Predictions and Corrections: Neural Predictors with Solver-based Correction for ODEs and PDEs

Reza Akbarian Bafghi[1] **Hasitha De Silva**[2] **Joseph G. Kocheemoolayil**[3]
John L. Davenport[3] **Sheik Dawood**[3] **Nastaran Shahmansouri**[3]
Adrian Lew[3] **Sarah Osentoski**[3] **Hardik Kabaria**[3]
Maziar Raissi[4]

[1]University of Colorado, Boulder [2]George Mason University [3]Vinci4D
[4]University of California, Riverside

Abstract

Neural predictors offer fast surrogates for simulating dynamical systems but suffer from error accumulation over long horizons. We study a unified predict-correct paradigm in which a learned one-step neural forecast is post-processed by a physics-based solver step ("correction"). The same idea applies to ordinary differential equations (ODEs) and partial differential equations (PDEs). We examine how training targets (losses) for the predictor, e.g., oracle RK4 or trapezoid/Crank-Nicolson (CN) targets-influence the quality of both the raw neural rollout and the corrected rollout. Experiments span classic ODEs (linear decay, Lorenz-63, Lorenz-96) and a 2D forced heat equation with homogeneous Dirichlet boundaries. Across settings, the correction step markedly improves stability and accuracy, often recovering near-reference solutions; the choice of training target shapes the magnitude of improvements and robustness.

1 Introduction

Predicting the evolution of dynamical systems with learned surrogates has become a popular alternative to traditional numerical solvers: once trained, neural models can offer inexpensive function evaluations and are straightforward to deploy on modern accelerators. However, long-horizon rollouts often accumulate error and drift from physical constraints, especially when supervision is local in time. A classical remedy in numerical analysis is the predictor-corrector (PC) paradigm [Hairer et al., 2009, Butcher, 2008, Ascher and Petzold, 1998]: a fast predictor proposes the next state, then an implicit (or otherwise higher-fidelity) corrector refines that proposal using the governing equations. In this work we adopt this principle in a neural setting: a learned one-step network produces a forecast ("prediction"), which is then post-processed by a physics-based solver step ("correction") such as the trapezoidal or Crank-Nicolson (CN) schemes [Crank and Nicolson, 1947b, LeVeque, 2007].

We study this predict-correct approach across both ordinary differential equations (ODEs) and partial differential equations (PDEs), with a unified emphasis on how the training target (loss) for the neural predictor influences (i) the raw neural rollout and (ii) the corrected rollout. Concretely, we consider oracle Runge-Kutta 4 (RK4) one-step targets [Butcher, 2008] as well as implicit trapezoidal/Crank-Nicolson (CN) targets [Crank and Nicolson, 1947b, LeVeque, 2007] that mirror the correction used at inference. For PDEs, we focus on a forced 2D heat equation with homogeneous Dirichlet boundary conditions [LeVeque, 2007]; for ODEs, we revisit scalar decay, Lorenz-63, and Lorenz-96 [Lorenz, 1963, 1996, Lorenz and Emanuel, 1998].

Our experiments indicate that the solver-based correction step substantially improves stability and accuracy across settings. Nevertheless, the choice of predictor loss shapes how much benefit the corrector can extract. For chaotic low-dimensional ODEs, RK4 targets are strong; for linear parabolic

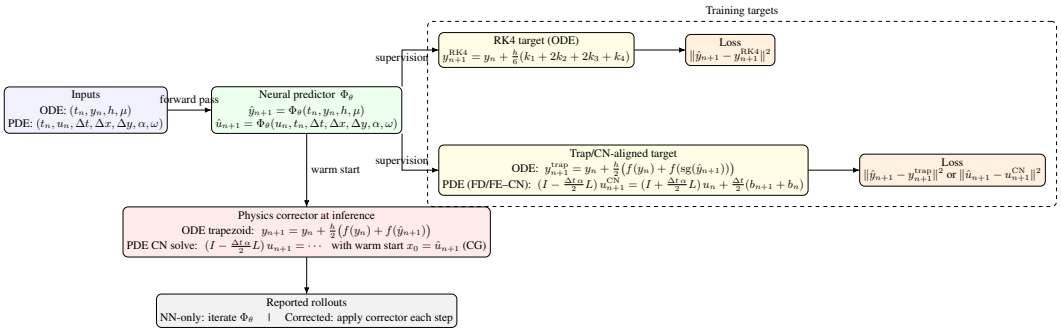


Figure 1: Master predictor-corrector schematic. Top/right: supervised targets (RK4; trap/CN). Bottom: inference-time corrector using the predictor as warm start (trapezoid; CN).

PDEs, CN-aligned targets provide a natural fit. Importantly, even when the predictor alone drifts, the corrector can recover solutions that closely match trusted references.

