolution; crossing an error threshold alone does not establish physical agreement. Comparisons for Secs. 4.1 and 4.3 are collected in Apps. D.5 and E.2.

The required features depend on the problem. Kovasznay and Taylor–Green predictions must recover the analytic solution's stripe and vortex patterns, respectively, in the u field at the evaluation times. Burgers 2D must retain the four bumps on its unit background. A constant field already achieves relative $L_2 = 0.044$ and could therefore pass a loose threshold without recovering those bumps. For Burgers 1D and Schrödinger, the final profile must follow the reference shape rather than its mean level. Convection and reaction must recover the travelling wave and logistic plateau, respectively, rather than a constant.

Each comparison in Table 1 is evaluated with three random seeds, with the same physical-field checks applied to both operators for each seed. If neither operator reaches the original threshold for

a seed, the pair is compared at a relaxed common threshold. The result is retained only after field inspection confirms that both operators recover the expected structure.

Every comparison with a numerical entry in Table 1 enters the paired statistics. A star marks a comparison in which at least one of the three seeds uses a relaxed threshold. It identifies threshold relaxation, not a reduction in the number of seeds. Results are not accepted if, even after threshold relaxation, failure to converge leaves large relative L_2 errors and the predicted fields fail to recover the physical features. Comparisons in which neither operator produces an acceptable result are marked *n/c*.

Within each accepted comparison, speedup is the median over thresholds and seeds. Aggregate statistics summarize these comparison-level values; no run is removed from a comparison after the fact.

C PER-STEP AND SAME-ACCURACY RESULTS

The detailed graph-level breakdown below uses the Kovasznay residual, the steady incompressible Navier–Stokes equations at $Re = 20$ in two dimensions, with velocity (u, v) and pressure p as the three network outputs and (x, y) as the two inputs:

$$\begin{aligned} R_u &= u\,u_x + v\,u_y + p_x - \tfrac{1}{Re}\,(u_{xx} + u_{yy}),\\ R_v &= u\,v_x + v\,v_y + p_y - \tfrac{1}{Re}\,(v_{xx} + v_{yy}),\\ R_c &= u_x + v_y, \qquad \mathcal{L}_{\mathrm{PDE}} = \tfrac{1}{N}\textstyle\sum_{i=1}^{N}\big(R_u^2 + R_v^2 + R_c^2\big). \end{aligned} \tag{15}$$

The residual needs ten derivative terms: six of first order, $u_x, u_y, v_x, v_y, p_x, p_y$, and four of second order, $u_{xx}, u_{yy}, v_{xx}, v_{yy}$; the mixed terms u_{xy} and v_{xy} do not occur.

C.1 STAGE-TIMING PROTOCOL AND GRAPH-LEVEL BREAKDOWN

Each route of Sec. 4.2 trains the same StdPINN with Adam on a fresh batch of 8192 collocation points per step, for 300 steps, at width 32 for the main-text stage-timing figure. CUDA events are recorded at the boundaries of the operations of that route, the device is synchronised at the end of every step, and the time of an operation is the median over the last 250 steps, with quartiles as error bars. One operation is one call of the route, a forward pass, one backward pass, one dual pass, or the packed call. AD is run in its one-pass-per-component form, each backward pass returning every input direction; some implementations take one backward pass per term instead, which is dearer, so the setting favors AD. FD evaluates its point sets in one network call, as Dual does its passes; one call per point set would leave five graphs in the loss instead of one and cost 3.6 ms per step on Kovasznay instead of 2.1. Unlike FD-PINN and DT-PINN, which train on a fixed mesh or point cloud and take the neighbors from it, the FD route keeps the random collocation sampling of the other three and builds its five-point stencil around every sampled point, so the point sets are fresh at every step. All four routes run in eager mode, so the comparison is one of arithmetic and graph structure, not of compilation; the matrix of Sec. 4.1 compiles the Dual route.

The supplementary Kovasznay breakdown in Tables 5 and 6 uses width 64, as specified in their captions. Its FD implementation evaluates the five point sets separately; it is distinct from the packed FD implementation above. These measurements supplement, rather than supply numerical proportions for, the width-32 main-text figure.

Table 5 splits backpropagation by the part of the graph it traverses. Under AD three quarters of it is spent traversing the four recorded second-order passes; a recorded backward pass costs about three times as much to traverse as it cost to run, a forward about 1.1 times. Under AD+FD the eight shifted passes cost four times their running cost to traverse, which is where the lowered derivative order is spent. Under Dual the value and tangent channels share one complex tensor in our implementation and cannot be separated; the wide graph costs 1.4 times its running cost to traverse, and two thirds of that is elementwise complex arithmetic rather than matrix products, an implementation cost that a real-pair dual type would reduce. Kernel counts tell the same story at small width, where a step is bound by kernel launches. Backpropagation launches 1131 kernels per step under AD, 1183 under AD+FD and 228 under Dual, and at width 32 each kernel takes about 6 μs whatever it computes.

Table 5: Backpropagation to θ split by the part of the graph it traverses, Kovasznay, width 64, batch 8192, eager mode. The full graph of each route is built every step; backpropagation is started from a loss that depends on a growing part of it (values only; plus first-order terms; plus second-order terms), one variant per step in rotation, and each increment is the traversal cost of the part added (CUDA-event median over 240 steps). "Run" is the time the same part took to execute (Table 6). Kernel counts come from a 30-step `torch.profiler` trace with every kernel attributed to the operation that launched it. Under Dual the value and tangent channels share one complex tensor and cannot be separated.

Route	Part of the graph traversed	Run (ms)	Traverse (ms)	Ratio
AD	forward graph	0.46	0.51	1.1
AD	three recorded first-order backward passes	0.82	2.12	2.6
AD	four recorded second-order backward passes	2.38	7.73	3.2
AD	total (1131 kernels)	3.66	10.36	
AD+FD	forward graph	0.46	0.69	1.5
AD+FD	three recorded base passes	0.50	2.12	4.2
AD+FD	four shifted forwards	1.62	2.17	1.3
AD+FD	eight recorded shifted passes	1.35	5.37	4.0
AD+FD	total (1183 kernels)	3.93	10.35	
FD	base forward	0.46	0.48	1.0
FD	four shifted forwards	1.61	2.16	1.3
FD	total (395 kernels)	2.07	2.64	
Dual	one complex forward graph, six batch rows wide	3.91	5.42	1.4
Dual	residual assembly		0.17	
Dual	total (228 kernels)	3.91	5.59	

C.2 The one-dimensional problems

The one-dimensional benchmarks supplement the main two-dimensional speed comparisons. Their seed-level errors and time-to-threshold records are listed in App. D. Convection and reaction contain no second-order derivative terms, whereas Burgers 1D and Schrödinger have different second-order derivative requirements; their speedups therefore need to be assessed separately.

Slower and non-reaching runs are retained alongside faster results.

After the field audit, 25 one-dimensional comparisons admit a pair. Dual-FD is faster in 19 of them, with a median of $1.29\times$ and a maximum of $12.92\times$ on Burgers 1D with IFeF. Of the six comparisons where AD is faster, three sit on Burgers 1D, which reaches its thresholds in under a minute, so the ratio there measures launch overhead; the other three, on Reaction and Convection, stay within 6% of parity. Three of the 25 comparisons pair at a relaxed threshold that both operators reach, Burgers 1D with IFeF at 2×10^{-2}, Schrödinger with sf-PINN at 5×10^{-2} and Reaction with PINNMamba at 10^{-1}, admitted only after the predicted fields were checked against the reference; the remaining 22 pair at the thresholds of the main matrix, so the two tables share one admission rule. Configurations that no threshold pairs carry the marks of Table 1 and enter no aggregate.

C.3 The finite-difference route on the tested subset

The cheapest way to remove the nested operator is to difference the network output instead, as FD-PINN does (Jiang et al., 2023). We added it to the experiments on a subset of four problems, under the same protocol, capacity, thresholds and seeds, with the stencil built around each randomly sampled point rather than on a mesh and its point sets evaluated in one call, as in Sec. 4.2 (Table 9). Per step it is the cheapest route (Fig. 4); over a whole run its wall clock is $0.70\times$ that of Dual on median, from $0.4\times$ on Taylor–Green to $1.9\times$ on reaction, where it stops later because it converges worse. It is the least accurate overall: the median L_2 ratio against AD is $6.6\times$ for FD and $1.14\times$ for Dual, and on Kovasznay FD ends 12 to $80\times$ less accurate, while on the QRes reaction comparison it ends below both. Of the seven comparisons where AD reaches a threshold, FD reaches none in three; Dual reaches a threshold in all seven and AD's tightest in five, the two exceptions being the reaction comparisons.

The reason is where each method spends its step δ. A central difference of function values carries a truncation of order δ^2 and a round-off of order ε/δ at first order, and at second order the same

Table 6: Every operation of one Kovasznay training step under each route, in execution order: whether it is a forward pass, a backward pass or a dual pass, what it runs on, which quantities it returns, its time and its kernel count. Setting: StdPINN, width 64, batch 8192, Adam, eager mode, RTX 3080, PyTorch 2.13. Times are CUDA-event medians over the last 250 of 300 training steps on freshly sampled points, each step synchronised at its end; quartiles are within 5% of the median except where marked. Kernel counts are per step from a 30-step `torch.profiler` trace, each kernel attributed to the operation that launched it. The three-part split of backpropagation is the marginal measurement of Table 5. "Recorded" means the pass is run with `create_graph=True` so that later backward passes and backpropagation can traverse it. $x^{\pm}$ denotes the points $(x \pm \delta, y)$, $y^{\pm}$ the points $(x, y \pm \delta)$, $\delta = 0.005$. "Discarded" marks a by-product a pass computes at no extra cost but the residual does not use.

