# INTEGRAL-FORM REFINEMENT SUPPRESSES ERROR ACCUMULATION IN NEURAL OPERATOR ROLLOUTS

**Anonymous authors**
Paper under double-blind review

## ABSTRACT

Neural operators (NOs) forecast PDE trajectories rapidly, but in autoregressive rollout each window becomes the next input and errors compound until the forecast is unusable. We refine each predicted window against the governing equation before feeding it forward, with the NO frozen. The residual is the integral (Duhamel) form of the equation, or the discrete balance of a numerical scheme where no suitable linear part exists. This lowers rollout error by 40–72% on Burgers, 83% on turbulent Navier–Stokes and 33–61% on the highly nonlinear degenerate Richards equation of variably saturated flow. The improvement is substantial for the tested NOs, and refinement largely equalizes them, though our primitive NO keeps an edge at small correction budgets. More importantly, the refinement stops rollout error from compounding on non-chaotic systems. The uncorrected Richards error grows 2.7-fold over a 38-day forecast, whereas the corrected error settles three to thirteen times lower. In turbulent flow, where chaos forbids any bound, the error still grows but stays 56 times lower. The integral-form residual is critical: at the coarse frame spacing of NOs, both the standard pointwise residual and a one-step scheme residual are dominated by their own time-discretization error, and refining against them worsens the rollout or diverges. On Burgers, our training-free correction also outperforms learned refiners, which need per-system training. Furthermore, the benefit can be anticipated from the residual's own error, measured once per system. This simple screen correctly identifies when refinement helps. Lastly, the correction can be strategically enhanced: semi-Lagrangian transport lowers Navier–Stokes error from 0.050 to 0.0083 in 17 ms per trajectory, below what any residual-minimizing refiner can reach. Together, these results show NO rollout errors to be measurable, predictable and suppressible, and establish test-time refinement as a practical tool for long forecasts.

## 1 INTRODUCTION

Neural operators approximate the solution map of a partial differential equation and thereby replace an expensive simulation with a single network evaluation (Li et al., 2021; Lu et al., 2021; Kovachki et al., 2023). Forecasts beyond the training horizon are obtained by autoregressive rollout, in which each predicted window of frames becomes the input to the next prediction. Accuracy on a single window does not control accuracy over a rollout. A small displacement of a front, or a persistent imbalance in a conserved quantity, is inherited by the subsequent window. This error compounding is well known and can render neural operator solutions unusable (Um et al., 2020; Lippe et al., 2023; Huang & Perdikaris, 2025). In our experiments the uncorrected rollout error grows by a factor of approximately 3 to 20 from the first to the final prediction window on several tested systems (Table 3). Single-window test errors, the usual basis for comparing operators, do not reveal this.

When the governing equation is known, its residual provides a label-free signal at inference. It has been used to fine-tune network weights on each instance (Li et al., 2024), to correct predictions through a linearized problem (Cao et al., 2023; Jha, 2023; Huang & Perdikaris, 2025), and to hand a rollout over to a classical solver (Roy et al., 2025). Learned refiners instead learn the correction from data (Lippe et al., 2023; Wandel et al., 2025) (Section 2). These studies establish that equation-based correction is feasible and that it can stabilize a rollout. They leave open what a practitioner most needs to know: whether correction can hold rollout error in check across networks and dynamics;



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.

which residual carries usable signal when frames are stored many solver steps apart; and whether the benefit can be anticipated rather than discovered by trying.

Here we study *window refinement* to address these questions. After the network predicts a window, and before that window is fed back, the predicted field values are adjusted to reduce the residual of the governing equation, with the network weights frozen. The corrected window serves as the next input, which we term *corrected feedback*. The residual is the integral (mild) form of the equation over the whole window, which we find far more robust than a pointwise residual. For equations without a suitable linear part, we use the discrete balance of a trusted numerical scheme. On six systems (Appendix A), we apply the same correction to up to three architectures each, and establish the following:

**Refinement reduces rollout error and stops error compounding.** Window refinement substantially lowers rollout error developed in forecasts of different architectures, and wherever the uncorrected error compounds on a non-chaotic system, it turns the rollout into one whose error settles at a bounded level (Sections 4.1, 4.2 and 4.6).

**The form of the residual decides how refinement reduces rollout error.** At the temporal resolution at which trajectories are stored, the pointwise residual degrades the forecast and a one-step scheme residual diverges, whereas the integral form carries usable signal (Section 4.3).

**Whether refinement will help is predictable in advance.** The residual's own error on a true window, measured once per system, indicates what refinement can reach. Comparing this error with the forecast error predicts the benefit and identifies in advance the one system on which refinement is harmful (Section 4.4).

**Error-targeted corrections address residual-insensitive errors at lower computational cost.** Identifying the dominant error mode of a system yields inexpensive corrections. On turbulent Navier–Stokes, the resulting correction achieves lower error than residual descent, at a fraction of its computational cost (Sections 4.2 and 4.5).

## 2 Related work

**Neural operators and rollout error.** FNO and DeepONet learn maps between function spaces (Li et al., 2021; Lu et al., 2021; Kovachki et al., 2023); graph networks (Pfaff et al., 2021) and the Transolver family (Wu et al., 2024; Luo et al., 2025) extend operator learning to general geometries, and implicit formulations target steady problems (You et al., 2022; Marwah et al., 2023). These models are fast, but their training objective is a data misfit over one or a few windows, which constrains error accumulation under rollout only weakly. Our hypothesis is that, however carefully trained, one-shot predictions still inject error that compounds. Solver-in-the-loop training exposes the network to its own errors (Um et al., 2020; Kochkov et al., 2021), at the cost of a differentiable solver and retraining for each system. McGreivy & Hakim (2024) show that many reported speedups disappear against accuracy-matched classical baselines; we therefore report complete online timings and classical controls.

**Learned refinement.** PDE-Refiner (Lippe et al., 2023) refines each predicted state by learned iterative denoising, which extends stable rollout horizons without using equations and must be trained for each system. Metamizer (Wandel et al., 2025) learns an optimizer that drives pointwise physics losses to very low values across many PDEs. Both amortize the correction into training. Our method derives it from the equation at inference; we evaluate adaptations of both (Section 4.3).

**Equation-based correction of predictions.** The pointwise residual is the objective of physics-informed training, and PINO also fine-tunes the network on it for each test instance (Raissi et al., 2019; Li et al., 2024). This objective is known to admit small residuals with large errors (Krishnapriyan et al., 2021; Wang et al., 2022); we find the same failure when it is minimized at inference (Section 4.3). Cao et al. (2023) and Jha (2023) correct a prediction with one Newton step on a linearized variational problem, for a single boundary-value solve rather than a rollout. PhysicsCorrect (Huang & Perdikaris, 2025) applies a cached Jacobian-based correction at every step of a rollout and is the closest training-free precedent; we do not claim first use of training-free correction or of corrected feedback, and we compare with it in Section 4.3. ANCHOR (Roy et al., 2025) monitors the residual during a rollout and hands over to a classical solver when it grows, whereas we retain

and adjust the NO forecast. Our projected updates follow coarse-space, reduced-order and multigrid constructions (Zhang et al., 2024; Melchers et al., 2026; Lindsay et al., 2022; Maynard et al., 2020). Neural operators have also been used to initialize iterative solvers (Eshaghi et al., 2026; Wang et al., 2026; Oh et al., 2026; Fan et al., 2025; Hao et al., 2024; Liao et al., 2026). These studies do not compare residual formulations or vary the forecasting network under a fixed correction.

**Numerical foundations.** The integral residual rests on exponential integrators (Cox & Matthews, 2002; Kassam & Trefethen, 2005; Hochbruck & Ostermann, 2010); its fixed-point iteration draws on spectral deferred correction (Dutt et al., 2000), and the transport correction builds on semi-Lagrangian advection (Stam, 1999). Furuya et al. (2024) prove that stacked operator layers approximate Picard iterations of the mild form.

# 3 METHOD

## 3.1 PROBLEM SETTING

A trajectory, simulated by a numerical model, is stored as frames separated by multiple solver steps. A neural operator, the *backbone*, maps a window of consecutive past frames $g$ to the next window, $h = \mathcal{G}_\theta(g)$, in a single evaluation. A rollout chains these predictions: the predicted window becomes the input from which the next window is predicted, and so on to the horizon. All backbones predict the increment from the last input frame and are trained on chains of one to five windows with small input perturbations (Appendix B).

## 3.2 WINDOW REFINEMENT WITH THE INTEGRAL RESIDUAL

Consider an evolution equation $\partial_t u = Lu + N(u)$, where $L$ and $N$ are spatial operators representing, for example, diffusion and nonlinear advection, and $u$ denotes the field over a prediction window. Consistency of a candidate window $h$ with the equation can be assessed in two ways. The pointwise (or strong) form requires a time derivative, which we approximate using fourth-order finite differences across stored frames. The integral (or mild) form requires no time derivative. By the variation-of-constants formula, the solution at time $t$ is the incoming state propagated by the linear part alone, plus the accumulated nonlinear contribution:

$$\mathcal{R}(h)(t) = h(t) - e^{tL}u_0 - \int_0^t e^{(t-s)L} N(h(s))\, ds = 0. \tag{1}$$

Here $u_0$ is the last frame preceding the window and $e^{tL}$ is the solution operator of the linear part. On a periodic domain $e^{tL}$ acts on each Fourier mode, of wavenumber $k$, as multiplication by the known factor $e^{t\lambda_k}$, so it is exact and involves no time stepping; the integral is evaluated by quadrature on the stored frames.

The mild form is preferred because of its own accuracy. Frames are stored many solver steps apart, so a finite-difference time derivative carries a truncation error comparable to the forecast error, whereas the mild form treats the linear dynamics exactly and approximates only the nonlinear integral (Section 4.3). For equations without a suitable linear part, such as the Richards equation, the discrete balance equations of a trusted numerical scheme take the role of $\mathcal{R}$.

Given a predicted window $h$, refinement minimizes $\|\mathcal{R}(h)\|^2$ over its stored field values, holding the incoming frame, physical parameters, and boundary data fixed; the network remains frozen. We use a few L-BFGS iterations, though any descent method would serve (Section 4.3 compares alternatives). For smooth solutions, a simpler update suffices: repeatedly replacing $h$ by the mild map $e^{tL}u_0 + \int_0^t e^{(t-s)L} N(h(s))\, ds$ of equation 1 gives Picard iteration, which requires no gradients. The refined window is then used as the next input (Figure 1a).

## 3.3 RESIDUAL WEIGHTING AND PROJECTED CORRECTIONS

A residual is most useful if it is sensitive to the forecast errors of interest relative to its own discretization error. Beyond the temporal sensitivity discussed above, it can fail in scale or mode when an error lies near a symmetry or the residual's null space. Spatial displacement is a clear example: a translated solution satisfies the equation within the window, so only its mismatch with the incoming

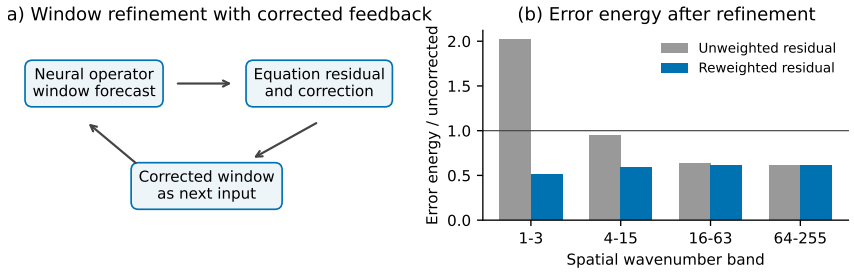


Figure 1: (a) Each predicted window is refined before being used as the next input. (b) Error energy after 100 refinement steps relative to the uncorrected forecast, by wavenumber band, for Burgers trajectories generated at a viscosity three times lower than in training. The unweighted residual amplifies the largest-scale error (depending on the step size), whereas the reweighted residual reduces error across all bands (Appendices C and H).

frame $u_0$ reveals the error. Large-scale displacement therefore produces a weak residual, and minimizing the unweighted residual leaves it largely uncorrected. In Figure 1b, the optimizer lowers the residual by correcting scales it sees, and distorting those it does not.