Problem setting and goal. We consider state sequences $\{u_n\}$ evolving under known physics, either in finite dimension (ODEs) or on a spatial grid (PDEs). Given a step size h, a neural predictor $\hat{u}_{n+1} = \Phi_\theta(u_n, h, \cdot)$ proposes the next state from side information. At inference, we apply a physics-based correction (e.g., trapezoid/CN) that refines $\hat{u}_{n+1}$ using the governing equations [Ascher and Petzold, 1998, LeVeque, 2007]. We ask: how do different training targets for Φ_θ impact accuracy and robustness of both the predictor-only rollout and the corrected rollout?

Contributions. Building on recent neural PC ideas, we:

- present a unified predict-correct framework spanning ODEs (linear decay, Lorenz-63, Lorenz-96) and a 2D forced heat equation with homogeneous Dirichlet boundaries [LeVeque, 2007, Lorenz, 1963, 1996, Lorenz and Emanuel, 1998];
- compare training targets for the predictor-oracle RK4 vs. trapezoid/CN-and measure their effects on NN-only vs. corrected rollouts [Butcher, 2008, Crank and Nicolson, 1947b];
- demonstrate that solver-based corrections (trapezoid/CN) markedly improve long-horizon accuracy, often recovering near-reference solutions even when the predictor alone drifts, in line with trends in hybrid ML-physics schemes [Bar-Sinai et al., 2019, Kochkov et al., 2021].

We focus on fixed-step integrators and single-step corrections, leaving adaptive stepping, multi-iteration corrections, and broader PDE classes for future work. Our goal is a coherent account of predictions and corrections: neural network prediction, solver correction, and the role of the predictor's loss in enabling effective post-processing by physics-based steps.

Related work. Our study connects to several strands: (i) neural ODE/sequence models and operator learning for dynamical systems surrogacy [Chen et al., 2018, Li et al., 2021, Lu et al., 2021]; (ii) physics-informed and physics-augmented learning, where governing equations guide losses or post-processing [Raissi et al., 2019a, Bar-Sinai et al., 2019]; (iii) predictor-corrector and implicit integrators in numerical analysis, which inspire stable inference-time corrections [Hairer et al., 2009, Ascher and Petzold, 1998, Butcher, 2008]; and (iv) hybrid solvers that combine learned predictors with classical steps to improve stability and fidelity [Sanchez-Gonzalez et al., 2020, Pfaff et al., 2021, Kochkov et al., 2021, Fablet et al., 2021]. Prior works show that coupling learned forecasts with physics-based updates can reduce drift and enhance generalization; our contribution is to unify ODE and PDE cases under a single predict-correct view and to systematically assess how the choice of predictor target (RK4 vs. trapezoid/CN) shapes both NN-only and corrected rollouts.

2 Method

We study a unified predict-correct pipeline for dynamical systems, spanning both ODEs and PDEs. A learned one-step neural predictor proposes the next state; a physics-based corrector refines this proposal using the governing equations [Hairer et al., 2009]. We highlight how the training target (loss) used for the predictor influences (i) raw neural rollouts and (ii) corrected rollouts.

2.1 Dynamics and notation

ODEs. We consider autonomous ODEs $\dot{y} = f(y)$ and a fixed step size $h > 0$. Let y_n denote the state at time $t_n = nh$. A neural predictor $\hat{y}_{n+1} = \Phi_\theta(y_n, h)$ forecasts the next state.

PDEs. 2D forced heat equation on $[0, 1]^2$ with homogeneous Dirichlet boundaries; we use a 5-point Laplacian and a fixed-step time discretization (details in Appendix B) [LeVeque, 2007, Strikwerda, 2004]. The neural predictor outputs a full-grid next state given the current field and scalar metadata.

2.2 Training targets for the predictor

We consider two families of supervised one-step targets: (i) an oracle RK4 step, which approximates the continuous-time flow map over one step; and (ii) trapezoid/CN-aligned targets, which mirror the implicit equation solved by the corrector and can be viewed as minimizing its residual. All targets are computed from the known right-hand side; further discussion is given in Appendix B.1.

2.3 Inference-time correction

At rollout we compare an NN-only trajectory (iterating the predictor) with a predictor+corrector trajectory that applies a physics-based update at each step. For ODEs we use a trapezoidal update [Butcher, 2008, Hairer et al., 2009], given $\hat{y}_{n+1}$, set

$$y_{n+1}^{\mathrm{corr}} = y_n + \tfrac{h}{2}\big[f(y_n) + f(\hat{y}_{n+1})\big].$$

For PDEs we use a Crank–Nicolson (CN) correction: given $\hat{u}_{n+1}$ as a warm start, we solve the CN system (Appendix B) for u_{n+1}, enforcing Dirichlet boundary zeros at each iterate [Crank and Nicolson, 1947b, LeVeque, 2007].