#	Operation	Runs on	Returns	ms	Kernels
AD (nested reverse mode; step 14.44 ms)					
1	forward, recorded	base points	u, v, p	0.46	23
2	backward pass from u, recorded	graph of 1	u_x, u_y	0.17	22
3	backward pass from v, recorded	graph of 1	v_x, v_y	0.29	22
4	backward pass from p, recorded	graph of 1	p_x, p_y	0.36	22
5	backward pass from u_x, recorded	graphs of 1 and 2	u_{xx} (and u_{xy}, discarded)	0.70	69
6	backward pass from u_y, recorded	graphs of 1 and 2	u_{yy} (and u_{yx}, discarded)	0.57	69
7	backward pass from v_x, recorded	graphs of 1 and 3	v_{xx} (and v_{xy}, discarded)	0.57	69
8	backward pass from v_y, recorded	graphs of 1 and 3	v_{yy} (and v_{yx}, discarded)	0.54	69
9	loss assembly (elementwise, mean)	outputs of 1 to 8	$\mathcal{L}$	0.05	
10	backpropagation to θ	graphs of 1 to 9	$\nabla_\theta \mathcal{L}$	10.63	1131
	through the forward graph			0.51	
	through the three recorded first-order passes			2.12	
	through the four recorded second-order passes			7.73	
11	Adam update	$\nabla_\theta \mathcal{L}$	θ	0.03	7
AD+FD (reverse mode at first order, differences of first derivatives at second; step 14.15 ms)					
1	forward, recorded	base points	u, v, p	0.46	23
2	three backward passes from u, v, p, recorded	graph of 1	$u_x, u_y, v_x, v_y, p_x, p_y$	0.50	66
3	four forwards, recorded	$x^{\pm}$, $y^{\pm}$	u, v, p at the four points	1.62	100
4	eight backward passes from u and v, recorded	graphs of 3	u_x, v_x at $x^{\pm}$; u_y, v_y at $y^{\pm}$ (the other direction of each pass discarded)	1.35	176
5	central differences and loss assembly	outputs of 2 and 4	$u_{xx}, u_{yy}, v_{xx}, v_{yy}$; $\mathcal{L}$	0.06	
6	backpropagation to θ	graphs of 1 to 5	$\nabla_\theta \mathcal{L}$	10.11	1183
	through the forward graph			0.69	
	through the three recorded base passes			2.12	
	through the four shifted forwards			2.17	
	through the eight recorded shifted passes			5.37	
7	Adam update		θ	0.03	7
FD (central differences of function values at both orders; step 4.86 ms)					
1	forward	base points	u, v, p	0.46	23
2	two forwards	$x^{\pm}$	u, v, p at $x^{\pm}$	0.80	48
3	two forwards	$y^{\pm}$	u, v, p at $y^{\pm}$	0.80	48
4	central differences and loss assembly	outputs of 1 to 3	all six first and four second derivatives; $\mathcal{L}$	0.10	
5	backpropagation to θ	graphs of 1 to 4	$\nabla_\theta \mathcal{L}$	2.66	395
	through the base forward			0.48	
	through the four shifted forwards			2.16	
6	Adam update		θ	0.03	7
Dual (exact first order from tangents, one difference of tangents at second; step 9.60 ms)					
1	one dual pass, six point sets packed into one call	base with seed $\hat{\mathbf{e}}_x$; base with $\hat{\mathbf{e}}_y$; $x^{\pm}$ with $\delta\hat{\mathbf{e}}_x$; $y^{\pm}$ with $\delta\hat{\mathbf{e}}_y$	u, v, p; u_x, v_x, p_x; u_y, v_y, p_y; $\delta u_x, \delta v_x$ at $x^{\pm}$; $\delta u_y, \delta v_y$ at $y^{\pm}$ (also $\delta p_x, \delta p_y$ and u, v, p at the four shifted points, discarded)	3.91	61
	run as four separate calls: base pass seeded $\hat{\mathbf{e}}_x$	base points	u, v, p; u_x, v_x, p_x	0.79	
	base pass seeded $\hat{\mathbf{e}}_y$	base points	u_y, v_y, p_y	0.63	
	probe pair seeded $\delta\hat{\mathbf{e}}_x$	$x^{\pm}$	$\delta u_x, \delta v_x$ at $x^{\pm}$	1.26	
	probe pair seeded $\delta\hat{\mathbf{e}}_y$	$y^{\pm}$	$\delta u_y, \delta v_y$ at $y^{\pm}$	1.26	
2	central differences and loss assembly	tangents of 1	$u_{xx}, u_{yy}, v_{xx}, v_{yy}$; $\mathcal{L}$	0.06	
3	backpropagation to θ	graph of 1 and 2	$\nabla_\theta \mathcal{L}$	5.60	228
	through the complex forward graph (value and tangent fused)			5.42	
	through the residual assembly			0.17	
4	Adam update		θ	0.03	7

Table 7: Passes per residual evaluation for the values and the ten derivative terms of Kovasznay, by derivative order and route. *Recorded* counts the backward passes left in the loss graph for the final backpropagation to traverse.

Order	Terms	AD+AD	AD+FD	FD+FD	Dual+FD
0	u, v, p	1 fwd	1 fwd	1 fwd	with order 1
1	$u_x, u_y, v_x, v_y, p_x, p_y$	3 bwd	3 bwd	4 fwd	2 dual
2	$u_{xx}, u_{yy}, v_{xx}, v_{yy}$	4 bwd	4 fwd + 8 bwd	same fwd	4 dual
Total		1 fwd + 7 bwd	5 fwd + 11 bwd	5 fwd	6 dual
Recorded		7 bwd	11 bwd	none	none
Accuracy		exact	2nd order $O(\delta^2)$	$O(\delta^2)$, floor 1.4×10^{-3}	2nd order $O(\delta^2)$

Table 8: Forward and backward counts per residual evaluation for every problem of the matrix, read from the residual code of each route (`count_passes.py`, `exact1_residuals.py`). AD is listed in the two forms that occur in this paper: the per-direction form of the reference implementations, one backward pass per derivative term, which the matrix of Table 1 runs; and the one-pass-per-component form, in which a first-order pass returns every input direction of one output, which the ledger of Sec. 4.2 prices as the cheaper lower bound for AD. Every AD pass is recorded. Dual passes are one per seed direction at the base point (exact first order) plus one probe pair per direction that carries a second-order term. FD counts are the plain forwards a central-difference residual needs. "Graphs" is the number of recorded graphs backpropagation to θ traverses.

	Terms		AD, per direction		AD, per component		Dual		FD	
Problem	1st	2nd	fwd	bwd (graphs)	fwd	bwd (graphs)	base	probe (graphs)	fwd	(graphs)
Reaction	1	0	1	1 (2)	1	1 (2)	1	0 (1)	3	(3)
Convection	2	0	1	2 (3)	1	1 (2)	2	0 (1)	4	(4)
Burgers 1D	2	1	1	3 (4)	1	2 (3)	2	2 (1)	5	(5)
Schrödinger	4	2	1	6 (7)	1	4 (5)	2	2 (1)	5	(5)
Kovasznay	6	4	1	10 (11)	1	7 (8)	2	4 (1)	5	(5)
Burgers 2D	6	4	1	10 (11)	1	6 (7)	3	4 (1)	7	(7)
Taylor–Green	8	4	1	12 (13)	1	7 (8)	3	4 (1)	7	(7)

subtraction is divided by δ^2, so the round-off grows as ε/δ^2; the measured slopes are -2 for the round-off and $+2$ for the truncation, meeting at a float32 floor of 1.4×10^{-3}. The Dual route takes its first derivatives from the tangent channel with no subtraction at all, exact to float32 round-off (8×10^{-7} relative) and independent of δ, and its single difference acts on quantities that already carry a factor δ, so the round-off grows as ε/δ, one power slower, and the floor drops to 1.3×10^{-4}.

Table 9: AD, Dual and real central differences (FD-PINN) on the tested subset, at matched capacity, protocol, thresholds and seeds. L_2 is the median best error over three seeds; Dual is the exact-1 operator of the matrix, and FD evaluates its point sets in one call.

Problem	Architecture	L_2 AD	L_2 Dual	L_2 FD	tightest reached (AD | Dual | FD)
Kovasznay	PINNsFormer	1.4e–3	4.0e–3	9.0e–2	5e–3 | 5e–3 | none
Kovasznay	QRes	4.6e–4	7.2e–4	5.6e–3	1e–3 | 1e–3 | 5e–3
Kovasznay	StdPINN	5.2e–4	6.0e–4	8.0e–3	1e–3 | 1e–3 | 1e–2
Reaction	PINNsFormer	4.2e–2	3.2e–2	1.0e–1	5e–2 | 5e–2 | 5e–2
Reaction	QRes	3.6e–2	4.7e–2	2.8e–2	5e–3 | 5e–2 | 5e–2
Reaction	StdPINN	7.0e–2	9.8e–1	9.9e–1	5e–2 | 1e–1 | none
Taylor–Green	StdPINN	4.7e–3	4.7e–3	3.2e–2	5e–3 | 5e–3 | none

The QRes and PINNsFormer comparisons of Taylor–Green reach no threshold in any method and carry no claim.

C.4 Width and depth scans

Table 10 lists every paired group of the width scan behind Fig. 3b. The AD side reuses the matrix runs; the Dual side was rerun under the operator of the matrix. Peak memory of the Dual run relative to the AD run at the same width, over the five architectures the width scan covers, is 0.30 to 0.37 on PINNsFormer and PINNMamba and 0.48 to 0.78 on the MLP-type architectures over the two-dimensional problems, against a median of 0.62 over the full matrix in Table 11; the width-128 PINNsFormer runs on Kovasznay and Taylor–Green did not complete under AD on the 20 GB card and completed under Dual at 10.4 and 11.5 GB.

Table 10: Width scan: ratio of total training wall-clock time, AD/Dual-FD (median over seeds). Columns give the hidden width. This measures whole-run cost rather than the time-to-threshold ratio in Fig. 3b. †: either AD or Dual-FD produces large errors and fails to capture the physical features at that size, so no ratio is reported.

Problem	**Architecture**	16	32	64	128	256
Kovasznay	StdPINN	2.28	1.75	1.94	2.41	2.40
Kovasznay	QRes	3.50	2.75	2.97	3.17	2.70
Kovasznay	PINNsFormer	5.93	4.72	3.81	–	–
Kovasznay	PINNMamba	–	7.13	–	–	–
Taylor–Green	StdPINN	2.63	2.01	2.31	2.61	3.14
Taylor–Green	QRes	4.46	2.66	2.83	3.69	3.29
Taylor–Green	PINNsFormer	3.71	4.21	4.49	–	–
Taylor–Green	PINNMamba	–	7.74	–	–	–
Burgers 1D	StdPINN	1.96	1.27	1.03	0.58	0.92
Burgers 1D	QRes	3.02	2.02	1.49	1.55	1.59
Burgers 1D	PINNsFormer	1.09	2.63	1.59	1.61	–
Burgers 1D	PINNMamba	–	2.41	–	–	–

Table 11 lists the peak-memory ratio Dual-FD over AD for every comparison of the matrix with both runs recorded.

Table 11: Peak GPU memory of the Dual-FD run over that of the AD run (median over the three seeds). A star marks comparisons outside the paired speed comparison of Table 1; –: one of the two runs is missing. Dual-FD runs under `torch.compile` except on PIKAN, which runs eagerly and is the one architecture where Dual-FD uses more memory than AD. †: the AD run of that comparison needs a chunked backward pass to fit on the card, which lowers its peak by construction, so the ratio is not comparable and is omitted.