Two remedies restore sensitivity, one for each kind of failure. For a failure in scale, we increase the weight of large scales in the objective,

$$\mathcal{L}(h) = \sum_k W(k)|\widehat{\mathcal{R}(h)}(k)|^2, \qquad W(k) = 1 + (w_{\max} - 1)e^{-|k|/k_c}, \tag{2}$$

where $k$ is the spatial wavenumber, $w_{\max} = 64$ and $k_c = 8$. With this objective the large-scale error energy in the same experiment is halved. Reweighting helps on problems whose forecast error is dominated by displacement, such as shock-forming Burgers flow; when the error is broadband the unweighted residual already sees it, and reweighting then costs accuracy (Appendix H).

For a failure in mode, the dominant error occupies a small subspace to which the residual is weakly sensitive. Examples include a nearly uniform pressure offset in each Richards soil column and misplaced vortices of the right shape in Navier–Stokes. We then correct that subspace directly by a Newton step restricted to it, a coarse-space correction of the kind used in multigrid and reduced-order solvers (Section 2). Let $P$ be the matrix whose columns are the chosen directions, so that $Pa$ is a field in the subspace with amplitudes $a$, and let $S$ select the corresponding components of the residual. The step solves

$$(SJ_{\mathcal{R}}(h)P)a = -S\mathcal{R}(h), \qquad h^+ = h + \eta Pa, \tag{3}$$

where $J_{\mathcal{R}}$ is the Jacobian of the residual and $\eta$ a step length. For Burgers the subspace consists of the nine lowest Fourier modes; for the Richards equation, it consists of one pressure offset per soil column; for Navier–Stokes, the correction takes the form of transport along the predicted velocity (Section 4.2). These numerical constructions are standard; the contribution lies in selecting the subspace from the observed structure of the forecast error.

A discrete residual does not vanish on the exact solution. Because the integral in equation 1 is evaluated by quadrature over sparsely stored frames, even a true window has nonzero residual. We measure this consistency error on training trajectories and call the normalized departure the *floor*, $F$, a property of the residual rather than the network.

The floor indicates the error scale that refinement can achieve. Refinement moves a forecast toward windows that satisfy the discrete residual, which lie roughly one floor from the truth. A forecast whose error $E$ exceeds $F$ can therefore be improved, whereas one near or below $F$ is likely to be displaced. Section 4.4 tests this on 144 cases.

### 3.4 Backbones

We refine three backbones with the window interface: an FNO, Transolver++ adapted to that interface, and a physics-augmented spectral operator, hereafter the primitive NO. Each block of the

primitive NO applies three closed-form Fourier multipliers (a diffusion semigroup, a translation and an inverse Laplacian) alongside the learned spectral kernel of an FNO, with coefficients predicted for each channel from a pooled summary of the hidden state (equation 4, Appendix B). These branches act on all Fourier modes, while the learned kernel is truncated to low modes; their coefficients are internal network features, not refinement parameters.

# 4 EXPERIMENTS

**Systems.** We use one-dimensional PDEBench Burgers as a controlled testbed (1,024 cells; 9,000 training and 1,000 test trajectories; nine 20-frame windows), where an inexpensive reference solver enables detailed evaluation. We also test turbulent Navier–Stokes flow, the Richards equation and a one-dimensional advection–diffusion problem with heterogeneous coefficients. Appendix A gives detailed descriptions.

**Protocol.** We report the normalized root-mean-square error (nRMSE), the relative $L^2$ error of each predicted frame, averaged over frames and trajectories. Unless noted otherwise, errors are computed over complete rollouts with corrected feedback. L-BFGS-$k$, Adam-$k$ and Picard-$k$ denote $k$ iterations of the respective method per window, and ID denotes test data drawn from the training distribution. Timings cover the complete online rollout, including the network and the correction.

## 4.1 ROLLOUT ERROR REDUCTION

Table 1 applies the same weighted refinement (Sections 3.2 and 3.3, 50 L-BFGS iterations per window) to four Burgers backbones across 1,000 test trajectories. Refinement improves the raw forecasts, reducing the error by 40% for FNO, 72% for Transolver++, 62% for the primitive NO, and 50% for its padded variant. Uncorrected errors span a factor of 2.9, compared with 1.4 after refinement.

This improvement depends strongly on feeding each corrected window forward. Table 2 compares the same corrections in two modes: postprocessing, where each window is corrected without feeding the correction forward, and corrected feedback, where the corrected window is used as the next input. With L-BFGS-50, rollout error decreases from 0.0291 under postprocessing to 0.0113 with corrected feedback, because each correction also reduces the error propagated to subsequent windows.

## 4.2 CONTROLLING ERROR ACCUMULATION ALONG ROLLOUTS

A rollout-averaged error conceals how the error evolves, so Table 3 lists the first- and last-window errors. Without correction, most systems except Burgers in Table 3 show error growth: 2.7-fold over a 38-day Richards forecast, threefold for the advection–diffusion problem under the statistics shift (the coefficient-field statistics differ from training), and 20-fold in turbulent flow. Unlike the single-window error, this accumulation limits the forecast horizon. The effect is even stronger for Richards under the statistics shift (5.5-fold), as shown for additional regimes in Table 12.

With corrected feedback, this growth is largely suppressed on the non-chaotic systems, mainly by curbing inherited error (Appendix I). The Richards ID error remains near 0.003, while the advection–diffusion error under the statistics shift remains near 0.01 with L-BFGS-200. For Burgers, the FNO error decreases by 70% (from 0.0252 to 0.0066), with the same decreasing trend across the four backbones. In turbulent Navier–Stokes, the final corrected error is 56 times below the uncorrected rollout after nine windows (0.0060 versus 0.337).

**The Richards equation.** The Richards equation governs variably saturated flow, with pressure-dependent storage and conductivity and strongly heterogeneous conductivity fields (Appendix A). The NO forecasts the pressure head in a hillslope cross-section over 38 days, with the conductivity field supplied as input. We consider three regimes: ID; a *statistics shift*, in which the log-conductivity has a larger variance than in training; and a *correlation shift*, in which its correlation lengths differ. Because the equation has no suitable linear part and the domain is nonperiodic, we refine against the discrete mixed-form mass balance of a standard implicit scheme (Celia et al., 1990) in place of equation 1.

Table 1: Burgers results over 1,000 test trajectories and nine rollout windows. All backbones use the same weighted L-BFGS-50 refinement with corrected feedback. † denotes models and methods proposed in this work. Time is the complete refined rollout per trajectory; the uncorrected rollout takes 2–13 ms depending on the backbone, so refinement costs roughly 20 to 130 forward passes.

| Backbone | Uncorrected | Refined† | Time |
|---|---|---|---|
| | nRMSE ($\times 10^{-2}$) ↓ | | (ms) |
| FNO | **2.03** | 1.22 | 222 |
| Transolver++ | 5.86 | 1.66 | 231 |
| Primitive NO† | 3.18 | 1.21 | 234 |
| Primitive NO† (padded) | 2.38 | **1.19** | 236 |

Bold indicates the lowest error.

Table 2: Effect of corrected feedback on Burgers rollout error using the primitive NO and 128 test trajectories. Postprocessing corrects each window without feeding it forward; corrected feedback uses the corrected window as the next input.

| Correction | Postprocess | Feedback |
|---|---|---|
| | nRMSE ($\times 10^{-2}$) ↓ | |
| low-$k$ projected step | 3.27 | 2.65 |
| L-BFGS-10 | 3.01 | 2.24 |
| L-BFGS-50 | 2.91 | 1.13 |

Table 3: First- and last-window errors over each rollout (nine windows; 19 windows over 38 days for Richards). Backbones match the corresponding detailed tables: FNO for Burgers (Table 1), advection–diffusion (Table 10) and Navier–Stokes (Table 5; two-seed mean), and primitive NO for Richards (Table 4). Corrections are weighted L-BFGS-50 for Burgers, L-BFGS-200 for ADE (L-BFGS-50 does not suffice), line/column for Richards, and finer transport for Navier–Stokes. Results for all backbones are in Table 12.

| System (Backbone) | Uncorrected: first → last | Corrected (ours): first → last |
|---|---|---|
| | nRMSE ($\times 10^{-2}$) ↓ | |
| Burgers (FNO) | 2.67 → 2.00 | 2.52 → **0.66** |
| Variable-coefficient ADE, statistics shift (FNO) | 2.74 → 8.39 | 0.94 → **1.23** |
| Richards equation, ID (primitive NO) | 0.30 → 0.82 | 0.25 → **0.28** |
| Navier–Stokes, turbulent (FNO) | 1.69 → 33.7 | 0.006 → 0.60 |

Bold marks rollouts whose error settles at a bounded level.

The dominant error of the NO forecast is a pressure offset nearly uniform over the depth of each soil column. Such an offset contributes appreciably to the error and only weakly to the residual (failure in mode of Section 3.3). We correct it by applying a vertical-line solve to correct the pressure profile within each column, then use the projected step of equation 3 to adjust one pressure-offset amplitude per column and time level (Appendix G).

The line/column correction reduces the primitive NO error by 54%, 61% and 33% in the three regimes, and that of the FNO by 54%, 73% and 27% (Table 4), at 0.19–0.23 s per 38-day forecast, and it holds the error level along the forecast (Figure 2). The correction relies on the network forecast; initialized from a forecast that repeats the last frame, it leaves an error near 0.3.

**Turbulent Navier–Stokes.** We also consider a demanding periodic test, two-dimensional turbulence in the vorticity form of the Navier–Stokes equations at viscosity $10^{-5}$ on a $128^2$ grid. For this test, each backbone is trained with two seeds, with checkpoints selected on separate validation trajectories (Appendix E). Generic refinement is ineffective here, where Picard iteration was found to diverge, and minimizing the integral residual with 100 Adam iterations reduces the FNO error only from 0.050 to 0.039.

This again reflects a failure in the dominant error mode. The main error lies in where the flow transports the vorticity, to which the residual is nearly blind, and is regenerated in each window even when initialized from the true history. We therefore correct that mode directly, recomputing the vorticity by semi-Lagrangian advection of the incoming state along the velocity predicted by the NO (Appendix E).

Table 4: Richards results for 12 held-out trajectories per regime and 38-day feedback forecasts. Each backbone has 28.5M parameters and one trained seed. Errors use the fixed training pressure scale. Times include online prediction, correction, and transfers, with ranges spanning the three regimes. The primitive NO and line/column corrections are ours; FNO is the baseline backbone.

| Method | ID | Statistics shift | Correlation shift | Time (ms) |
|---|---|---|---|---|
| | nRMSE ($\times 10^{-3}$) ↓ | | | |
| FNO | 6.76 | 34.37 | 9.77 | 37–38 |
| FNO + line/column | 3.11 | 9.11 | 7.15 | 185–191 |
| Primitive NO | 5.81 | 18.91 | 7.43 | 78–80 |
| Primitive NO + column | 3.77 | 11.31 | 6.48 | 136–139 |
| **Primitive NO + line/column** | **2.65** | **7.44** | **5.00** | 227–233 |

Bold indicates the lowest error.

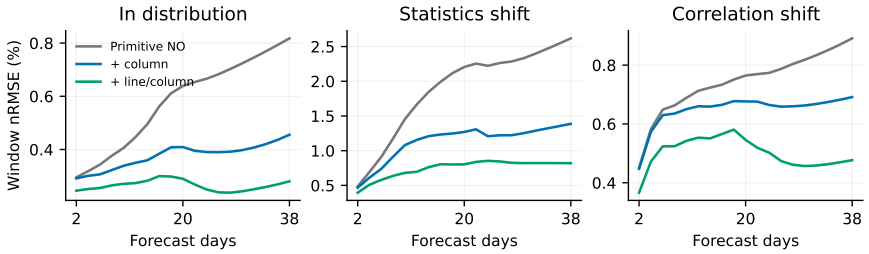


Figure 2: nRMSE results for Richards equation with primitive backbone on 12 held-out trajectories per regime over 38 forecast days.

Three passes reduce the FNO error by 83% (from 0.0497 to 0.0083) in 17 ms per trajectory, equivalent to about eleven forward passes (Table 5). A local frame-wise correction reaches only 0.032 at similar cost. The time-varying NO velocity is important, where replacing it with the last observed velocity held constant increases error to 0.0173. A higher-accuracy variant using higher-order interpolation and a retained ($256^2$) state attains 0.0026 at 54 ms. On a nine-window rollout, three times longer than any chain used in training, it attains 0.0021 against 0.17 for the uncorrected forecast (Figure 5, Appendix E).