	StdPINN	QRes	sf-PINN	PIKAN	PINNsF.	PMamba	ProPINN	FPINN	PIG	IFeF
Reaction	0.89	0.94	0.92*	1.22*	0.57	1.00	0.39	0.92	0.17*	0.63
Convection	1.02*	1.16*	1.05*	2.04*	0.63*	1.00*	0.45	1.11*	0.25*	0.97*
Burgers 1D	1.11	1.19	1.05	2.95	0.61	0.92	0.49	1.09	0.71	1.17
Schrödinger	0.69	0.70	0.72	2.20*	0.39	0.55*	0.27	0.64	0.20	0.88*
Kovasznay	0.58	0.65	0.65	2.18	0.30	0.32	0.26	0.59	0.17	0.82
Burgers 2D	0.69	0.73	0.73	2.50*	0.38*	0.37	†	0.66	0.39*	0.96*
Taylor–Green	0.60	0.63	0.65	2.40*	0.33	0.32	†	0.56	–	0.84

D CAPACITY-CONTROLLED RE-RUN OF THE SPEED MATRIX

This appendix records capacities and runs underlying the speed matrix. The intended selection rule is to retain the published configuration for an architecture–problem pair where available and otherwise select capacity using AD, then use the same capacity for Dual-FD without retuning. The capacity table distinguishes reference configurations (R), sweep-selected configurations (S), carried-over defaults (P) and parameter-matched configurations (T).

Of the 68 entries, 19 are R, taken from a reference implementation; 8 are S, selected by a completed AD width sweep; 36 are P, the architecture's published default carried to a problem its paper does not report; and 5 are T, the PIKAN rows widened to 32 with grid size 10 for parameter parity with the standard MLP, a choice made once and applied to every PIKAN configuration rather than tuned per problem. The selection rule of Sec. 4 is therefore realised in full only on the R and S entries. AD width sweeps have been run for StdPINN, QRes and PINNsFormer on Kovasznay, Taylor–Green and Burgers 1D, and for sf-PINN where its S rows record; the S code appears only where such a sweep saturated. The P entries of PIG, IFeF, ProPINN, FPINN, PINNMamba and the remaining QRes and PINNsFormer configurations have no sweep behind them and are not claimed to be the attainable accuracy of those architectures: a P configuration measures the published default on the problem in question. Because a pair shares its capacity across the two operators, the provenance code affects what an absolute error means, not the speedup, which is always read within a pair at fixed capacity.

What is recorded per run. Each run directory keeps the full loss trace, the L_2 history, the wall-clock time at which each admitted threshold is first reached, the peak memory, the initialization and collocation hashes that establish the pairing, and the resolved configuration.

D.1 Network capacity per architecture and problem

Capacity is set per architecture, never copied from the standard MLP. Where an architecture's own reference implementation fixes a size for a problem we adopt that size unchanged, neither larger nor smaller. Where it does not, we either take the smallest width at which our own sweep stops improving, or carry that architecture's published default. Within a configuration AD and Dual always share the capacity, so a speed comparison never involves two different networks.

Sizes are written width $\times$ number of Linear layers, the convention of the reference code, in which four hidden layers of 100 are 100×5. Provenance codes: **R** reference implementation, **S** our width sweep at saturation, **P** that architecture's published default carried to a problem its paper does not report, **T** tuned by us for parameter parity.

Rows marked **P** carry the architecture's published default to a problem its paper does not report; a P configuration measures that published default rather than the architecture's attainable accuracy, and each P row is replaced by a swept size (code **S**) once its width sweep completes.

Table 12: Network capacities used for each architecture–problem pair. AD and Dual-FD share the same capacity; provenance codes are defined above.

Architecture	Problem	Capacity	Provenance
StdPINN	Schrödinger	100×5	**R**
	Convection	512×4	**R**
	Reaction 1D	512×4	**R**
	Kovasznay	32×6	**S**
	Taylor–Green	64×6	**S**
	Burgers 2D	32×5	**S**
	Burgers 1D	20×9	**R**
QRes	Schrödinger	14×7	**P**
	Convection	256×4	**R**
	Reaction 1D	256×4	**R**
	Kovasznay	14×7	**P**
	Taylor–Green	14×7	**P**
	Burgers 2D	14×7	**P**
	Burgers 1D	14×7	**R**
sf-PINN	Schrödinger	512×4	**S**
	Convection	512×4	**R**
	Reaction 1D	512×4	**R**
	Kovasznay	512×4	**S**
	Taylor–Green	512×4	**S**
	Burgers 2D	32×4	**S**
	Burgers 1D	512×4	**S**
PIKAN	Schrödinger	$2-32-32-2$	**T**
	Convection	$2-5-5-1$	**R**
	Reaction 1D	$2-5-5-1$	**R**
	Kovasznay	$2-32-32-3$	**T**
	Taylor–Green	$3-32-32-3$	**T**
	Burgers 2D	$3-32-32-2$	**T**
	Burgers 1D	$2-32-32-1$	**T**
PINNsFormer	Schrödinger	d_m=32, d_h=512, N=1	**P**
	Convection	d_m=32, d_h=512, N=1	**R**
	Reaction 1D	d_m=32, d_h=512, N=1	**R**
	Kovasznay	d_m=32, d_h=128, N=1	**P**
	Taylor–Green	d_m=32, d_h=512, N=1	**P**
	Burgers 2D	d_m=32, d_h=512, N=1	**P**
	Burgers 1D	d_m=32, d_h=128, N=1	**P**
PINNMamba	Schrödinger	32×1	**P**
	Convection	32×1	**R**
	Reaction 1D	32×1	**R**
	Kovasznay	32×1	**P**
	Taylor–Green	32×1	**P**
	Burgers 2D	32×1	**P**
	Burgers 1D	32×1	**P**
ProPINN	Schrödinger	32×3	**P**

Architecture	Problem	Capacity	Provenance
	Convection	32×3	**R**
	Reaction 1D	32×3	**R**
	Kovasznay	16×3	**P**
	Burgers 1D	32×3	**P**
FPINN	Schrödinger	200×4	**P**
	Convection	200×4	**P**
	Reaction 1D	200×4	**P**
	Kovasznay	400×4	**P**
	Taylor–Green	200×4	**P**
	Burgers 2D	200×4	**P**
	Burgers 1D	100×4	**P**
PIG	Schrödinger	G=800, k=4, 16×2	**P**
	Convection	G=800, k=4, 16×2	**P**
	Reaction 1D	G=800, k=4, 16×2	**P**
	Kovasznay	G=800, k=4, 16×2	**P**
	Taylor–Green	G=800, k=4, 16×2	**P**
	Burgers 2D	G=800, k=4, 16×2	**P**
	Burgers 1D	G=200, k=4, 16×2	**P**
IFeF	Schrödinger	K=800, 64×6	**P**
	Convection	K=800, 64×6	**R**
	Reaction 1D	K=800, 64×6	**P**
	Kovasznay	K=800, 64×6	**P**
	Taylor–Green	K=800, 64×6	**P**
	Burgers 2D	K=800, 64×6	**P**
	Burgers 1D	K=800, 64×6	**R**

Problems absent from the table are absent for every architecture, so the problem set stays identical across the matrix. KdV is excluded because no reference solution of sufficient accuracy is available for it, and Helmholtz is not implemented in our residual library.

D.2 Detailed training results by problem

Table 13 lists the recorded training outcomes. AD and Dual-FD use the same capacity within each configuration (App. D.1); capacity-audit status is described in App. D. Errors are relative L_2 errors against the reference solutions in App. B.2.

Table 13: Training outcomes by problem and architecture on an NVIDIA RTX 3080.

Arch.	Capacity	Params	Operator	Best L_2	Wall (s)
Schrödinger					
StdPINN	100×5	30 802	AD	1.54×10^{-3}	964
StdPINN	100×5	30 802	Dual-FD	2.21×10^{-3}	496
QRes	14×7	2 942	AD	2.43×10^{-3}	455
QRes	14×7	2 942	Dual-FD	5.50×10^{-2}	175
sf-PINN	512×4	527 874	AD	3.58×10^{-3}	1019
sf-PINN	512×4	527 874	Dual-FD	4.09×10^{-2}	352
PIKAN	$2-32-32-2$	17 280	AD	4.28×10^{-1}	10717
PIKAN	$2-32-32-2$	17 280	Dual-FD	4.94×10^{-1}	2003
PINNsFormer	d_m=32, d_h=512, N=1	454 074	AD	7.53×10^{-3}	1739
PINNsFormer	d_m=32, d_h=512, N=1	454 074	Dual-FD	3.46×10^{-2}	366
PINNMamba	32×1	286 276	AD	5.13×10^{-3}	4499
PINNMamba	32×1	286 276	Dual-FD	4.03×10^{-1}	758
ProPINN	32×3	15 095	AD	2.04×10^{-3}	841
ProPINN	32×3	15 095	Dual-FD	8.33×10^{-3}	263
FPINN	200×4	123 256	AD	1.98×10^{-3}	589
FPINN	200×4	123 256	Dual-FD	2.76×10^{-3}	321
PIG	G=800, k=4, 16×2	16 114	AD	5.05×10^{-4}	11140
PIG	G=800, k=4, 16×2	16 114	Dual-FD	6.02×10^{-4}	924
IFeF	K=800, 64×6	24 322	AD	1.45×10^{-1}	4191
IFeF	K=800, 64×6	24 322	Dual-FD	1.57×10^{-1}	1377
Convection					
StdPINN	512×4	527 361	AD	9.48×10^{-1}	253
StdPINN	512×4	527 361	Dual-FD	8.86×10^{-1}	242
QRes	256×4	526 849	AD	1.00×10^{0}	296
QRes	256×4	526 849	Dual-FD	9.99×10^{-1}	204
sf-PINN	512×4	527 361	AD	1.00×10^{0}	179
sf-PINN	512×4	527 361	Dual-FD	1.00×10^{0}	185

Continued on next page

Table 13 (continued)

Arch.	Capacity	Params	Operator	Best L_2	Wall (s)
PIKAN	$2-5-5-1$	400	AD	1.00×10^{0}	110
PIKAN	$2-5-5-1$	400	Dual-FD	1.00×10^{0}	105
PINNsFormer	d_m=32, d_h=512, N=1	453 561	AD	8.81×10^{-1}	701
PINNsFormer	d_m=32, d_h=512, N=1	453 561	Dual-FD	1.00×10^{0}	272
PINNMamba	32×1	285 763	AD	1.00×10^{0}	932
PINNMamba	32×1	285 763	Dual-FD	1.00×10^{0}	516
ProPINN	32×3	15 030	AD	8.58×10^{-1}	196
ProPINN	32×3	15 030	Dual-FD	8.26×10^{-2}	162
FPINN	200×4	123 241	AD	9.51×10^{-1}	155
FPINN	200×4	123 241	Dual-FD	9.88×10^{-1}	147
PIG	G=800, k=4, 16×2	16 097	AD	9.99×10^{-1}	653
PIG	G=800, k=4, 16×2	16 097	Dual-FD	9.99×10^{-1}	141
IFeF	K=800, 64×6	22 657	AD	1.00×10^{0}	362
IFeF	K=800, 64×6	22 657	Dual-FD	1.00×10^{0}	199
Reaction 1D					
StdPINN	512×4	527 361	AD	7.01×10^{-2}	57
StdPINN	512×4	527 361	Dual-FD	9.83×10^{-1}	35
QRes	256×4	526 849	AD	3.59×10^{-2}	182
QRes	256×4	526 849	Dual-FD	4.74×10^{-2}	128
sf-PINN	512×4	527 361	AD	9.78×10^{-1}	47
sf-PINN	512×4	527 361	Dual-FD	9.61×10^{-1}	41
PIKAN	$2-5-5-1$	400	AD	4.40×10^{-1}	176
PIKAN	$2-5-5-1$	400	Dual-FD	4.36×10^{-1}	190
PINNsFormer	d_m=32, d_h=512, N=1	453 561	AD	4.16×10^{-2}	137
PINNsFormer	d_m=32, d_h=512, N=1	453 561	Dual-FD	3.15×10^{-2}	87
PINNMamba	32×1	285 763	AD	9.65×10^{-1}	654
PINNMamba	32×1	285 763	Dual-FD	9.63×10^{-1}	415
ProPINN	32×3	15 030	AD	6.06×10^{-2}	75
ProPINN	32×3	15 030	Dual-FD	5.69×10^{-2}	65
FPINN	200×4	123 241	AD	2.65×10^{-2}	60
FPINN	200×4	123 241	Dual-FD	8.95×10^{-1}	74
PIG	G=800, k=4, 16×2	16 097	AD	9.40×10^{-1}	764
PIG	G=800, k=4, 16×2	16 097	Dual-FD	9.59×10^{-1}	146
IFeF	K=800, 64×6	22 657	AD	2.00×10^{-2}	503
IFeF	K=800, 64×6	22 657	Dual-FD	2.27×10^{-2}	252
Kovasznay					
StdPINN	32×6	4 419	AD	5.22×10^{-4}	149
StdPINN	32×6	4 419	Dual-FD	5.97×10^{-4}	97
QRes	14×7	2 957	AD	4.64×10^{-4}	431
QRes	14×7	2 957	Dual-FD	7.16×10^{-4}	151
sf-PINN	512×4	528 387	AD	6.50×10^{-4}	747
sf-PINN	512×4	528 387	Dual-FD	1.71×10^{-3}	272
PIKAN	$2-32-32-3$	17 760	AD	5.93×10^{-4}	8477
PIKAN	$2-32-32-3$	17 760	Dual-FD	5.71×10^{-4}	7043
PINNsFormer	d_m=32, d_h=128, N=1	194 619	AD	1.38×10^{-3}	1759
PINNsFormer	d_m=32, d_h=128, N=1	194 619	Dual-FD	4.00×10^{-3}	339
PINNMamba	32×1	286 789	AD	7.06×10^{-3}	6147
PINNMamba	32×1	286 789	Dual-FD	1.57×10^{-2}	1008
ProPINN	16×3	13 916	AD	7.82×10^{-4}	509
ProPINN	16×3	13 916	Dual-FD	1.22×10^{-3}	228
FPINN	400×4	486 471	AD	1.51×10^{-4}	1045
FPINN	400×4	486 471	Dual-FD	1.54×10^{-4}	513
PIG	G=800, k=4, 16×2	16 131	AD	1.27×10^{-4}	7634
PIG	G=800, k=4, 16×2	16 131	Dual-FD	1.48×10^{-4}	695
IFeF	K=800, 64×6	25 987	AD	1.67×10^{-2}	2904
IFeF	K=800, 64×6	25 987	Dual-FD	1.71×10^{-2}	1031
Taylor–Green					
StdPINN	64×6	17 091	AD	4.66×10^{-3}	356
StdPINN	64×6	17 091	Dual-FD	4.74×10^{-3}	167
QRes	14×7	2 985	AD	1.53×10^{-2}	1049
QRes	14×7	2 985	Dual-FD	1.48×10^{-2}	371
sf-PINN	512×4	528 899	AD	5.60×10^{-3}	1462
sf-PINN	512×4	528 899	Dual-FD	4.87×10^{-3}	814
PIKAN	$3-32-32-3$	18 240	AD	8.44×10^{-1}	3390
PIKAN	$3-32-32-3$	18 240	Dual-FD	8.44×10^{-1}	2591
PINNsFormer	d_m=32, d_h=512, N=1	454 619	AD	9.12×10^{-2}	2616
PINNsFormer	d_m=32, d_h=512, N=1	454 619	Dual-FD	9.34×10^{-2}	693
PINNMamba	32×1	286 821	AD	7.35×10^{-3}	8184
PINNMamba	32×1	286 821	Dual-FD	1.06×10^{-2}	1845

Continued on next page

Table 13 (continued)

Arch.	Capacity	Params	Operator	Best L_2	Wall (s)
FPINN	200×4	123 479	AD	7.97×10^{-3}	851
FPINN	200×4	123 479	Dual-FD	6.93×10^{-3}	361
PIG	G=800, k=4, 16×2	22 531	Dual-FD	3.83×10^{-1}	718
IFeF	K=800, 64×6	26 051	AD	3.10×10^{-1}	5778
IFeF	K=800, 64×6	26 051	Dual-FD	3.25×10^{-1}	1699
Burgers 2D					
StdPINN	32×5	3 362	AD	2.79×10^{-3}	88
StdPINN	32×5	3 362	Dual-FD	4.73×10^{-3}	76
QRes	14×7	2 970	AD	2.96×10^{-3}	431
QRes	14×7	2 970	Dual-FD	3.08×10^{-3}	203
sf-PINN	32×4	2 306	AD	3.31×10^{-3}	109
sf-PINN	32×4	2 306	Dual-FD	3.31×10^{-3}	91
PIKAN	$3-32-32-2$	17 760	AD	4.35×10^{-2}	2105
PIKAN	$3-32-32-2$	17 760	Dual-FD	4.35×10^{-2}	1778
PINNsFormer	d_m=32, d_h=512, N=1	454 106	AD	4.05×10^{-2}	1189
PINNsFormer	d_m=32, d_h=512, N=1	454 106	Dual-FD	4.10×10^{-2}	359
PINNMamba	32×1	286 308	AD	2.08×10^{-2}	7353
PINNMamba	32×1	286 308	Dual-FD	1.89×10^{-2}	1684
FPINN	200×4	123 464	AD	1.93×10^{-3}	493
FPINN	200×4	123 464	Dual-FD	1.91×10^{-3}	249
PIG	G=800, k=4, 16×2	22 514	AD	4.40×10^{-2}	1212
PIG	G=800, k=4, 16×2	22 514	Dual-FD	4.40×10^{-2}	328
IFeF	K=800, 64×6	24 386	AD	3.47×10^{-2}	2947
IFeF	K=800, 64×6	24 386	Dual-FD	3.54×10^{-2}	999
Burgers 1D					
StdPINN	20×9	3 021	AD	6.54×10^{-4}	20
StdPINN	20×9	3 021	Dual-FD	8.57×10^{-4}	17
QRes	14×7	2 927	AD	6.42×10^{-4}	129
QRes	14×7	2 927	Dual-FD	1.43×10^{-3}	65
sf-PINN	512×4	527 361	AD	1.22×10^{-3}	97
sf-PINN	512×4	527 361	Dual-FD	1.98×10^{-3}	139
PIKAN	$2-32-32-1$	16 800	AD	1.03×10^{-4}	1035
PIKAN	$2-32-32-1$	16 800	Dual-FD	1.76×10^{-4}	1794
PINNsFormer	d_m=32, d_h=128, N=1	194 361	AD	5.77×10^{-4}	287
PINNsFormer	d_m=32, d_h=128, N=1	194 361	Dual-FD	1.40×10^{-3}	227
PINNMamba	32×1	285 763	AD	6.75×10^{-4}	2005
PINNMamba	32×1	285 763	Dual-FD	1.68×10^{-3}	883
ProPINN	32×3	15 030	AD	3.72×10^{-4}	155
ProPINN	32×3	15 030	Dual-FD	1.25×10^{-3}	152
FPINN	100×4	31 641	AD	2.52×10^{-4}	86
FPINN	100×4	31 641	Dual-FD	1.82×10^{-4}	64
PIG	G=200, k=4, 16×2	4 097	AD	2.98×10^{-4}	125
PIG	G=200, k=4, 16×2	4 097	Dual-FD	3.18×10^{-4}	58
IFeF	K=800, 64×6	22 657	AD	1.05×10^{-2}	1173
IFeF	K=800, 64×6	22 657	Dual-FD	1.39×10^{-2}	622

Best L_2 and wall-clock time are medians over the available seeds, whose sets may differ between operators. Wall-clock time covers the full run and is distinct from the paired time-to-threshold statistics in App. D.3.

D.3 Paired time to threshold

A row exists only where both operators reached the threshold on the same seed, which is the pairing rule of Sec. 4. τ is the median wall-clock time to first cross the threshold over the n paired seeds.

Table 14: Paired time to threshold for configurations in which both operators reached the threshold.