Both transport corrections go below the residual's own floor (0.0089), far below what residual descent reaches, since they reconstruct the field from the transport law rather than minimize the residual. On a cohort generated after all correction settings were frozen, three transport passes reduce the error by 85.9%, with the ordering of the arms unchanged across backbones and training seeds (Table 14, Appendix J).

### 4.3 The residual form

We next change only the residual, replacing the integral residual with the pointwise residual $\partial_t u - Lu - N(u)$ minimized by physics-informed neural operators (Li et al., 2024), with fourth-order finite differences in time. We test this on five systems with both backbones, matched L-BFGS-50 and identical weighting (Table 9, Appendix D). The integral residual gives the lower error in all cases, reducing it by 11–97%. The pointwise residual makes the forecast worse in half of the cases and improves it by at most 9% (except reaction–diffusion with the primitive NO, up to 40%). A likely cause is the finite-difference time derivative, which is inaccurate across widely spaced frames.

A one-step scheme residual fails for a related reason. PhysicsCorrect (Huang & Perdikaris, 2025), the closest training-free method, corrects each predicted frame against one semi-implicit scheme step with a cached Jacobian inverse. With the authors' code and weights, we reproduce its benefit when the network advances one solver step at a time (Kuramoto–Sivashinsky, rollout error 0.45 to 0.11), but there the corrected state is simply the scheme's solution, so the network serves only as an initial guess. On our Burgers windows, with frames about ten CFL units apart, it diverges within six frames: the cached iteration no longer contracts, and one scheme step has a truncation error as large

Table 5: Turbulent Navier–Stokes results on 24 test trajectories over three windows. Each backbone is trained with two seeds; entries report the seed mean, with seed ranges where shown. The two backbones are size-matched (6.6M parameters). † denotes methods proposed in this work. FNO and Adam residual descent are baselines. Time is the complete rollout per trajectory.

| | FNO | | Primitive NO† | |
|---|---|---|---|---|
| Correction | nRMSE ($\times 10^{-2}$) ↓ | Time (ms) | nRMSE ($\times 10^{-2}$) ↓ | Time (ms) |
| None | 4.97 [4.97, 4.98] | 1.5 | **4.58** [4.57, 4.59] | 3.6 |
| Adam-100 on the residual | 3.94 [3.93, 3.94] | 271 | **3.58** [3.57, 3.59] | 284 |
| local correction† | 3.22 [3.20, 3.24] | 18.9 | **3.02** [3.00, 3.04] | 21.0 |
| transport†, 1 pass | 1.29 [1.23, 1.35] | 6.8 | **1.18** [1.14, 1.22] | 9.2 |
| transport†, 3 passes | 0.833 | 16.9 | 0.831 | 19.7 |
| transport†, finer version | 0.256 | 54.3 | 0.256 | 58.4 |

Bold indicates the lowest error.

Table 6: Refinement across systems. The floor is the residual's error on a reference window (Section 3.3); dashes mark unmeasured floors. Gain is uncorrected over refined error. FNO backbones, except the primitive NO on Richards; Adam-100 is the generic baseline; all other corrections are ours. Time in ms per trajectory, for the correction (C) or the full rollout (O).

| System | Correction | Floor | Uncorrected | Refined | Gain | Time (ms) |
|---|---|---|---|---|---|---|
| | | | nRMSE ↓ | | (×) | |
| 1D Burgers | L-BFGS-50 | .016 | .0203 | .0122 | 1.7 | 222 (O) |
| 1D reaction–diffusion | Picard-5 | .00001 | .000071 | .000022 | 3.2 | 17 (C) |
| 1D ADR | L-BFGS-50 | .00021 | .00012 | .00023 | 0.5 | 81 (C) |
| 2D Burgers | L-BFGS-50 | – | .0069 | .00066 | 10.5 | 305 (C) |
| 2D Navier–Stokes, smooth | Picard-3 | – | .0019 | .00063 | 3.0 | 5.8 (C) |
| 2D Navier–Stokes, turbulent | Adam-100 | .0089 | .0497 | .0394 | 1.3 | 271 (O) |
| ↪ same trajectories | transport, 3 passes | .0089 | .0497 | .0083 | 6.0 | 16.9 (O) |
| Richards equation, ID | line/column | – | .0058 | .0027 | 2.1 | 230 (O) |

as the network's error. In the authors' own system the gain likewise erodes with correction spacing and reverses at 32 solver steps (Appendix D.1, Table 8). The integral residual propagates the linear dynamics exactly across the gap and thus remains valid at window-forecast frame spacing.

We also evaluated two learned refiners (Appendix D, Table 7). PDE-Refiner, a learned denoiser that uses no equation, reduces error by only 2%. Metamizer, a learned optimizer, diverges with its published pointwise losses. Fine-tuned on our integral residual, it reaches 0.0130 after 20 steps against 0.0113 for L-BFGS-50. Its runtime is about eight times that of L-BFGS at batch 64, however its cost is lower at batch 8.

### 4.4 Testing the refinement diagnostic

Refinement tends to improve a forecast when its error exceeds the floor and to degrade it otherwise. The floor needs one measurement per system and is independent of the forecast. We tested this on 144 refinement runs spanning four one-dimensional systems (Burgers, advection, reaction–diffusion and advection–diffusion–reaction), four quadrature orders, synthetic degradations of true windows and real forecasts from three networks (Appendix H). The outcome follows the ratio of floor to forecast error. When $F/E$ is below 0.5, 73 runs improve, and when it exceeds 1.5, 91% degrade. The mean improvement crosses zero between 0.8 and 1.2.

Table 6 applies this diagnostic across systems. On reaction–diffusion, where the forecast error is seven times the floor, refinement reduces it by a factor of 3. For one-dimensional Burgers, with $F/E \approx 0.8$ just below the crossover, the gain is a modest 1.7. Two-dimensional Burgers and smooth Navier–Stokes, whose floors we did not measure, improve by factors of 10 and 3, respectively. We observed that a few Picard iterations suffice for the smooth systems. This diagnostic test then predicts whether refinement can help, but it does not guarantee that an optimizer realizes

the improvement. On turbulent Navier–Stokes the forecast error is 5.6 times the floor, yet residual descent (Adam-100) removes only 21% of it, while transport correction on the same trajectories reaches the floor (0.0083 against 0.0089) in one-sixteenth of the time (Section 4.2).

### 4.5 Targeted corrections and their cost

Full refinement is expensive relative to forward forecasts. Refinement multiplies rollout cost by roughly 20–130 on the Burgers system (batch 8 in Table 1), and by about 4 at batch 64 (Table 7). For Richards equation, it increases by 3 to 5 (Table 4). The corrections targeted at the dominant error of each system (Section 3.3) cost far less. A single half step of equation 3 applied to the nine lowest Fourier modes lowers the error for Burgers by 20% (0.0331 to 0.0265) adding only 0.23 ms to a 9.6 ms rollout (Figure 3, Appendix C). On trajectories generated at a viscosity three times lower than in training, the same step lowers the error from 0.192 to 0.162 (Appendix B). Three transport passes in Navier–Stokes take 17 ms, about eleven times the uncorrected rollout, whereas 100 Adam iterations take 271 ms and achieve far less. For Richards equation the column correction brings the rollout to 0.14 s, under twice the uncorrected 0.08 s. Control experiments on Burgers rule out generic smoothing as the source of the gain and show that a smaller residual does not necessarily mean a smaller error (Appendices C and H). We also compared runtimes with classical solvers (Appendix G, Table 11). For the Richards equation in the ID regime, a converged classical solver on the same GPU needs 0.67 s to attain 0.0053 and 1.3 s to attain 0.0022, whereas the corrected network attains 0.0027 in 0.23 s. However, under the statistics shift, the solver at 0.67 s is more accurate (0.0044 against 0.0074).

### 4.6 Sensitivity to backbones

Backbones benefit from refinement substantially, especially the least accurate ones (Section 4.1). For the Richards equation, both backbones improve by 27–73% in all regimes (Table 4), on turbulent Navier–Stokes both by more than 80% (Table 5), and on the variable-coefficient problem the three backbones improve (Appendix F). Under strong refinement, the networks converge where the spread of Burgers errors contracts from a factor of 2.9 to 1.4 (Table 1), that of the variable-coefficient problem from 30 to 1.8 (ID, L-BFGS-200; Table 10), and under three transport passes the two Navier–Stokes backbones agree within 0.3% (Table 5). At small correction budgets the better forecast stays ahead. On both two-dimensional problems the primitive NO is the more accurate uncorrected forecast at matched parameter count, by 8% on Navier–Stokes, and by 14% and 45% for Richards equation under ID and statistics shift respectively (Tables 5 and 4, Appendix F); an FNO with up to 5.7 times as many parameters does not close the gap. The advantage persists under the local correction and one transport pass on Navier–Stokes, and under the full Richards correction (0.0027 against 0.0031 ID). On the smooth one-dimensional testbeds the FNO is the more accurate uncorrected forecast (Appendix F).

## 5 Discussion and limitations

A neural operator rolled out on its own accumulates error. Refining each predicted window against the equation and feeding the refined window forward lowers rollout error substantially. Where the dynamics are not chaotic, it also stops the error compounding by removing what each window inherits from the last. The cost ranges from a fraction of a millisecond to a quarter of a second per trajectory. Two conditions govern whether this works, both concerning the residual rather than the network: the equation must be imposed in integral form, which we attribute mainly to the inaccuracy of finite-difference time derivatives across stored frames, and the correction must act on the error the forecast actually contains, which is why the residual's own error predicts the benefit and why the dominant error mode is sometimes better corrected directly. Under strong correction the network matters little, suggesting that effort spent comparing architectures may be better spent on the correction. The study has some limitations. It requires the governing equation, and its inexpensive corrections require identifying each system's dominant error mode from reference trajectories. The screen predicts whether refinement can help, not whether an optimizer realizes the gain, and refinement degrades forecasts already more accurate than the residual can certify. Classical solvers remain competitive at some accuracy targets. Finally, refinement results use one seed per backbone, learned refiners were compared on Burgers only, and Transolver++ was not tuned for this task.

## REPRODUCIBILITY STATEMENT

We provide descriptions of the systems, including equations, coefficients, grids and regime shifts in Appendix A. Appendix B presents different architectures used, training schedules, and evaluation metrics. Appendices C and D provide additional information on the residual forms and correction algorithms. Code to reproduce experiments will be made publicly available upon acceptance.

## AI USE DISCLOSURE

We used large language models (LLMs) to (i) edit the manuscript for readability and concision; (ii) draft and condense parts of the main text, abstract, tables, captions, and appendices; (iii) identify and summarize related work; (iv) assist with experiment planning, debugging, and analysis; and (v) generate and implement some of the experimental code. All AI-assisted text, code and results were reviewed and verified by the authors. All cited works were also checked by the authors. The authors take full responsibility for the content of this paper.

## REFERENCES

Lianghao Cao, Thomas O'Leary-Roseberry, Prashant K. Jha, J. Tinsley Oden, and Omar Ghattas. Residual-based error correction for neural operator accelerated infinite-dimensional bayesian inverse problems. *Journal of Computational Physics*, 486:112104, 2023. doi: 10.1016/j.jcp.2023.112104. Preprint arXiv:2210.03008, 2022.

Michael A Celia, Efthimios T Bouloutas, and Rebecca L Zarba. A general mass-conservative numerical solution for the unsaturated flow equation. *Water Resources Research*, 26(7):1483–1496, 1990.

Steven M Cox and Paul C Matthews. Exponential time differencing for stiff systems. *Journal of Computational Physics*, 176(2):430–455, 2002.

Alok Dutt, Leslie Greengard, and Vladimir Rokhlin. Spectral deferred correction methods for ordinary differential equations. *BIT Numerical Mathematics*, 40(2):241–266, 2000.

Mohammad Sadegh Eshaghi, Cosmin Anitescu, Navid Valizadeh, Yizheng Wang, Xiaoying Zhuang, and Timon Rabczuk. NOWS: Neural operator warm starts for accelerating iterative solvers. *Computer Methods in Applied Mechanics and Engineering*, 458, 2026.