Problem	Arch.	ε	n	τ_{AD} (s)	τ_{Dual} (s)	Speedup
Schrödinger	StdPINN	0.01	3	915.8	460.8	1.99×
Schrödinger	StdPINN	0.005	3	915.8	460.8	1.99×
Schrödinger	StdPINN	0.002	1	937.8	463.0	2.03×
Schrödinger	QRes	0.01	1	289.2	132.3	2.19×
Schrödinger	QRes	0.005	1	322.2	144.5	2.23×
Schrödinger	PINNsFormer	0.01	1	934.9	185.9	5.03×
Schrödinger	ProPINN	0.01	2	312.9	140.6	2.23×

Problem	Arch.	ε	n	τ_{AD} (s)	τ_{Dual} (s)	Speedup
Schrödinger	FPINN	0.01	2	460.8	227.1	2.03×
Schrödinger	FPINN	0.005	2	486.3	239.8	2.03×
Schrödinger	FPINN	0.002	1	511.4	245.4	2.08×
Schrödinger	PIG	0.01	3	1212.2	125.9	9.63×
Schrödinger	PIG	0.005	3	1616.7	126.3	12.80×
Schrödinger	PIG	0.002	2	1637.8	181.9	9.00×
Schrödinger	PIG	0.001	2	3431.1	319.4	10.74×
Convection	ProPINN	0.1	1	130.7	135.4	0.97×
Reaction 1D	StdPINN	0.1	1	14.6	15.5	0.94×
Reaction 1D	QRes	0.1	3	64.5	82.2	0.78×
Reaction 1D	QRes	0.05	2	141.6	93.4	1.52×
Reaction 1D	PINNsFormer	0.1	3	53.2	56.0	0.95×
Reaction 1D	PINNsFormer	0.05	1	99.2	74.0	1.34×
Reaction 1D	ProPINN	0.1	2	37.8	36.7	1.03×
Reaction 1D	FPINN	0.1	1	27.7	20.5	1.35×
Reaction 1D	FPINN	0.05	1	38.7	32.6	1.19×
Reaction 1D	IFeF	0.1	3	32.1	23.7	1.36×
Reaction 1D	IFeF	0.05	3	110.5	35.9	3.08×
Kovasznay	StdPINN	0.01	3	93.5	45.6	2.05×
Kovasznay	StdPINN	0.005	3	107.1	59.3	1.81×
Kovasznay	StdPINN	0.001	3	121.0	68.4	1.77×
Kovasznay	QRes	0.01	3	174.5	54.6	3.20×
Kovasznay	QRes	0.005	3	324.9	100.7	3.23×
Kovasznay	QRes	0.001	2	343.5	112.0	3.07×
Kovasznay	sf-PINN	0.01	3	154.6	85.0	1.82×
Kovasznay	sf-PINN	0.005	3	233.3	126.2	1.85×
Kovasznay	sf-PINN	0.001	1	389.4	199.7	1.95×
Kovasznay	PIKAN	0.01	3	929.5	564.8	1.65×
Kovasznay	PIKAN	0.005	3	1319.0	705.8	1.87×
Kovasznay	PIKAN	0.001	3	4710.8	3103.7	1.52×
Kovasznay	PINNsFormer	0.01	3	294.9	99.3	2.97×
Kovasznay	PINNsFormer	0.005	2	574.9	159.2	3.61×
Kovasznay	ProPINN	0.01	3	100.3	51.7	1.94×
Kovasznay	ProPINN	0.005	3	100.4	52.5	1.91×
Kovasznay	ProPINN	0.001	1	297.9	129.2	2.31×
Kovasznay	FPINN	0.01	3	486.5	309.9	1.57×
Kovasznay	FPINN	0.005	3	812.6	389.5	2.09×
Kovasznay	FPINN	0.001	3	908.1	436.9	2.08×
Kovasznay	PIG	0.01	3	389.1	40.2	9.69×
Kovasznay	PIG	0.005	3	389.1	40.2	9.69×
Kovasznay	PIG	0.001	3	774.6	70.3	11.02×
Taylor–Green	StdPINN	0.01	3	219.6	113.8	1.93×
Taylor–Green	StdPINN	0.005	2	352.3	153.5	2.29×
Taylor–Green	sf-PINN	0.01	3	730.0	403.3	1.81×
Taylor–Green	FPINN	0.01	2	660.5	285.4	2.31×
Burgers 2D	StdPINN	0.01	3	30.0	36.6	0.82×
Burgers 2D	StdPINN	0.005	1	30.0	26.0	1.16×
Burgers 2D	QRes	0.01	2	270.3	109.7	2.46×
Burgers 2D	QRes	0.005	2	356.4	137.6	2.59×
Burgers 2D	sf-PINN	0.01	3	38.2	25.3	1.51×
Burgers 2D	sf-PINN	0.005	3	38.5	35.6	1.08×
Burgers 2D	FPINN	0.01	3	163.4	83.7	1.95×
Burgers 2D	FPINN	0.005	2	223.5	121.5	1.84×
Burgers 1D	StdPINN	0.01	3	9.8	11.3	0.86×
Burgers 1D	StdPINN	0.005	3	9.8	11.3	0.86×
Burgers 1D	StdPINN	0.001	1	9.6	11.1	0.86×
Burgers 1D	QRes	0.01	3	11.4	16.8	0.68×
Burgers 1D	QRes	0.005	3	34.2	22.5	1.52×
Burgers 1D	QRes	0.001	1	115.0	63.1	1.82×
Burgers 1D	sf-PINN	0.01	3	29.1	33.1	0.88×
Burgers 1D	sf-PINN	0.005	3	54.8	61.0	0.90×
Burgers 1D	PIKAN	0.01	3	60.9	95.2	0.64×
Burgers 1D	PIKAN	0.005	3	61.5	95.2	0.65×

Problem	Arch.	ε	n	τ_{AD} (s)	τ_{Dual} (s)	Speedup
Burgers 1D	PIKAN	0.001	3	61.5	189.7	0.32×
Burgers 1D	PINNsFormer	0.01	3	102.4	78.8	1.30×
Burgers 1D	PINNsFormer	0.005	3	102.4	78.8	1.30×
Burgers 1D	PINNMamba	0.01	3	416.4	245.9	1.69×
Burgers 1D	PINNMamba	0.005	3	416.4	245.9	1.69×
Burgers 1D	PINNMamba	0.001	1	823.4	247.4	3.33×
Burgers 1D	ProPINN	0.01	3	56.2	53.1	1.06×
Burgers 1D	ProPINN	0.005	3	56.2	53.1	1.06×
Burgers 1D	ProPINN	0.001	1	76.0	50.8	1.50×
Burgers 1D	FPINN	0.01	3	7.6	9.8	0.78×
Burgers 1D	FPINN	0.005	3	15.2	15.0	1.01×
Burgers 1D	FPINN	0.001	3	38.7	25.3	1.53×
Burgers 1D	PIG	0.01	3	35.8	17.0	2.11×
Burgers 1D	PIG	0.005	3	35.8	17.0	2.11×
Burgers 1D	PIG	0.001	3	35.8	17.0	2.11×

D.4 Configurations without acceptable results

Table 15 reports the available records for 16 configurations marked n/c in Table 1. Best L_2 is the lowest recorded relative error across the available operator runs; acceptability also requires recovery of the physical features (App. B.2). Capacity provenance codes follow App. D.1.

Table 15: Available training records for configurations marked n/c in Table 1.

Problem	Arch.	Capacity	Best L_2	Prov.	Operators run
Schrödinger	PIKAN	$2-32-32-2$	4.23×10^{-1}	**T**	AD, Dual-FD
Schrödinger	IFeF	K=800, 64×6	1.02×10^{-1}	**P**	AD, Dual-FD
Convection	QRes	256×4	8.80×10^{-1}	**R**	AD, Dual-FD
Convection	sf-PINN	512×4	9.56×10^{-1}	**R**	AD, Dual-FD
Convection	PIKAN	$2-5-5-1$	1.00×10^{0}	**R**	AD, Dual-FD
Convection	PINNMamba	32×1	1.00×10^{0}	**R**	AD, Dual-FD
Convection	FPINN	200×4	9.40×10^{-1}	**P**	AD, Dual-FD
Convection	PIG	G=800, k=4, 16×2	6.71×10^{-1}	**P**	AD, Dual-FD
Convection	IFeF	K=800, 64×6	9.61×10^{-1}	**R**	AD, Dual-FD
Reaction 1D	PIKAN	$2-5-5-1$	4.31×10^{-1}	**R**	AD, Dual-FD
Taylor–Green	PIKAN	$3-32-32-3$	8.36×10^{-1}	**T**	AD, Dual-FD
Taylor–Green	PIG	G=800, k=4, 16×2	2.19×10^{-1}	**P**	Dual-FD
Burgers 2D	PIKAN	$3-32-32-2$	4.33×10^{-2}	**T**	AD, Dual-FD
Burgers 2D	PINNsFormer	d_m=32, d_h=512, N=1	3.73×10^{-2}	**P**	AD, Dual-FD
Burgers 2D	PIG	G=800, k=4, 16×2	4.39×10^{-2}	**P**	AD, Dual-FD
Burgers 2D	IFeF	K=800, 64×6	3.18×10^{-2}	**P**	AD, Dual-FD

The layer count of the reference implementations is the number of Linear layers, so the layer list $[2, 100, 100, 100, 100, 2]$ is 100×5 and not 100×4.

D.5 Physical-field comparisons for the accuracy-matched benchmarks

The plates below show the learned fields of every matrix run with a saved model, one page per problem and one row per architecture, at seed 0. Within a row the left panel is Dual-FD, the middle panel is AD and the right panel is their difference. The first two share the reference colour scale so the two solutions are comparable by eye, and each carries that run's final relative L_2. The difference panel has its own symmetric scale, since it is normally orders of magnitude smaller, and carries its peak absolute value. Steady Kovasznay is shown as $u(x, y)$, Taylor–Green and Burgers 2D at the final time, and the one-dimensional problems as profiles at the final time with the reference drawn in grey beneath, using $|h|$ for Schrödinger. PIG has no Taylor–Green pair in this results tree, and on

Burgers 2D the PIKAN and PIG runs leave a field without structure under either operator, so those three rows are omitted.

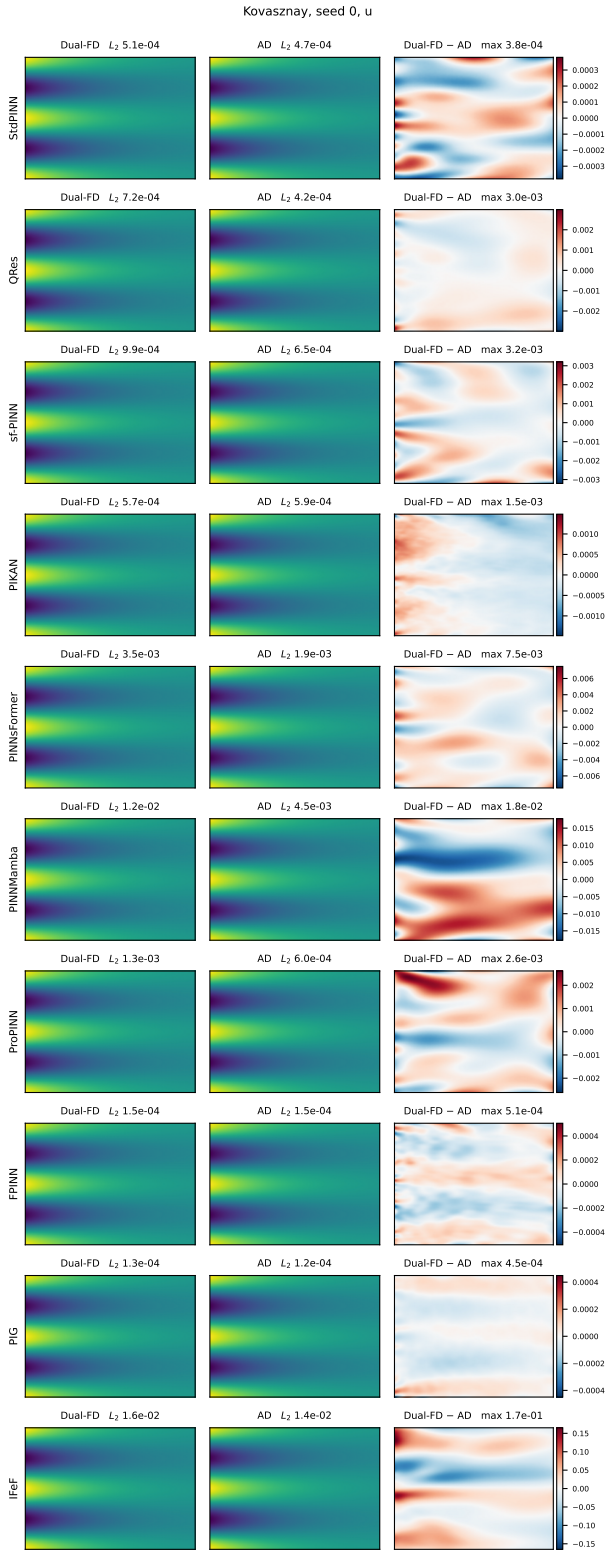


Figure 6: Kovasznay fields at seed 0. Rows are architectures; left Dual-FD, middle AD, right their difference. The first two panels of a row share the reference colour scale and carry the run's final relative L_2, and the difference panel carries its peak absolute value.

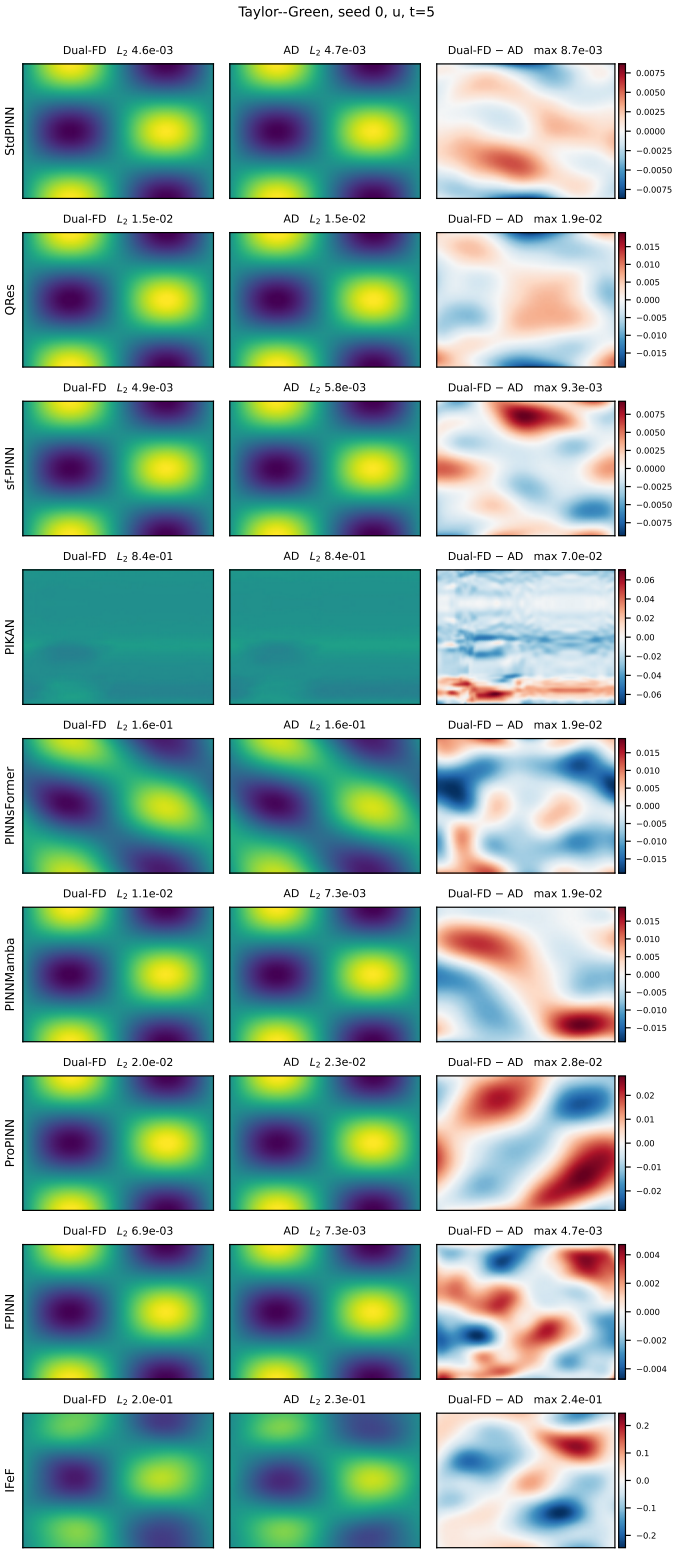


Figure 7: Taylor–Green fields at seed 0. Rows are architectures; left Dual-FD, middle AD, right their difference. The first two panels of a row share the reference colour scale and carry the run's final relative L_2, and the difference panel carries its peak absolute value.

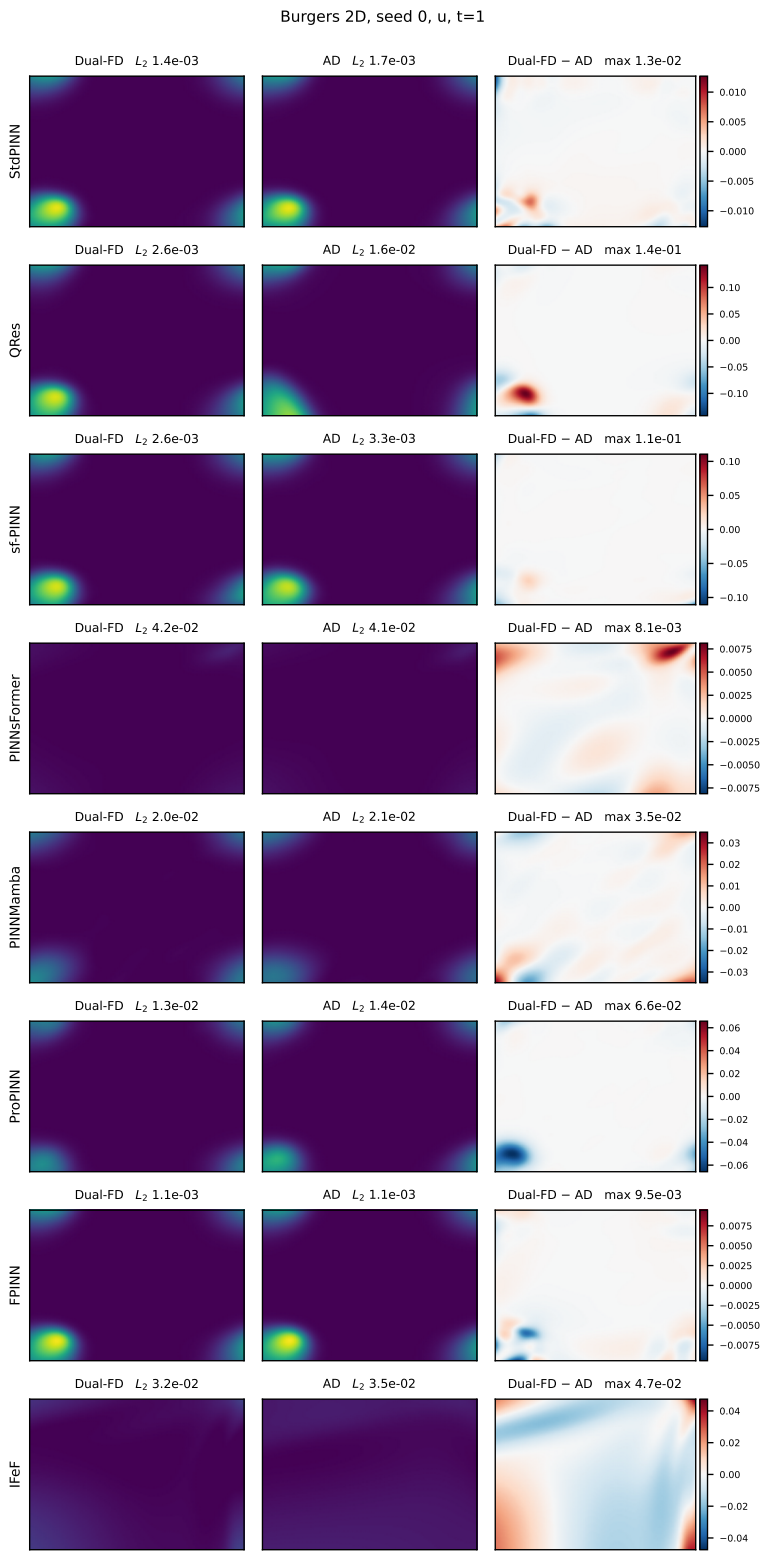


Figure 8: Burgers 2D fields at seed 0. Rows are architectures; left Dual-FD, middle AD, right their difference. The first two panels of a row share the reference colour scale and carry the run's final relative L_2, and the difference panel carries its peak absolute value.

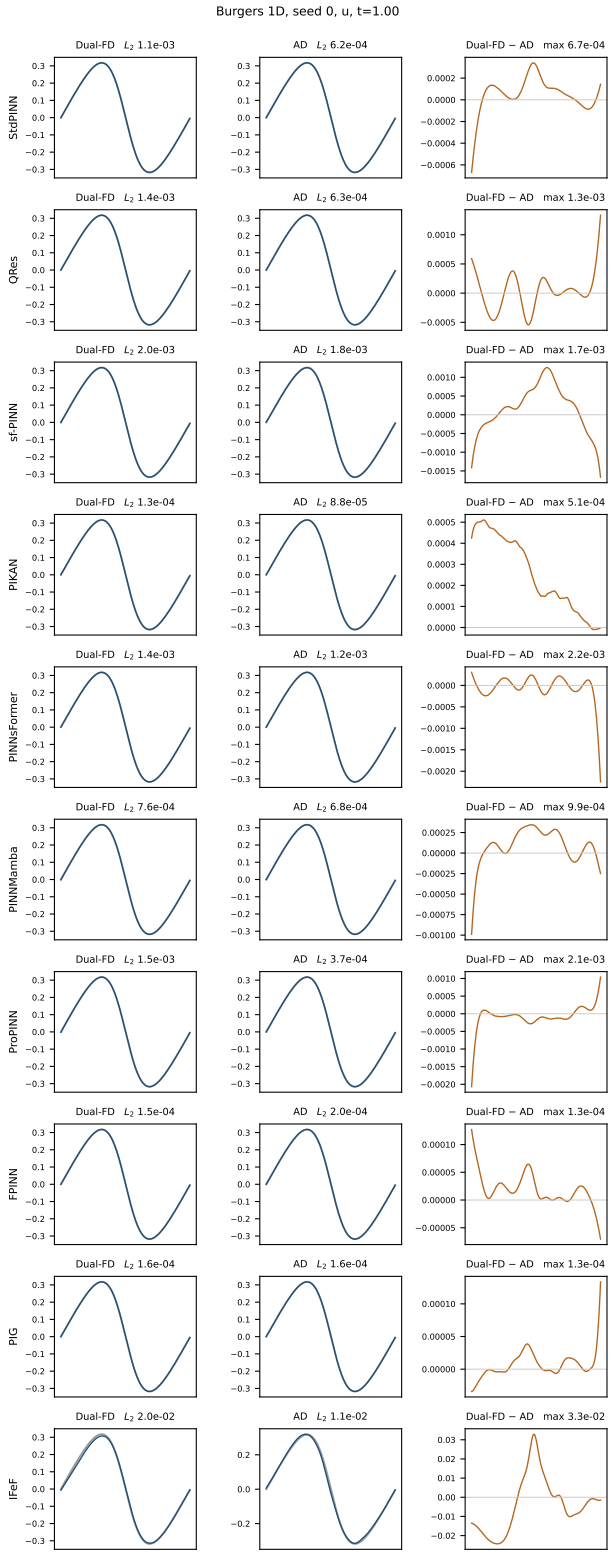


Figure 9: Burgers 1D fields at seed 0. Rows are architectures; left Dual-FD, middle AD, right their difference. The first two panels of a row share the reference colour scale and carry the run's final relative L_2, and the difference panel carries its peak absolute value.