Zexi Fan, Yan Sun, Shihao Yang, and Yiping Lu. Physics-informed inference time scaling via simulation-calibrated scientific machine learning. *arXiv preprint arXiv:2504.16172*, 2025.

Takashi Furuya, Koichi Taniguchi, and Satoshi Okuda. Quantitative approximation for neural operators in nonlinear parabolic equations. *arXiv preprint arXiv:2410.02151*, 2024.

Wenrui Hao, Xinliang Liu, and Yahong Yang. Newton-informed neural operator for computing multiple solutions of nonlinear PDEs. In *Advances in Neural Information Processing Systems*, 2024. arXiv:2405.14096.

Marlis Hochbruck and Alexander Ostermann. Exponential integrators. *Acta Numerica*, 19:209–286, 2010.

Xinquan Huang and Paris Perdikaris. PhysicsCorrect: A training-free approach for stable neural PDE simulations. *arXiv preprint arXiv:2507.02227*, 2025.

Prashant K. Jha. Residual-based error corrector operator to enhance accuracy and reliability of neural operator surrogates of nonlinear variational boundary-value problems. *arXiv preprint arXiv:2306.12047*, 2023.

Aly-Khan Kassam and Lloyd N Trefethen. Fourth-order time-stepping for stiff PDEs. *SIAM Journal on Scientific Computing*, 26(4):1214–1233, 2005.

Dmitrii Kochkov, Jamie A Smith, Ayya Alieva, Qing Wang, Michael P Brenner, and Stephan Hoyer. Machine learning–accelerated computational fluid dynamics. *Proceedings of the National Academy of Sciences*, 118(21), 2021.

Nikola Kovachki, Zongyi Li, Burigede Liu, Kamyar Azizzadenesheli, Kaushik Bhattacharya, Andrew Stuart, and Anima Anandkumar. Neural operator: Learning maps between function spaces with applications to PDEs. *Journal of Machine Learning Research*, 24(89):1–97, 2023.

Aditi Krishnapriyan, Amir Gholami, Shandian Zhe, Robert Kirby, and Michael W Mahoney. Characterizing possible failure modes in physics-informed neural networks. In *Advances in Neural Information Processing Systems*, 2021.

Zongyi Li, Nikola Kovachki, Kamyar Azizzadenesheli, Burigede Liu, Kaushik Bhattacharya, Andrew Stuart, and Anima Anandkumar. Fourier neural operator for parametric partial differential equations. In *International Conference on Learning Representations*, 2021.

Zongyi Li, Hongkai Zheng, Nikola Kovachki, David Jin, Haoxuan Chen, Burigede Liu, Kamyar Azizzadenesheli, and Anima Anandkumar. Physics-informed neural operator for learning partial differential equations. *ACM/IMS Journal of Data Science*, 1(3):1–27, 2024.

Weixin Liao, Mingquan Feng, Zhizhou Zhang, Youjia Wu, Yifan Fu, and Junchi Yan. G-RANS: Generalizable residual-aware neural solvers for sparse systems. In *International Conference on Machine Learning*, 2026.

Payton Lindsay, Jeffrey Fike, Irina Tezaur, and Kevin Carlberg. Preconditioned least-squares petrov-galerkin reduced order models. *arXiv preprint arXiv:2203.12180*, 2022.

Phillip Lippe, Bastiaan S Veeling, Paris Perdikaris, Richard E Turner, and Johannes Brandstetter. PDE-Refiner: Achieving accurate long rollouts with neural PDE solvers. In *Advances in Neural Information Processing Systems*, 2023.

Lu Lu, Pengzhan Jin, Guofei Pang, Zhongqiang Zhang, and George Em Karniadakis. Learning nonlinear operators via DeepONet based on the universal approximation theorem of operators. *Nature Machine Intelligence*, 3(3):218–229, 2021.

Huakun Luo, Haixu Wu, Hang Zhou, Lanxiang Xing, Yichen Di, Jianmin Wang, and Mingsheng Long. Transolver++: An accurate neural solver for pdes on million-scale geometries. *arXiv preprint arXiv:2502.02414*, 2025.

Tanya Marwah, Ashwini Pokle, J Zico Kolter, Zachary C Lipton, Jianfeng Lu, and Andrej Risteski. Deep equilibrium based neural operators for steady-state PDEs. In *Advances in Neural Information Processing Systems*, 2023.

Christopher Maynard, Thomas Melvin, and Eike Müller. Multigrid preconditioners for the mixed finite element dynamical core of the lfric atmospheric model. *Quarterly Journal of the Royal Meteorological Society*, 2020. doi: 10.1002/qj.3880.

Nick McGreivy and Ammar Hakim. Weak baselines and reporting biases lead to overoptimism in machine learning for fluid-related partial differential equations. *Nature Machine Intelligence*, 6: 1256–1269, 2024.

Hugo Melchers, Michael Abdelmalik, and Victorita Dolean. When can a neural operator replace a coarse solve? architectural principles for two-level preconditioning. *arXiv preprint arXiv:2605.19867*, 2026.

Jaemin Oh, Youngkyu Lee, Jerome Darbon, and George Em Karniadakis. Spectrally safe neural operator warm-starts for large-scale Newton solvers. *arXiv preprint arXiv:2606.21828*, 2026.

Tobias Pfaff, Meire Fortunato, Alvaro Sanchez-Gonzalez, and Peter W Battaglia. Learning mesh-based simulation with graph networks. In *International Conference on Learning Representations*, 2021.

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.

Rajyasri Roy, Dibyajyoti Nayak, and Somdatta Goswami. ANCHOR: Error-controlled adaptive numerical correction for neural operator time marching. *arXiv preprint arXiv:2512.19643*, 2025.

Jos Stam. Stable fluids. In *Proceedings of SIGGRAPH*, pp. 121–128, 1999. doi: 10.1145/311535.311548.

Kiwon Um, Robert Brand, Yun Raymond Fei, Philipp Holl, and Nils Thuerey. Solver-in-the-loop: Learning from differentiable physics to interact with iterative PDE-solvers. In *Advances in Neural Information Processing Systems*, 2020.

Nils Wandel, Stefan Schulz, and Reinhard Klein. Metamizer: A versatile neural optimizer for fast and accurate physics simulations. In *International Conference on Learning Representations*, 2025. arXiv:2410.19746.

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.

Yizheng Wang, Zhongkai Hao, Mohammad Sadegh Eshaghi, Cosmin Anitescu, Xiaoying Zhuang, Timon Rabczuk, and Yinghua Liu. Pretrain finite element method: A pretraining and warm-start framework for PDEs via physics-informed neural operators. *arXiv preprint arXiv:2601.03086*, 2026.

Haixu Wu, Huakun Luo, Haowen Wang, Jianmin Wang, and Mingsheng Long. Transolver: A fast transformer solver for pdes on general geometries. In *International Conference on Machine Learning*, 2024. arXiv:2402.02366.

Huaiqian You, Quinn Zhang, Colton J Ross, Chung-Hao Lee, and Yue Yu. Learning deep implicit Fourier neural operators (IFNOs) with applications to heterogeneous material modeling. *Computer Methods in Applied Mechanics and Engineering*, 398:115296, 2022.

Enrui Zhang, Adar Kahana, Alena Kopaničáková, Eli Turkel, Rishikesh Ranade, Jay Pathak, and George Em Karniadakis. Blending neural operators and relaxation methods in PDE numerical solvers. *Nature Machine Intelligence*, 6(11):1303–1313, 2024. doi: 10.1038/s42256-024-00910-x.

# A SYSTEMS

All spectral systems are posed on periodic domains and split as $\partial_t u = Lu + N(u)$, with the linear symbol $\lambda(k)$ used by the mild kernel of equation 1. The Richards equation has no suitable linear part and is refined against a discrete scheme instead.

**1D Burgers (primary testbed).** $\partial_t u + u\,\partial_x u = \nu_{\text{eff}}\,\partial_{xx} u$ on $[0, 1)$, with $\lambda(k) = -\nu_{\text{eff}} k^2$, $N(u) = -\frac{1}{2}\partial_x(u^2)$ and $\nu_{\text{eff}} = \nu/\pi$ (the PDEBench generator convention; Appendix B).

**1D linear advection.** $\partial_t u + \beta\,\partial_x u = 0$ with $\beta = 1$; $\lambda(k) = -i\beta k$ and $N \equiv 0$. The equation is linear, so the mild map is the exact solution operator.

**1D reaction–diffusion.** $\partial_t u = \nu\,\partial_{xx} u + \rho\,u(1-u)$ with $\nu = 10^{-3}$ and $\rho = 5$. The linear part of the reaction is absorbed into the kernel: $\lambda(k) = -\nu k^2 + \rho$ and $N(u) = -\rho u^2$.

**1D advection–diffusion–reaction (ADR).** $\partial_t u + \beta\,\partial_x u = \nu\,\partial_{xx} u + \rho\,u(1-u)$ with constant coefficients $\lambda(k) = -i\beta k - \nu k^2 + \rho$ and $N(u) = -\rho u^2$.

**1D variable-coefficient advection–diffusion (ADE).** $\partial_t u + v(x)\,\partial_x u = \partial_x\big(D(x)\,\partial_x u\big)$ on a 1,024-cell grid, with per-trajectory log-Gaussian fields $v(x)$ and $D(x)$ (log-standard deviations 0.3 and 0.5, correlation length 0.1), supplied to the network as input channels. The spatial means $\bar{v}$ and $\bar{D}$ enter the symbol, $\lambda(k) = -i\bar{v}k - \bar{D}k^2$, and the heterogeneous remainder enters the forcing, $N(u) = -(v-\bar{v})\,\partial_x u + \partial_x\big((D-\bar{D})\,\partial_x u\big)$. The statistics shift multiplies the log-standard deviations by 1.5; the correlation shift halves the correlation length (Appendix F).

**2D Burgers (scalar).** $\partial_t u + u\,(\partial_x u + \partial_y u) = \nu\,\Delta u$ with $\nu = 0.02$ on a $128^2$ grid; $\lambda(\mathbf{k}) = -\nu|\mathbf{k}|^2$ and $N(u) = -u\,(\partial_x u + \partial_y u)$.

**2D incompressible Navier–Stokes (vorticity form).** $\partial_t \omega + \mathbf{u}\cdot\nabla\omega = \nu\,\Delta\omega$ with $\mathbf{u} = \nabla^{\perp}\psi$ and $\Delta\psi = -\omega$ on a $128^2$ grid; $\lambda(\mathbf{k}) = -\nu|\mathbf{k}|^2$ and $N(\omega) = -\mathbf{u}\cdot\nabla\omega$. The per-mode Poisson inversion keeps $N$ pointwise computable, and the unknown remains scalar. The smooth case uses $\nu = 10^{-3}$; the turbulent case uses $\nu = 10^{-5}$, with reference trajectories integrated at $512^2$ and restricted to $128^2$ (Appendix E).

**Richards equation (mixed form).** $\partial_t \theta(\psi) = \nabla\cdot\big[K(\psi)\,\nabla(\psi + z)\big]$ on a sloped $210\,\text{m} \times 8\,\text{m}$ domain, with van Genuchten retention $\theta(\psi)$ and Mualem conductivity $K(\psi) = K_s\sqrt{S_e}\,\big[1 - (1 - S_e^{1/m})^m\big]^2$, $S_e = \big[1 + (\alpha|\psi|)^n\big]^{-m}$, where $\ln K_s$ is a Gaussian random field drawn per sample. The boundaries are no-flow on the left and bottom, hydrostatic Dirichlet on the right, and a time-dependent recharge flux on top. The statistics shift raises the standard deviation of $\ln K_s$ from 0.8 to 1.2; the correlation shift changes the horizontal and vertical correlation lengths from 30 and 0.6 m to 60 and 0.3 m at unchanged marginal variance. The residual is the mixed-form discretization of Celia et al. (1990), not a mild form (Appendix G).