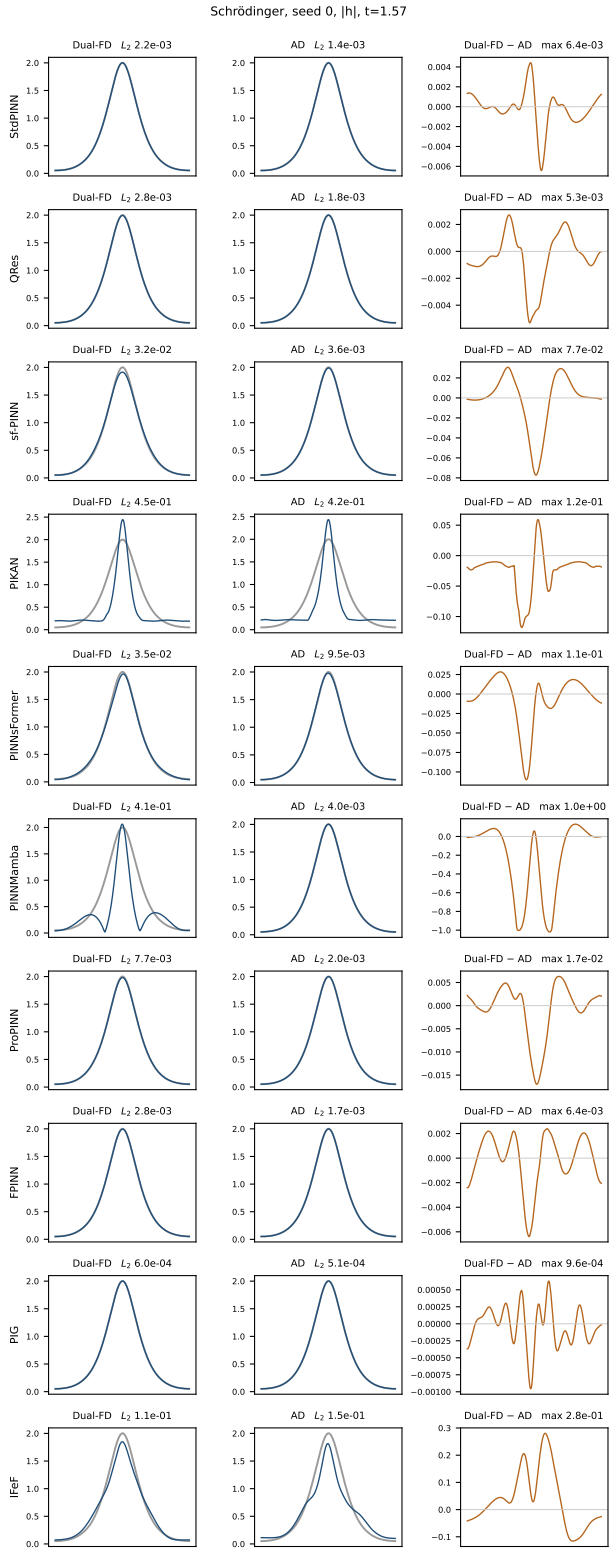


Figure 10: Schrödinger fields at seed 0. Rows are architectures; left Dual-FD, middle AD, right their difference. The first two panels of a row share the reference colour scale and carry the run's final relative L_2, and the difference panel carries its peak absolute value.

E LONG-HORIZON SOLVING

This appendix defines the evaluation metrics and training settings for Sec. 4.3, and provides whole-horizon comparisons. Training starts from analytic initial conditions and uses no reference data; reference solutions are used only for evaluation.

E.1 EXPERIMENTAL SETTINGS AND EVALUATION METRICS

Capacity and training settings. Window-based methods are matched to RD-PINN in the parameter budget per network; whole-horizon networks retain their published model sizes. RD-PINN uses discrete-time windows for dissipative and transport-dominated problems and continuous-time windows for Schrödinger. The cylinder and Taylor–Green runs use $\Delta t = 0.25$ unless stated otherwise. A window stops when all constraints fall below 10^{-5}, no improvement is observed for 800 epochs, or the 5000-epoch cap is reached; its network is then frozen. Plateau or cap stopping does not imply that the constraint tolerance was met. Random seeds vary network initialization and collocation resampling.

Field error and runtime. Let u_θ and u_{ref} denote the predicted and reference fields on the same evaluation points. Mean relative L_2 error averages the snapshot errors,

$$\bar{\varepsilon} = \frac{1}{N_t}\sum_{k=1}^{N_t}\frac{\|u_\theta(\cdot,t_k)-u_{\mathrm{ref}}(\cdot,t_k)\|_2}{\|u_{\mathrm{ref}}(\cdot,t_k)\|_2}, \qquad \varepsilon_{\mathrm{final}} = \frac{\|u_\theta(\cdot,t_{N_t})-u_{\mathrm{ref}}(\cdot,t_{N_t})\|_2}{\|u_{\mathrm{ref}}(\cdot,t_{N_t})\|_2}. \tag{16}$$

Both errors are reported as percentages. Cylinder errors use the aligned evaluation interval defined below. Table 2 and Table 16 report three-seed means. Runtime is the mean wall-clock training time, and S is the mean of the paired AD/Dual-FD runtime ratios across the three seeds. These full-run ratios differ from the time-to-threshold comparisons in Sec. 4.1.

EDNN (Du & Zaki, 2021) instead obtains the time derivative of the network parameters by least-squares projection and then advances those parameters. It has no corresponding paired operator substitution in residual training, so its runtime is reported but it is excluded from the paired speedup statistics.

Relative L_2 error can remain small when a large background masks missing local fluctuations. We therefore also report the following problem-specific amplitude ratios as the physical index Φ.

Spatial fluctuation amplitude. For Burgers 1D, Taylor–Green and Schrödinger, the physical index compares spatial RMS magnitudes after removing each field's own spatial mean. On the same N_x evaluation points, define

$$\bar{u}_j(t_k) = \frac{1}{N_x}\sum_{i=1}^{N_x} u_j(x_i,t_k), \qquad R_j(t_k) = \left[\frac{1}{N_x}\sum_{i=1}^{N_x}|u_j(x_i,t_k)-\bar{u}_j(t_k)|^2\right]^{1/2}, \tag{17}$$

where $j \in \{\theta, \mathrm{ref}\}$ denotes the prediction or reference field and $|\cdot|$ denotes the scalar modulus or vector norm, as appropriate. With $\mathcal{K} = \{k : t_k \geq T/2\}$, the reported percentage is

$$\Phi_{\mathrm{RMS}} = \frac{100}{|\mathcal{K}|}\sum_{k\in\mathcal{K}}\frac{R_\theta(t_k)}{R_{\mathrm{ref}}(t_k)}. \tag{18}$$

This averages the per-snapshot ratios, rather than taking a ratio of time-averaged RMS values.

Burgers 2D: temporal fluctuation amplitude. For Burgers 2D, choose the evaluation node nearest the centre probe,

$$\mathbf{x}_p = \arg\min_{\mathbf{x}_i\in\mathcal{X}}\|\mathbf{x}_i-(1,1)\|_2, \qquad q_j(t_k) = u_j(\mathbf{x}_p,t_k), \quad j\in\{\theta,\mathrm{ref}\}. \tag{19}$$

The reference is interpolated to the same evaluation points. On the 48×48 grid, $\mathbf{x}_p \simeq (1.0213, 1.0213)$. Retain the common times at which the reference is available, let T_B be their

largest value, and define $\mathcal{K}_B = \{k : t_k \geq T_B/2\}$. The reported index is the ratio of the peak-to-trough excursions of the two signals, sampled at the same times,

$$\Phi_{\mathrm{B2}} = 100\,\frac{\max_{k\in\mathcal{K}_B} q_\theta(t_k) - \min_{k\in\mathcal{K}_B} q_\theta(t_k)}{\max_{k\in\mathcal{K}_B} q_{\mathrm{ref}}(t_k) - \min_{k\in\mathcal{K}_B} q_{\mathrm{ref}}(t_k)}. \tag{20}$$

Prediction and reference are sampled at the same times without a phase shift. This amplitude ratio does not measure agreement in pulse timing.

Cylinder: temporal fluctuation amplitude and aligned field error. For the cylinder, the implementation first checks that the evaluation points are a subset of the DNS reference-grid nodes, so the prediction and reference are compared at the same node. The probe is the node nearest $(2, 0)$, located at approximately $\mathbf{x}_w = (1.961, -0.032)$, and the signal is the transverse velocity $q_j(t) = v_j(\mathbf{x}_w, t)$. The prediction is read at its snapshots in the last ten time units and the reference at the same times shifted onto the reference record,

$$\mathcal{K}_\theta = \{k : T - 10 \leq t_{\theta,k} \leq T\}, \qquad t_k^{\mathrm{ref}} = t_{\theta,k} - (T - 10) + \tau^*, \tag{21}$$

where $\tau^* \in [0, 10]$ is the phase shift selected for the aligned field error below, so both signals cover the same part of the shedding cycle with the same number of samples. The reported index is

$$\Phi_{\mathrm{cyl}} = 100\,\frac{\max_{k\in\mathcal{K}_\theta} q_\theta(t_{\theta,k}) - \min_{k\in\mathcal{K}_\theta} q_\theta(t_{\theta,k})}{\max_{k\in\mathcal{K}_\theta} q_{\mathrm{ref}}(t_k^{\mathrm{ref}}) - \min_{k\in\mathcal{K}_\theta} q_{\mathrm{ref}}(t_k^{\mathrm{ref}})}. \tag{22}$$

For both probe indices, 100% means that the predicted and reference peak-to-trough amplitudes agree. The reference is a DNS trajectory on the shedding attractor covering $t \in [0, 20]$. The prediction interval is $[10, 20]$ for whole-horizon networks ($T = 20$) and $[40, 50]$ for the temporal methods and RD-PINN ($T = 50$). For cylinder field errors, both mean and final L_2 are taken over the last ten time units of the run after aligning it in time to the best-matching ten-unit slice of the reference record, and Φ_{cyl} uses the same shift τ^*.

Whole-horizon cylinder runs store snapshots every two time units, leaving only six samples in the evaluation interval. The reference is read at the same six shifted times, so both amplitudes are resolved equally coarsely, but an extreme that falls between snapshots is missed on both sides. The number of snapshots, $n_{\mathrm{snap}} = |\mathcal{K}_\theta|$, is recorded with the metric; window-based runs use denser snapshots. The problem-specific indices above are reported as Φ in Table 2; 100% denotes agreement in the measured fluctuation magnitude, without establishing agreement in the full waveform or spatial field.

E.2 Supplementary comparisons

A small PDE residual does not by itself establish agreement with the solution selected by the initial and boundary conditions. Trivial or steady solutions may have zero residual, and training can remain near such states rather than propagate the initial condition through the domain (Rohrhofer et al., 2022; Daw et al., 2022). Whole-horizon training can also reduce late-time residuals before earlier states are accurately resolved, motivating causal training strategies (Wang et al., 2024). Loss ill-conditioning and spectral bias provide additional possible sources of difficulty (Krishnapriyan et al., 2021; Wang et al., 2022; Rahaman et al., 2019). These mechanisms motivate temporal decomposition; the comparisons in Sec. 4.3 do not isolate their individual contributions.

Whole-horizon networks jointly optimize one parameter set over the full space–time domain. Table 16 complements the temporal-method comparison in Table 2, using the metrics and three-seed averaging defined above. Paired entries are ordered Dual-FD / AD; bold marks the best Dual-FD result in each column (lowest L_2, highest S, or Φ closest to 100%). Cylinder runs end at $T = 20$ and are evaluated using the six stored snapshots in $t \in [10, 20]$.

Fig. 5 shows the cylinder speed field at $t = 10$, 30 and 50 and the transverse-velocity signal at a fixed location. These views complement the scalar metrics by showing the development of wake asymmetry and oscillations. Figs. 11 and 12 give the field views for the cylinder, the one long-horizon problem whose physical feature is a sustained oscillation rather than a decay. Both show the transverse velocity of the Dual-FD arm at seed 0 on one colour scale. A run started from uniform inflow selects its own shedding phase, so the reference record is shifted in time by τ, chosen to minimise the mean relative L_2 of (u, v) over the final ten time units, which is the shift used for the reported cylinder errors.

Table 16: Whole-horizon networks on the long-horizon problems of Table 2. Paired entries are ordered Dual-FD / AD; speedup is reported as a single ratio.

Framework / scheme	Metric	Burgers 1D (ν=0.1) $T = 2$	Burgers 2D $T = 8$	Taylor–Green $T = 20$	Schrödinger $T = 1.5$	Cylinder $T = 20$
StdPINN	Mean L_2 (%)	1.6 / 0.8	2.3 / 2.4	59.8 / 59.8	18.8 / 18.9	**44.6** / 43.5
(Raissi et al., 2019)	Final L_2 (%)	3.4 / 1.6	**1.4** / 1.3	84.1 / 84.1	37.6 / 38.0	**46.6** / 45.3
	Speedup S	1.16	1.91	2.05	2.04	2.08
	Physics Φ	99.7 / 100.0	1.7 / 2.4	25.6 / 25.6	99.4 / 99.4	0.5 / 0.5
QRes	Mean L_2 (%)	**0.4** / 0.2	**2.2** / 2.3	71.9 / 56.4	**9.0** / 10.2	46.0 / 46.2
(Bu & Karpatne, 2021)	Final L_2 (%)	**0.5** / 0.4	**1.4** / 1.4	91.0 / 84.4	**18.8** / 21.1	47.3 / 47.4
	Speedup S	1.52	2.26	2.69	1.76	1.97
	Physics Φ	**99.8** / 100.0	2.6 / 1.9	16.2 / 31.5	**99.6** / 99.6	0.3 / 0.1
PINNsFormer	Mean L_2 (%)	2.6 / 12.6	2.6 / 2.8	57.1 / 63.9	92.8 / 98.1	53.6 / 53.2
(Zhao et al., 2023)	Final L_2 (%)	5.9 / 22.5	1.6 / 1.8	80.5 / 85.3	103.0 / 142.8	55.2 / 57.1
	Speedup S	**2.28**	**2.74**	**3.05**	**3.71**	**7.72**
	Physics Φ	100.7 / 86.5	**11.9** / 15.3	**40.9** / 30.1	61.4 / 76.8	**0.8** / 0.6
FPINN	Mean L_2 (%)	0.5 / 0.5	2.7 / 2.5	**52.0** / 59.4	14.3 / 15.2	50.7 / 45.6
(Wu et al., 2025b)	Final L_2 (%)	1.0 / 1.2	1.7 / 1.6	**67.9** / 76.5	29.0 / 30.4	52.8 / 47.7
	Speedup S	1.60	1.41	1.87	1.70	1.55
	Physics Φ	100.3 / 99.9	2.4 / 1.0	38.8 / 30.1	**99.6** / 99.6	0.2 / 0.6

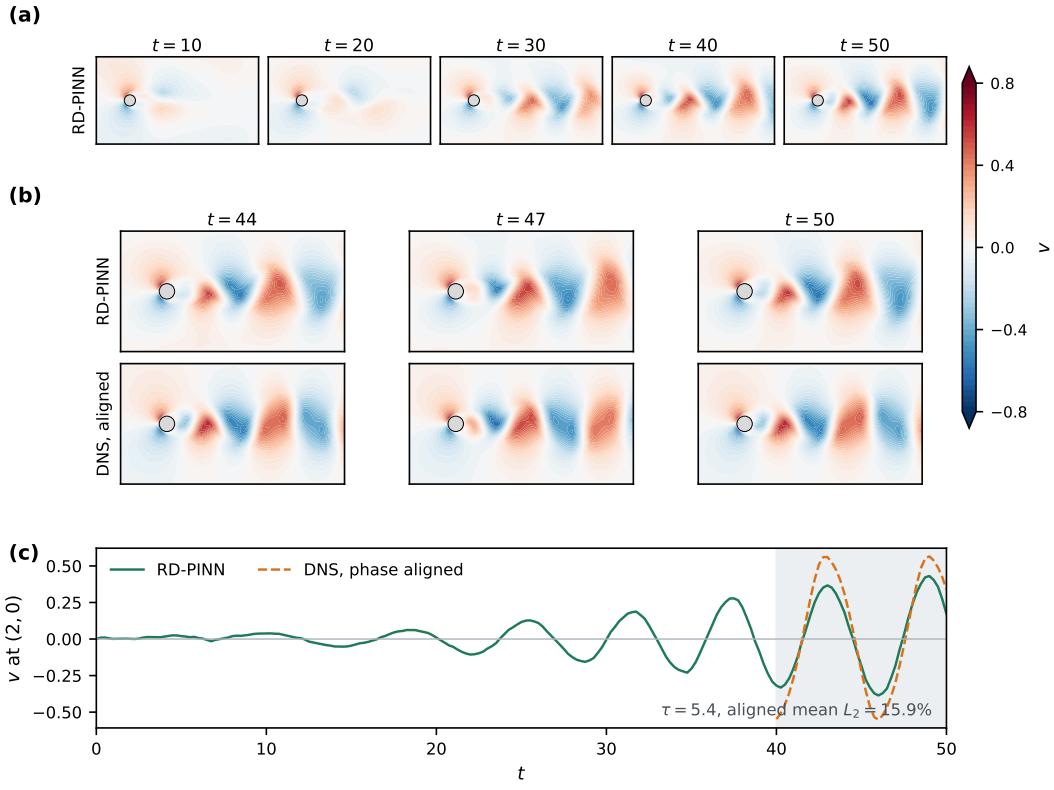


Figure 11: RD-PINN on the cylinder without reference training data, Dual-FD, seed 0. (a) Transverse velocity at five times, almost symmetric at $t = 10$ and carrying an alternating vortex street from $t = 30$ onwards. (b) The final three snapshots above the phase-aligned reference. (c) The probe signal at $(2, 0)$, with the shaded band marking the final ten time units over which the physical index is read. Here $\tau = 5.4$ and the aligned mean relative L_2 is 15.9%.

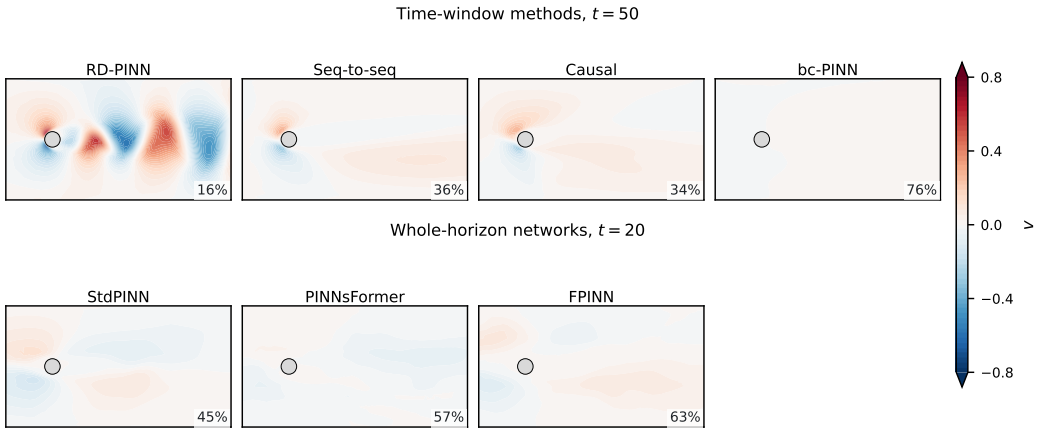


Figure 12: Cylinder wake of every method that reached its horizon, Dual-FD, seed 0, on the colour scale of Fig. 11. Upper row, the time-window methods at $t = 50$; lower row, the whole-horizon networks, which are trained to $t = 20$, at that time. Only RD-PINN develops a vortex street. The number in each panel is that run's mean relative L_2 against the reference over its own final ten time units, with τ solved separately for each run so that a different shedding phase is not charged as error.