# B ARCHITECTURE, TRAINING AND EVALUATION

The primitive NO lifts a window to 32 hidden channels and applies four space–time spectral blocks. Each block pools the hidden state over space and time and predicts channel coefficients: $\nu_c = 10^{-2}\,\text{softplus}(\cdot)$, $c_c = 3\tanh(\cdot)$ and $\alpha_c = 0.3\tanh(\cdot)$. In each block the closed-form branches and the learned kernel act on channel $c$ of the hidden state $h$ as

$$\begin{aligned} \widehat{D}_c &= e^{-\nu_c|k|^2 T_w}\widehat{h}_c, \quad & \widehat{S}_c &= e^{-ik\cdot c_c T_w}\widehat{h}_c, \\ \widehat{P}_c &= \alpha_c|k|^{-2}\widehat{h}_c, & \widehat{H} &= W_\theta(k)\widehat{h}. \end{aligned} \tag{4}$$

where $T_w$ is the window length. The physical branches of equation 4 are combined with a learned $48 \times 6$ Fourier kernel, a channel mixer and a residual connection. The inverse-Laplacian zero mode is zero. The output is anchored at the last input frame. Parameters in these feature operators are learned; they are not the physical coefficients used by the refinement residual.

The main Burgers model is trained for 400,000 Adam updates, batch 8, with learning rate $5 \times 10^{-4}$, warmup and cosine decay. The rollout length is sampled from $\{1, 2, 3, 5\}$ windows, with 1% input noise in training-standard-deviation units. Its full-test score is 0.0318. The corresponding 128-case correction cohort uses ICs 9032–9159 and has raw score $3.3085 \times 10^{-2}$; its weighted L-BFGS-50 score is $1.1322 \times 10^{-2}$. The full 1,000-case refined evaluation gives $1.15 \times 10^{-2}$. We report these cohorts separately rather than replacing one full-test number with a favorable subset. Table 1 instead scores every backbone in one harness on all 1,000 ICs at batch 8, where the same refinement gives 0.0121; batched L-BFGS line search makes the refined value mildly batch dependent.

The time-padded variant appends ten zero frames to the time axis before each block's joint space–time FFT, so the last frame of a window is not treated as periodic with its first. It is otherwise identical (same parameters, training schedule and data) and lowers raw error from 0.0318 to 0.0238.

**Matched periodic FNO baseline.** The FNO window model uses four spatial Fourier layers, width 64 and 32 modes, pointwise channel mixing and a two-layer output projection (553,236 parameters under PyTorch's complex parameter counting convention). Time samples are input/output channels; periodic convolutions use no spatial padding. It shares the 20-to-20 interface, training normalization, last-frame residual anchor, 9,000 training ICs, 1,000 test ICs, random rollout lengths and 400,000-update schedule with the other Burgers window models. No test checkpoint selection is used. All rows forecast frames 20–199 on ICs 9000–9999. The comparison controls the data and rollout protocol, but does not isolate an individual physical branch or match parameter counts.

**Transolver-family implementation.** The official Transolver++ implementation is adapted to one spatial coordinate and 20 input/output channels, with width 256, five layers, eight heads and 32 slices. A last-frame residual anchor and data normalization use training statistics. It receives 400,000 updates, batch 8, random rollout lengths $\{1, 2, 3, 5\}$, 1% input noise, and the same Adam warmup/cosine schedule as the primitive NO. No test checkpoint selection is used. Official Gumbel slicing remains stochastic in evaluation; three scores are $(5.87,\ 5.88,\ 5.89) \times 10^{-2}$ (mean $5.88 \times 10^{-2}$, standard deviation $1.2 \times 10^{-4}$). This is inference variability of one trained model. The 1.50M-parameter model trains for 8.68 hours on V100S.

Original Transolver under the same adaptation and training schedule has persistently poor fit (1.7979 test nRMSE). We exclude that failed configuration; it does not establish the architecture's achievable accuracy. Transolver++ is trained with the shared window recipe rather than the official PDEBench recipe (AdamW with weight decay, one-cycle schedule, gradient clipping, 64 slices); architecture-specific tuning and multiple trained seeds remain open, and its raw accuracy here should not be read as the architecture's best. Under the matched recipe it is the least accurate raw backbone on Burgers (0.0586 versus 0.0203 for the FNO) and reaches 0.0166 after the shared refinement.

**Metrics and shifts.** For Burgers and NS, we average framewise spatial RMSE divided by the frame's reference RMS, then average trajectories. Richards uses pressure RMSE normalized by the fixed training scale 1.9788 m. PDEBench Burgers uses effective diffusivity $\nu/\pi$; using the filename's nominal $\nu$ in the residual changes the equation. The viscosity-shift experiment uses independent Cole–Hopf data at effective diffusivity $10^{-4}$ with the training initial-condition amplitude distribution. The half projected step was fixed before its 128-case confirmation: raw 0.1918 becomes 0.1615, a 15.8% reduction (paired bootstrap 95% interval 13.1–18.3%). The stable Cole–Hopf evaluation uses log-sum-exp quadrature with periodic images.

**Training budgets across the breadth cases.** Advection uses 9,000 training ICs and 400,000 updates; reaction–diffusion uses 1,500 and 250,000; ADR uses 1,500 and 200,000. All three use the primitive window architecture with random rollout lengths and noisy inputs. Smooth NS and 2D Burgers each use 1,000 training ICs and 25,000 single-window Adam updates, width 32 and 20 modes. These schedules and their small raw errors distinguish them from the original 48-trajectory difficult-NS pilot; they do not establish globally optimal training. The matched one-dimensional FNOs use the same data, rollout lengths, noise and update counts as the primitive model for each system. The Richards FNO and primitive models each use 800 trajectories, 100,000 single-window updates and 30,000 four-window updates, with no skipped training steps. The primitive model's mean training loss changes by about 3% over the final 5,000 fine-tuning updates. Architecture com-

parisons still confound the complete inductive biases and training interfaces; no individual physical branch is credited with the entire improvement.

## C RESIDUAL AND CORRECTION IMPLEMENTATION

For a constant-coefficient spectral linear part with symbol $\lambda_k$, the nonlinear Duhamel integral is evaluated by the trapezoidal recurrence

$$I_j = E_k I_{j-1} + \frac{\Delta t}{2}\big(E_k \widehat{N}_{j-1} + \widehat{N}_j\big), \qquad E_k = e^{\lambda_k \Delta t}.$$

The incoming endpoint uses $N(u_0)$ exactly. The recurrence can be evaluated as a causal convolution using an FFT in time. Quadratic products use spatial dealiasing. Burgers uses endpoint quadrature at the saved times ($q = 1$); reaction–diffusion uses fourfold internal quadrature. These choices are assessed offline against reference trajectories. A small residual on reference data is a consistency diagnostic, not a rigorous forecast-error bound.

Weighted Burgers refinement uses strong-Wolfe L-BFGS on field values with $W(k)$ from equation 2; network parameters are frozen. Each window has a fixed iteration budget and restores the incoming mean. The main ID setting is $(w_{\max}, k_c) = (64, 8)$; the earlier OOD weighting experiment in Figure 1 uses $(16, 8)$. The figure compares the same first-window, 100-step refinement budgets, rather than the newer projected correction. Its error-energy ratios are:

| Spatial band | 1–3 | 4–15 | 16–63 | 64–255 |
|---|---|---|---|---|
| Unweighted OOD | 2.02 | 0.95 | 0.63 | 0.61 |
| Reweighted OOD | 0.51 | 0.59 | 0.61 | 0.61 |

Physics-free low-pass filtering fails to reproduce the benefit: ID rollout error is 0.152 at cutoff 8 and 0.057 at cutoff 32. Larger spectral weights do not monotonically help; weight 256 is worse than 64 in the weighting ablation. These controls support a frequency-specific correction rather than generic smoothing.

**Projected Burgers correction.** For $q = 1$, the Fourier residual has recurrence

$$r_j = \widehat{h}_j - E\widehat{h}_{j-1} - \frac{\Delta t}{2}\big(\widehat{N}(h_j) + E\widehat{N}(h_{j-1})\big).$$

Restricting to modes $-4, \dots, 4$ gives twenty $9 \times 9$ complex linear systems per window. The projected Jacobian includes the derivative of the full-field nonlinear flux and the causal previous-frame coupling. A half step is used in the reported inexpensive correction. The projected Jacobian is checked against an independent automatic-differentiation action. The residual recurrence agrees with the existing mild evaluator within $10^{-6}$ relative tolerance. References are never used for online amplitudes. Increasing a correction's dimension or iteration count need not improve reference accuracy when the evaluator is biased.

**Scope of the shared formulation.** For smooth cases, Picard iteration applies the mild map directly. On a linear equation a full application can recover the discrete solution independently of the neural candidate; such a row illustrates correction, not a neural speed advantage. The harder nonlinear experiments instead test whether a limited correction retains useful information from the neural prediction.

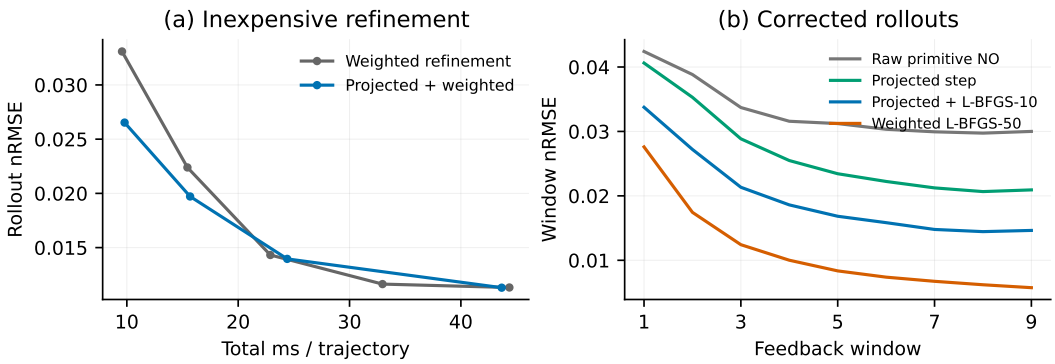


Figure 3: Burgers, 128 test trajectories, primitive backbone. (a) Accuracy against total online cost. The projected low-wavenumber step improves the low-cost end of the trade-off. (b) Error along the rollout for the uncorrected forecast and three corrections with corrected feedback.

## D LEARNED-REFINER ADAPTATIONS AND RESIDUAL FORMS

Metamizer is adapted to the space–time Burgers field, padding the 20-frame time dimension to 32. With its released pretrained weights and its own pointwise physics loss, it diverges from the first step. Its optimizer is therefore fine-tuned on our weighted mild residual, starting from frozen primitive-NO forecasts, with ten-step unrolling, 3,000 training updates, batch 4 and learning rate $10^{-4}$. Evaluation on 128 ICs gives $1.304 \times 10^{-2}$ after 20 steps and $1.298 \times 10^{-2}$ after 50 (batch 64).

The PDE-Refiner adaptation uses a noise-conditioned spectral network with the same four-block, 32-channel window backbone. With the primitive NO frozen as predictor, the refiner learns noise removal at three exponentially spaced noise levels. Training uses the 9,000 training ICs for 30,000 Adam updates, batch eight, with cosine decay from learning rate $10^{-3}$. Inference evaluates noise scales $10^{-3}$ and $10^{-4}$, partial schedules, and deterministic/stochastic variants. The best tested partial schedule gives 0.0324. The best full schedule gives 0.0767 using the $10^{-3}$ checkpoint without noise injection. These scores describe the tested settings; the schedule sweep is not an independent final validation of a universally optimal setting. Joint predictor/refiner training was also attempted, but its poor predictor fit makes it a less informative comparison than the frozen-predictor adaptation.

**Matched timing and cohort conventions.** Table 7 reruns the existing checkpoints on ICs 9000–9127 at both batch eight and batch 64 on V100S. Each arm has one complete batch-rollout warmup and three timed full-cohort repeats. Timers include prediction, all refinement work and output assembly from GPU-resident inputs; loading, training and reference scoring are excluded. PDE-Refiner uses the existing $10^{-4}$ noise-scale checkpoint, one finest-level denoising step and no noise injection. No weights or schedules are selected by these timing runs. The 128-case frontier and feedback cohort instead uses ICs 9032–9159; these slightly different subsets explain $1.134 \times 10^{-2}$ versus $1.132 \times 10^{-2}$ for L-BFGS-50 at batch 64.

### D.1 ONE-STEP SCHEME CORRECTION (PHYSICSCORRECT)

PhysicsCorrect (Huang & Perdikaris, 2025) writes the residual as one step of a time-stepping scheme between consecutive states, caches the inverse of the Jacobian of its linear part, and applies one correction per predicted state. We ran the authors' released code, network weights and Kuramoto–Sivashinsky (KS) test set (64 trajectories, 999 steps, $\Delta t = 0.05$, equal to the step of the generating solver). The mean relative error falls from 0.453 to 0.107 for their FNO and from 1.19 to 0.136 for their UNet. Our implementation, which writes the cached inverse as a Fourier multiplier, agrees with theirs (0.10719 against 0.10725). After one correction the first-step error is $1.50 \times 10^{-4}$, against $3.7 \times 10^{-4}$ for the network and $1.51 \times 10^{-4}$ for the scheme solved to convergence without a network: the corrected state is the solution of the scheme, and the network supplies the initial guess that makes a single cached iteration sufficient (from persistence, one iteration gives a rollout error of 0.387). The reported overhead of under 5% is relative to a network that already advances at the solver step;

Table 7: Burgers learned-refiner comparison, 128 identical ICs and nine windows. Each batch column uses matched complete online timing on V100S, including the frozen predictor; three warmed repeats. Costs depend on batch size. Metamizer and PDE-Refiner are existing methods in our adaptation to window forecasts.

| | nRMSE ($\times 10^{-2}$) ↓ | | Time (ms) | |
|---|---|---|---|---|
| Correction | batch 64 | batch 8 | batch 64 | batch 8 |
| Primitive NO, no refinement | 3.312 | 3.312 | 9.6 | 10.6 |
| Weighted L-BFGS-50 on the integral residual (ours) | **1.134** | **1.213** | 40.7 | 222.1 |
| Metamizer, fine-tuned on our objective, 20 steps | 1.304 | 1.304 | 308.2 | 147.7 |
| Metamizer, fine-tuned on our objective, 50 steps | 1.298 | 1.299 | 756.2 | 354.0 |
| PDE-Refiner, partial schedule | 3.236 | 3.236 | 18.9 | 20.7 |
| Metamizer, pretrained, pointwise objective | diverged | | | |

Bold indicates the lowest error.

per unit of physical time the method requires one network evaluation and one linear solve for every solver step.

Table 8 varies the frame spacing in their system: the network advances $s$ steps between corrections and the residual is written at $s\Delta t$. The gain erodes as the fixed-point factor of the cached iteration, $\max_k \frac{1}{2}k|u|/(1/(s\Delta t) + \frac{1}{2}(k^4 - k^2))$, approaches one and the truncation error of the scheme approaches the error of the network, and it reverses at $s = 32$. The released implementation loses the gain already at $s = 2$, which we attribute to the truncation of small singular values by the single-precision pseudo-inverse and not to the method; the table therefore uses the exact inverse. On our Burgers windows (frames $\Delta t = 0.01$ apart, about ten advective CFL units, fixed-point factor up to 4.9) one correction raises the first-frame error from 0.015 to 0.060, identically for their corrector and ours (cached Jacobians agree to $4 \times 10^{-7}$), and the rollout diverges by the sixth frame. The same correction applied to the true frame yields the same error, so the outcome is set by the consistency error of the one-step residual (0.02 for a single exactly solved step from true frames), as the diagnostic of Section 3.3 predicts. The scheme solved to convergence and run as a solver at this spacing attains 0.030, against 0.020 for the uncorrected FNO and 0.012 after integral refinement.

Table 8: One-step scheme correction on KS as a function of frame spacing (authors' FNO and test set, exact cached inverse). Mean relative error over the rollout; first corrected step in the last row.

| Steps between corrections $s$ | 1 | 2 | 4 | 8 | 16 | 32 |
|---|---|---|---|---|---|---|
| fixed-point factor | 0.13 | 0.23 | 0.40 | 0.72 | 1.36 | 2.77 |
| uncorrected | 0.453 | 0.452 | 0.452 | 0.451 | 0.456 | 0.466 |
| corrected | 0.107 | 0.109 | 0.117 | 0.158 | 0.299 | 0.595 |
| first step, corrected ($\times 10^{-3}$) | 0.15 | 0.27 | 0.50 | 1.05 | 4.4 | 30 |

### D.2 Pointwise against integral residual

Table 9: Pointwise against integral residual on a single forecast window, with matched L-BFGS-50, identical weighting and both backbones; only the residual differs. The pointwise residual uses fourth-order finite differences in time. The integral residual gives the lowest error in every setting, although the pointwise arm lowers its own objective to 2–44% of its initial value.

| System | Backbone | Raw | Pointwise | Integral |
|---|---|---|---|---|
| | | nRMSE ↓ | | |
| Burgers ($\times 10^{-2}$) | FNO | 2.710 | 3.132 | **2.410** |
| | Primitive NO | 4.512 | 4.200 | **2.875** |
| Advection ($\times 10^{-3}$) | FNO | 2.18 | 12.42 | **0.66** |
| | Primitive NO | 2.87 | 12.33 | **0.66** |
| Reaction–diffusion ($\times 10^{-4}$) | FNO | 2.3 | 2.1 | **0.3** |
| | Primitive NO | 3.5 | 2.1 | **0.3** |
| ADR ($\times 10^{-4}$) | FNO | 5.2 | 17.0 | **0.2** |
| | Primitive NO | 8.8 | 19.4 | **0.3** |
| Variable-coefficient ADE, ID ($\times 10^{-3}$) | FNO | 3.39 | 3.70 | **1.50** |
| | Primitive NO | 7.76 | 7.34 | **3.11** |
| Variable-coefficient ADE, statistics shift ($\times 10^{-2}$) | FNO | 2.768 | 2.729 | **1.809** |
| | Primitive NO | 3.087 | 2.884 | **1.897** |

Bold indicates the lowest error.

## E NAVIER–STOKES PROTOCOL AND TRANSPORT CORRECTION

The difficult-NS study replaces the original 48-trajectory, 20,000-single-window-update pilot with two independently trained FNOs (seeds 151 and 152), width 32, 20 modes and four layers, and two primitive NOs with the same seeds, frame and size (Appendix F). There are 1,024 training and 128 separate validation trajectories; the trainer never reads the existing short or extended test sets. New reference trajectories use $512^2$ FP64 integration, $\delta t = 10^{-4}$, and 41 saved frames at spacing 0.02, restricted spectrally to $128^2$. On the highest- and median-initial-RMS validation cases, halving the timestep and doubling the grid change projected nRMSE by $4.75 \times 10^{-6}$ and $6.25 \times 10^{-8}$, respectively.

Training uses batch eight and Adam: 10,000 single-window updates at learning rate $10^{-3}$, followed by random $K \in \{1, 2, 3\}$-window training with 1% input noise, gradient clipping at one and cosine decay from $3 \times 10^{-4}$ to $4.5 \times 10^{-5}$ over the remaining 70,000 updates. All 128 validation ICs are evaluated on three-window rollouts every 2,000 updates.

The independent extended dataset uses eight new initial conditions (different seed), $512^2$ FP64 integration, $\delta t = 10^{-4}$, and 100 saved intervals at spacing 0.02. Ten history frames precede nine ten-frame forecast windows. Halving the timestep and doubling the grid on two cases change mean projected error by $1.98 \times 10^{-5}$ and $5.86 \times 10^{-5}$; maximum frame-mean differences are $5.53 \times 10^{-5}$ and $1.41 \times 10^{-4}$. Preliminary low-order transport results on this dataset were inspected during development. It is an independent-IC stress test, not an untouched validation set for every development choice.

Given window $w^{(k)}$, reconstruct $v^{(k)}$ by periodic Poisson inversion and compute

$$w^{(k+1)} = \mathcal{T}[v^{(k)}; w_0], \qquad z_t + v^{(k)} \cdot \nabla z = \nu \Delta z, \quad z(0) = w_0.$$

Each pass uses the whole lagged velocity window and reconstructs vorticity causally from the same incoming state. The inexpensive version uses exact diffusion half steps, midpoint backtracing, four-point cubic spatial interpolation and linear temporal velocity interpolation. The finer version uses eight-point interpolation on a retained $256^2$ state, two substeps per saved interval, cubic temporal interpolation and three passes, preceded by $F_8$ correction. The first fine state is interpolated from the observed coarse field; subsequent high modes come from the correction, never the reference. The NO sees restricted $128^2$ history.

The original pilot diagnostic distinguished new forecast defects from inherited propagation. Restarting $LF_8$ from true history still left growing window errors. Fine evolution of its observed inherited

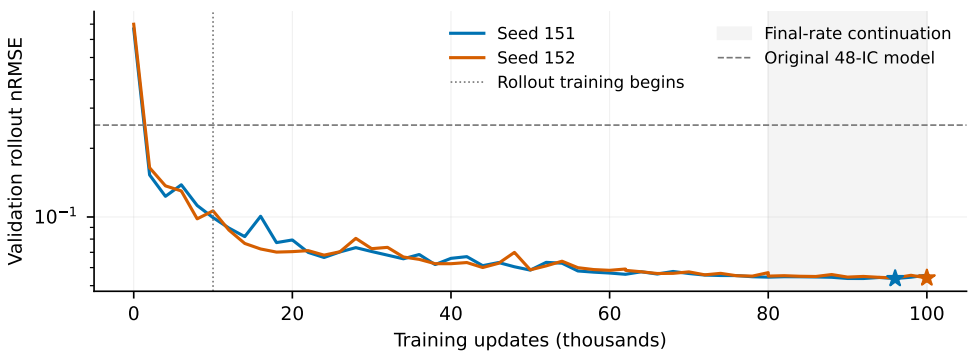


Figure 4: Difficult-NS training audit. Separate validation trajectories select checkpoints (stars); the dashed line evaluates the original 48-IC model on that same validation set. Curves show both trained seeds, including the learning-rate decay. Test errors never select checkpoints.

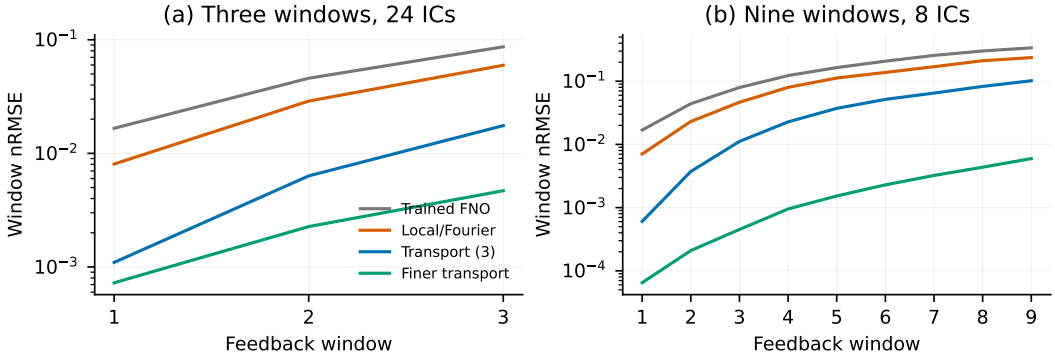


Figure 5: Navier–Stokes, FNO backbone, three-window and nine-window rollouts; bands span two training seeds. The transport correction reduces the error well below that of the uncorrected forecast and of a local correction. The finer configuration retains a $256^2$ state between windows.

error gave mean endpoint gains 0.983/0.992 over the next two windows. These directional, finite-time measurements motivated causal reconstruction but do not prove general contraction. The updated results use the stronger trained models throughout; the pilot's numerical scores are not mixed into the new comparison. For the new models, finer nine-window correction error grows from $6.49 \times 10^{-5}$ to $5.96 \times 10^{-3}$ (means over the two seeds). It is substantially smaller than raw error but does not exhibit the near-flat late behavior observed for Richards.

## F Backbone comparisons

This appendix supports the backbone comparisons of Section 3.4. On the two-dimensional problems, the primitive NO is the more accurate uncorrected forecast, and this holds under a larger FNO mode budget and a second training seed. On the smooth one-dimensional problems the FNO is more accurate. A one-dimensional problem with heterogeneous coefficients partly reproduces the two-dimensional pattern.

**Primitive NO in two dimensions.** The two-dimensional primitive NO extends the block of Appendix B (Eq. 4) to two dimensions. For Navier–Stokes it is placed in the FNO's frame, sharing its lift, coordinate channels, projection, last-frame anchor, width 32, 20 modes and four layers. Each block predicts five scalars per channel from the spatially pooled hidden state: two anisotropic diffusion rates, a two-component transport velocity and an inverse-Laplacian gain. It applies the corresponding closed-form Fourier multipliers on every mode of the periodic $128^2$ grid, and mixes the three branches with the FNO's learned spectral kernel through a $4C \to C$ pointwise mixer with a residual connection. The result has 6,626,090 parameters, against 6,563,754 for the FNO.

Both are trained with the same data, seeds, objective, schedule and validation-based checkpoint selection (Appendix E); the primitive NO is trained in a single uninterrupted run. Each seed takes about 5,000 s on a V100S, against roughly 2,200 s for the FNO. The Richards block uses the same construction on the zero-padded non-periodic domain, with 28.5M parameters for both backbones.

**Mode budget on Navier–Stokes.** The primitive branches act on all modes while the FNO kernel is truncated, so we trained FNOs with more modes, and with the same seeds and schedule. Validation three-window rollout nRMSE on the 128 validation trajectories:

| Backbone | Parameters | Seed 151 | Seed 152 |
|---|---|---|---|
| | | nRMSE ($\times 10^{-2}$) ↓ | |
| FNO, 20 modes (Table 5) | 6.56M | 5.372 | 5.406 |
| FNO, 20 modes, uninterrupted | 6.56M | 5.317 | 5.395 |
| FNO, 32 modes | 16.8M | 5.097 | 5.241 |
| FNO, 48 modes | 37.8M | 5.200 | 5.203 |
| Primitive NO, 20 modes | 6.63M | **4.984** | **4.986** |

Bold indicates the lowest error.

The uninterrupted FNO control reproduces the Table 5 seeds, so their continuation restarts do not explain the gap. More modes help modestly and then stall. On the eight-IC nine-window set the raw errors are 0.166/0.174 (FNO) and 0.161/0.169 (primitive), and the finer transport gives 0.0021 for both.

**Second training seeds.** Backbone comparison was repeated with a second training seed under identical data, budget and protocol (Table F). On the one-dimensional systems the matched FNO leads for both seeds; on the Richards development cohort the primitive NO leads for both seeds in the three regimes.

| | FNO | | Primitive NO | |
|---|---|---|---|---|
| System | Seed 1 | Seed 2 | Seed 1 | Seed 2 |
| Burgers ($\times 10^{-2}$) | **2.03** | **2.03** | 2.38 | 2.53 |
| Advection ($\times 10^{-3}$) | **2.66** | **2.61** | 5.00 | 4.55 |
| Reaction–diffusion ($\times 10^{-5}$) | **7.0** | **7.2** | 11.7 | 9.6 |
| ADR ($\times 10^{-4}$) | **1.19** | **1.20** | 2.50 | 2.72 |
| Richards, ID ($\times 10^{-3}$) | 7.2 | 7.0 | **6.2** | **6.0** |
| Richards, statistics shift ($\times 10^{-2}$) | 3.25 | 3.70 | **1.94** | **2.26** |
| Richards, correlation shift ($\times 10^{-3}$) | 10.1 | 8.9 | **8.5** | **7.8** |

**Transolver++ on Richards.** The official Transolver++ (width 256, five layers, 32 slices, 1.49M parameters) is trained under the Richards protocol: the same 800 trajectories, 100,000 single-window and 30,000 four-window updates. Each grid point is a token carrying the pressure window, log conductivity and recharge, with its coordinates as position. On the development test sets (40 ID trajectories, 32 per shift; not the Table 4 cohort), fine-tuned errors are:

| | | ID | Statistics shift | Correlation shift | |
|---|---|---|---|---|---|
| Backbone | Parameters | nRMSE ($\times 10^{-2}$) ↓ | | | Time (ms) |
| FNO | 28.5M | 0.721 | 3.25 | 1.01 | 38 |
| Primitive NO | 28.5M | **0.624** | **1.94** | **0.85** | 86 |
| Transolver++ | 1.49M | 3.07 | 5.54 | 3.05 | 201 |

Three inference seeds agree within $2 \times 10^{-4}$. It is the least accurate and the slowest backbone here; it is not size-matched and uses the shared training recipe, so this does not bound the architecture's achievable accuracy.

Table 10: Variable-coefficient advection–diffusion with per-trajectory log-Gaussian $v(x)$, $D(x)$ supplied to every backbone as two input channels. 128 held-out trajectories per regime, nine feedback windows. In distribution (ID); field log-standard deviation $\times 1.5$ (statistics shift); correlation length halved (correlation shift). Refinement is weighted L-BFGS on the mild residual with 50 or 200 iterations; correction cost is 0.11 and 0.44 s per trajectory for every backbone. FNO 0.55M, primitive NO 2.46M and Transolver++ 1.50M parameters; one trained seed per backbone.

| Regime | Backbone | Raw | L-BFGS-50 | L-BFGS-200 |
|---|---|---|---|---|
| | | nRMSE ($\times 10^{-2}$) ↓ | | |
| ID | FNO | **0.188** | **0.099** | **0.137** |
| | Primitive NO | 0.365 | 0.148 | 0.142 |
| | Transolver++ | 5.63 | 1.45 | 0.241 |
| Statistics shift | FNO | 6.39 | 3.69 | 1.25 |
| | Primitive NO | **4.18** | **2.61** | **1.19** |
| | Transolver++ | 9.40 | 5.92 | 1.44 |
| Correlation shift | FNO | **0.624** | **0.456** | **0.279** |
| | Primitive NO | 1.29 | 0.778 | 0.308 |
| | Transolver++ | 6.19 | 2.77 | 0.472 |

Bold indicates the lowest error in each regime.

**One-dimensional breadth with both backbones.** We used the same test ICs, correction, quadrature and corrected feedback; only the warm start differs. Floors are the mild map applied to reference windows.

| System | Correction | Floor | Primitive NO | | Matched FNO | |
|---|---|---|---|---|---|---|
| | | | Raw | Refined | Raw | Refined |
| | | nRMSE ↓ | | | | |
| Reaction–diffusion ($\times 10^{-5}$) | Picard-5 | 1.05 | 11.8 | 2.11 | 7.09 | 2.17 |
| ADR ($\times 10^{-4}$) | L-BFGS-50 | 2.14 | 2.56 | 2.82 | 1.22 | 2.27 |

For linear advection the mild map does not depend on the candidate, so a single Picard application reproduces the exact spectral propagator from the last frame and gives 0.0036 for either backbone; we therefore omit it from the refinement table.

**Supplied heterogeneous coefficients in one dimension.** To test the two-dimensional pattern in a controlled setting, we solve $u_t + v(x)u_x = (D(x)u_x)_x$ on a periodic 1,024-cell grid with per-trajectory log-Gaussian fields (log-standard deviations 0.3 and 0.5, correlation length 0.1) by pseudo-spectral RK4, verified against the closed-form constant-coefficient solution (relative error $3 \times 10^{-6}$). The fields enter each backbone as two static input channels; 2,400 training trajectories and 200,000 updates per backbone. The mild residual splits the operator into its spatial mean, in the Fourier symbol, and the heterogeneous remainder, in the forcing. The residual floor is $2.6 \times 10^{-4}$ ID, $4.9 \times 10^{-4}$ under the variance shift and $4.0 \times 10^{-4}$ under the correlation shift. With the unweighted objective, refinement gains only 4–29%; the low-wavenumber weighting of equation 2 is required, as on Burgers.

## G Richards formulation and numerical controls

Richards flow obeys $\partial_t \theta(\psi) = \nabla \cdot [K(\psi)\nabla(\psi + z)]$. The domain is a sloped $210\,\mathrm{m} \times 8\,\mathrm{m}$ soil section with van Genuchten storage/conductivity, no-flow left/bottom boundaries, a hydrostatic right boundary and top recharge. Log conductivity is a random field. The statistics shift changes its standard deviation from 0.8 to 1.2; the correlation shift changes horizontal/vertical lengths from $30/0.6\,\mathrm{m}$ to $60/0.3\,\mathrm{m}$ at unchanged marginal variance. All 12 trajectories per regime are disjoint from correction-design pilots. Forcing-extrapolation results are excluded because the reference was not consistently converged.

Table 11: Richards equation: neural and classical solvers compared. Mean rollout nRMSE on the 12 held-out trajectories per regime, and total time per 38-day rollout at batch one; time ranges span the three regime means. All classical runs converge. CPU and GPU classical runs agree to $2 \times 10^{-7}$ in nRMSE and are shown as one row.

| | ID | Statistics shift | Correlation shift | Time (s) | |
|---|---|---|---|---|---|
| Method | nRMSE ($\times 10^{-2}$) ↓ | | | GPU | CPU |
| FNO | 0.676 | 3.437 | 0.977 | 0.037–0.038 | – |
| FNO + line/column | 0.311 | 0.911 | 0.715 | 0.185–0.191 | – |
| Primitive NO | 0.581 | 1.891 | 0.743 | 0.078–0.080 | – |
| Primitive NO + column | 0.377 | 1.131 | 0.648 | 0.136–0.139 | – |
| Primitive NO + line/column | **0.265** | **0.744** | **0.500** | 0.227–0.233 | – |
| Classical, $\Delta t = 12\,\mathrm{h}$ | 1.048 | 0.918 | 0.981 | 0.371–0.398 | 7.8–8.5 |
| Classical, $\Delta t = 6\,\mathrm{h}$ | 0.527 | 0.444 | 0.477 | 0.667–0.720 | 13.9–15.5 |
| Classical, $\Delta t = 3\,\mathrm{h}$ | 0.216 | 0.192 | 0.177 | 1.3–1.4 | 25.8–27.8 |

Bold indicates the lowest error among different backbones.

The residual is the mixed-form discrete mass balance (Celia et al., 1990). The line/column correction applies two steps in sequence. First, a vertical-line Jacobian solve corrects each column's local structure. The column step sums residuals over depth, which exposes storage and horizontal transport imbalance, and lifts one amplitude per column and hourly node with a depth-dependent profile that preserves internal vertical flux to first order. A 48-hour window has 5,040 correction amplitudes versus 408,240 free field values, and the small solve is causal in time. The correction takes a fixed number of steps. In contrast, the classical control iterates the full spatial system to convergence.

**Supporting accuracy–cost controls.** Following McGreivy & Hakim (2024), we compare against converged classical solvers to delimit where the neural approach pays off. The classical solver at $\Delta t = 3\,\mathrm{h}$ is more accurate than every neural result in all three regimes, at about six times the cost of the corrected network (Table 11). The GPU classical solver uses FP64 analytic inexact Newton with BiCGSTAB, preconditioned by line and column solves of the same structure as our correction. 108 trajectory/time-step configurations converge at nonlinear-residual and pressure-correction thresholds of $10^{-6}$. The maximum CPU/GPU discrepancy in trajectory nRMSE is $2.12 \times 10^{-7}$, and full pressure trajectories on three cases differ by at most $4.66 \times 10^{-6}\,\mathrm{m}$ at every time step. Timings follow a complete trajectory warmup and use three repeats; they include all online assembly, solves, stopping checks and transfers; compilation, graph setup, loading and scoring are excluded. The classical setup takes 1–34 s per configuration depending on caching.

Classical characteristic transport is a particularly relevant Navier–Stokes control because it uses the same interpolation machinery without the NO. On the 24-case three-window comparison, a $128^2$ stepper with two velocity updates per 0.02 step gives 0.00866 at 16.63 ms, against 0.00833 at 16.86 ms for three-pass transport on the FNO forecast. On the nine-window dataset, a $256^2$ classical stepper with three updates per 0.01 step gives 0.00223 at 176.93 ms against 0.00212 at 163.07 ms for the finer neural correction. Thus the neural seed helps over persistence at a fixed window-pass budget, while classical stepping remains competitive.

For Burgers, the matched classical control with $N = 128$ gives 0.02682 at 28.46 ms, against 0.0266 at 9.80 ms for the projected neural correction, so the neural correction matches it at a third of the cost. With $N = 1024$ the classical solver gives 0.00131 at 147.38 ms, so at strict error targets the numerical solver is preferable. Grid transfers and stepping are included. These controls share the reference and cohort of Figure 3. Reproduction commands and complete raw experiment provenance are maintained with the code.

## H Refinement diagnostic discussion

The floor $F$ of Section 3.3 is one application of the residual map to a true window. Its converged counterpart, $E^*$, is where refinement lands when run to convergence from a true window, measured as the same normalized departure. Refinement carries any forecast to about the same place, so with

$E$ the uncorrected and $E'$ the refined error the expected improvement is

$$\frac{E - E'}{E} \approx 1 - \frac{E^*}{E}. \tag{5}$$

Section 4.4 tests the diagnostic on 144 refinement runs. Points are generated along four axes. The system varies over 1D Burgers, advection, reaction–diffusion and the advection–diffusion–reaction problem. The quadrature order varies over 1, 2, 4 and 8, which moves the floor substantially at a fixed forecast error; refinement always uses the same quadrature as the floor measurement. The forecast error is swept by perturbing true windows with a controlled root-mean-square amplitude, eight amplitudes per system, which varies the ratio continuously without retraining anything. Finally, real forecasts from the trained backbones supply anchor points, confirming that the synthetic arm lands where real forecasts of comparable ratio do.

The transition is monotone across eight bins of $F/E$, with mean improvement $+0.82$, $+0.73$, $+0.57$, $+0.36$, $0.00$, $-0.62$, $-1.64$ and $-3.10$; it crosses zero in the bin $[0.8, 1.2]$. 73 runs with $F/E < 0.5$ improve and 91% of those with $F/E > 1.5$ degrade. Refinement lands in nearly the same place regardless of where it starts: sweeping the incoming error over two orders of magnitude changes the refined error by a median of 8.7% within a system and quadrature order, and for linear advection not at all, which is what identifies $E^*$ as a fixed point. This fixed-point behaviour is established here only for the four one-dimensional systems tested. Predicting improvement by $1 - E^*/E$ rather than $1 - F/E$ lowers the mean absolute error from 0.469 to 0.152 and raises the correlation from 0.840 to 0.963. The median ratio $E^*/F$ is 1.30, with a range of 0.31 to 1.68, so the floor is a proxy good to about a factor of three.

**Perturbation spectrum.** Holding the amplitude fixed and varying the spectral content of the perturbation isolates what the reweighting of equation 2 achieves. Both objectives are normalized to equal magnitude, and the residual reduction actually attained is recorded, so that an insensitive objective is distinguishable from a stalled optimizer. On advection, reaction–diffusion and the advection–diffusion–reaction problem the flat and reweighted objectives agree to four digits. On Burgers they differ: the fraction of the error removed rises from 0.214 to 0.514 for a pure translation and from 0.762 to 0.909 for a low-$k$ perturbation, but falls from 0.743 to 0.615 for a broadband one. At four times the budget the low-$k$ advantage closes while the translation advantage persists. Reweighting is therefore a remedy for displacement rather than a general improvement.

**Weighting controls.** The control quoted in Section 4.5 was rerun with the two objectives placed on an equal footing in two independent ways, since the unweighted branch of the objective returns a value three orders of magnitude smaller and could in principle stall the optimizer. It does not: the published runs use Adam, whose update is invariant to a global rescaling of the loss, and a flat-weight control reproduces the unweighted arm to three or four significant digits at every step size. At a matched step size and 100 steps the unweighted residual gives 0.0231 and the reweighted residual 0.0143; tuning the step size separately for each gives 0.0145 and 0.0110. At the largest step size the unweighted arm attains the lower residual, $2.2 \times 10^{-3}$ against $3.4 \times 10^{-3}$, and the worse forecast. The amplification of the lowest band by the unweighted objective in Figure 1b depends on the step size; the advantage of reweighting in that band does not.

# I Error accumulation for backbones

Table 3 in the main text shows one backbone per system. Table 12 gives the same first-to-last-window errors for the backbones we trained, where the pattern is the same for each. Without correction the error compounds on the advection–diffusion problem under the statistics shift and on the Richards equation in the three regimes, while it falls or stays level on Burgers and on the advection–diffusion problem in distribution and under the correlation shift. With corrected feedback, the last-window error always lies below the uncorrected one. On Burgers, the advection–diffusion problem, and Navier–Stokes the corrected rollouts of different backbones end at nearly the same error. On turbulent flow both the uncorrected and the corrected errors grow, as chaotic dynamics require.

**Error recurrence.** Writing $e_n$ for the error after window $n$, we fit $e_{n+1} = \rho e_n + \epsilon$ by least squares to the per-window errors and bootstrap over trajectories to obtain intervals on $\rho$. Here $\rho$ is

Table 12: Error in the first and the last window of a rollout, from the runs behind Tables 1, 4 and 10, and for Navier–Stokes from the nine-window extended dataset (Appendix E). Nine windows, except the Richards equation (19 windows, 38 days). Corrections as in Table 3; Navier–Stokes is the two-seed mean under the finer transport correction.

| System | Backbone | Uncorrected:first → last | Corrected: first → last |
|---|---|---|---|
| | | nRMSE ($\times 10^{-2}$) ↓ | |
| Burgers | FNO | 2.67 → 2.00 | 2.52 → 0.66 |
| | Primitive NO | 4.10 → 2.97 | 2.80 → 0.63 |
| | Primitive NO (padded) | 3.08 → 2.30 | 2.63 → 0.63 |
| | Transolver++ | 6.32 → 5.72 | 3.71 → 0.80 |
| Variable-coefficient ADE, ID | FNO | 0.33 → 0.14 | 0.12 → 0.13 |
| | Primitive NO | 0.76 → 0.24 | 0.14 → 0.13 |
| | Transolver++ | 5.00 → 4.67 | 0.38 → 0.17 |
| Variable-coefficient ADE, statistics shift | FNO | 2.74 → 8.39 | 0.94 → 1.23 |
| | Primitive NO | 3.12 → 5.00 | 1.19 → 0.97 |
| | Transolver++ | 6.27 → 10.23 | 1.89 → 1.08 |
| Variable-coefficient ADE, correlation shift | FNO | 0.79 → 0.58 | 0.19 → 0.32 |
| | Primitive NO | 1.47 → 1.29 | 0.25 → 0.35 |
| | Transolver++ | 5.28 → 5.42 | 0.68 → 0.39 |
| Richards equation, ID | FNO | 0.29 → 1.00 | 0.24 → 0.31 |
| | Primitive NO | 0.30 → 0.82 | 0.25 → 0.28 |
| Richards equation, statistics shift | FNO | 0.55 → 5.84 | 0.49 → 0.90 |
| | Primitive NO | 0.48 → 2.62 | 0.39 → 0.82 |
| Richards equation, correlation shift | FNO | 0.52 → 1.28 | 0.45 → 0.71 |
| | Primitive NO | 0.45 → 0.89 | 0.37 → 0.48 |
| Navier–Stokes, turbulent | FNO | 1.69 → 33.7 | 0.006 → 0.60 |
| | Primitive NO | 1.53 → 33.3 | 0.007 → 0.59 |

the fraction of inherited error that each window passes on and $\epsilon$ the fresh error it adds; for $\rho < 1$ the error approaches $\epsilon/(1-\rho)$, and for $\rho \geq 1$ it does not settle. The Richards cohorts have 19 windows and 12 trajectories per regime.

Table 13: Fitted error recurrence on the Richards equation. Intervals are 95% bootstrap over trajectories. The plateau $\epsilon/(1-\rho)$ is the level the error approaches when $\rho < 1$. For uncorrected rollouts, $\rho$ is close to 1.

| Backbone, regime | Correction | $\rho$ | $\epsilon$ | Plateau |
|---|---|---|---|---|
| Primitive NO, ID | none | 0.967 [0.91, 1.01] | $4.8 \times 10^{-4}$ | 0.0146 |
| Primitive NO, ID | line/column | 0.842 [0.66, 0.90] | $4.4 \times 10^{-4}$ | 0.0028 |
| Primitive NO, statistics shift | none | 0.881 [0.81, 0.94] | $3.4 \times 10^{-3}$ | 0.0284 |
| Primitive NO, statistics shift | line/column | 0.754 [0.60, 0.88] | $2.1 \times 10^{-3}$ | 0.0084 |
| FNO, ID | none | 0.974 [0.94, 1.00] | $5.6 \times 10^{-4}$ | 0.0217 |
| FNO, ID | line/column | 0.780 [0.69, 0.85] | $7.2 \times 10^{-4}$ | 0.0033 |
| FNO, statistics shift | none | 0.969 [0.94, 0.99] | $4.0 \times 10^{-3}$ | 0.1265 |
| FNO, statistics shift | line/column | 0.692 [0.62, 0.78] | $3.0 \times 10^{-3}$ | 0.0099 |

On the Richards equation (Table 13), uncorrected rollouts have $\rho$ between 0.88 and 0.97, with intervals reaching 1 in distribution; corrected rollouts have $\rho$ between 0.69 and 0.84, every interval excluding 1. This suggests that corrected feedback primarily reduces the propagation of inherited error, producing asymptotic error levels 3–13 times lower than those estimated for the uncorrected rollouts. The injection term $\epsilon$ changes much less than $\rho$: it rises for the FNO in distribution and falls by 25–38% under the statistics shift, so the correction acts mainly on inherited error rather than on what each window introduces. The corrected plateaus agree with the observed last-window errors

(Table 12). The fit explains 0.99 or more of the variance for the uncorrected series and 0.72 to 0.98 for the corrected ones, which are flatter and therefore harder to fit.

On the shifted advection–diffusion problem the same fit gives, for the FNO, $\rho = 0.726$ uncorrected and 0.034 after 200 L-BFGS iterations. The uncorrected error is bounded but approaches a high level, and full refinement resets the inherited error almost completely, whereas on the Richards equation the line/column correction only reduces it ($\rho = 0.69$–$0.84$). On turbulent Navier–Stokes $\rho$ exceeds 1, as chaotic dynamics require, and refinement reduces $\epsilon$ by a factor of 145. Turbulent flow differs qualitatively: chaotic dynamics amplify any perturbation, $\rho$ exceeds 1, and no bounded level exists, so the remaining error must grow. Refinement can then only reduce $\epsilon$, which can still matter. After nine windows the corrected Navier–Stokes error (0.0060) is 56 times lower than the uncorrected error (0.34), and it remains below the uncorrected error after a single window (0.017). Neural operators are known to struggle on turbulent flow, whose energy spans many scales, and their rollouts drift within a few windows (Um et al., 2020; Kochkov et al., 2021; Lippe et al., 2023); a 56-fold lower error materially extends the horizon over which such a rollout is usable.

## J Navier–Stokes additional results

The correction settings for turbulent Navier–Stokes were chosen while inspecting part of the test cohort. To remove this dependence, we recorded the correction hyperparameters from Table 5, together with the checkpoints, and committed this record before generating any new data. We then generated 24 trajectories with a previously unused seed and evaluated each arm once, with no tuning afterwards.

We found similar conclusions (Table 14) to what was observed before. The ordering of the arms is unchanged for both backbones and both training seeds, and under three transport passes the two backbones agree within 0.2%. Errors are lower throughout, as this cohort is easier than the published one: three transport passes remove 85.9% of the FNO error (83% on the published cohort), and the uncorrected advantage of the primitive NO rises from 7.8% to 9.1%. The residual floor, averaged over the rollout, is $0.894 \times 10^{-2}$ and independent of quadrature order ($0.8945 \times 10^{-2}$, $0.8938 \times 10^{-2}$ and $0.8940 \times 10^{-2}$ at orders 1, 2 and 4); per window it grows from $0.212 \times 10^{-2}$ to $1.854 \times 10^{-2}$.

Table 14: Turbulent Navier–Stokes on a new cohort of 24 trajectories, generated after all correction settings were frozen, against the published cohort of Table 5. Three windows; entries are the mean over two training seeds.

| Correction | FNO | | Primitive NO | |
|---|---|---|---|---|
| | Frozen cohort | Published | Frozen cohort | Published |
| | nRMSE ($\times 10^{-2}$) $\downarrow$ | | | |
| None | 4.784 | 4.97 | 4.347 | 4.58 |
| Adam-100 | 3.637 | 3.94 | 3.278 | 3.58 |
| Local correction | 2.812 | 3.22 | 2.664 | 3.02 |
| Transport, 1 pass | 1.177 | 1.29 | 1.087 | 1.18 |
| Transport, 3 passes | 0.672 | 0.833 | 0.671 | 0.831 |
| Transport, finer | 0.255 | 0.256 | 0.255 | 0.256 |