2.4 Neural predictor

For ODEs we use a small MLP; for the PDE we use a compact CNN that ingests the current field and scalar channels (time, step sizes, coefficients) and outputs a full-grid next state. Dirichlet zeros are enforced on the PDE network output to match boundary conditions. See Appendix B.

3 Results

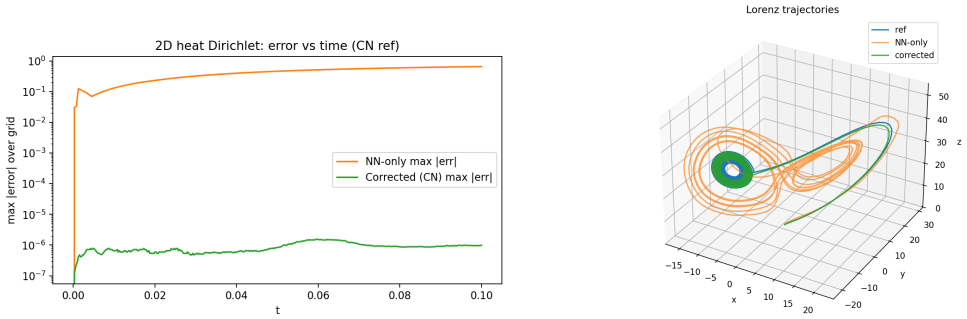
We evaluate our unified predict-correct scheme on both systems of ODEs and a two-dimensional parabolic PDE. For each task, we compare a standard neural network (NN)-only predictor to a predictor+corrector approach, where each NN output is refined using a physics-based numerical method (trapezoid or CN) at every step. The influence of the predictor's training target (e.g., RK4 vs. trapezoid/CN) is also assessed.

3.1 Experimental Setup

Our experiments span three ODE systems scalar decay, Lorenz-63, and Lorenz-96, and the 2D forced heat equation with Dirichlet boundaries. We use fixed-step rollouts and train predictors on either RK4, trapezoid, or CN targets as appropriate. Both NN-only and predictor+corrector rollouts are compared for each system. See Appendix B.

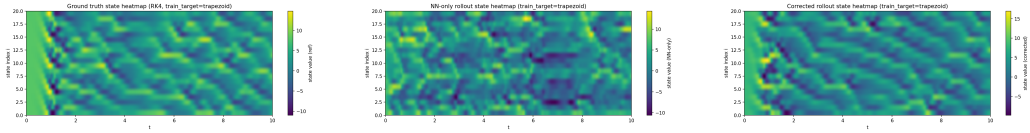
3.2 PDE Results: 2D Heat Equation

In the PDE setting, the impact of correction is dramatic. The CN-corrected predictions closely track the reference, while the NN-only model accumulates drift and increasingly diverges over time. For further discussion, see Appendix C. Figure 2a (left) quantifies stability and accuracy: the NN-only rollout's maximum absolute error rapidly increases by orders of magnitude, whereas applying the CN-based corrector holds errors at the level of the numerical reference. For additional error curves, centerline comparisons, and extended visualizations, see Appendix C.



(a) 2D heat: max absolute error v. time (CN reference). (b) Lorenz-63: 3D trajectory overlay (RK4 reference).

Figure 2: Error reduction and trajectory fidelity with solver correction. (a) On the 2D heat equation, NN-only error quickly grows and drifts, while the CN-corrected rollout remains stable. (b) Lorenz-63: the PC-corrected trajectory tightly follows the true attractor across lobe transitions, whereas the NN-only diverges significantly during chaotic excursions. The solver-based corrector thus enhances stability and fidelity by orders of magnitude in both PDE and chaotic ODE regimes.



(a) Ground truth (b) NN-only (c) PC-corrected

Figure 3: Lorenz-96 (N=20), trapezoid target: state heatmaps over time. The NN-only predictor exhibits phase lags and amplitude smearing, but the corrected output restores key spatial-temporal structures and closely aligns with the ground truth.

3.3 ODE Results

Predictor+corrector rollouts generally reduce error compared to NN-only, especially in chaotic ODEs. In Lorenz-96, Figure 3 shows state evolution for ground truth, NN-only, and corrected predictions. While the NN-only results display phase lag and amplitude smearing, the corrector recovers dominant structures and dynamics much more accurately, although rare spikes in maximum error remain (Table 1). Figure 2 (right) illustrates Lorenz-63 rollouts in three-dimensional state space. The corrected trajectory stays close to the true attractor across lobe transitions, while NN-only diverges markedly during chaotic excursions. See Appendix D.

Takeaways. In the ODE and PDE systems we study, solver-based correction improves rollout stability and typically reduces error relative to NN-only. The extent of improvement depends on the system and the predictor target: RK4 targets are especially effective for the low-dimensional chaotic ODEs considered here, while CN and trapezoid targets are a natural choice for our linear parabolic PDE and simple ODEs.

4 Conclusion and Discussion

We studied the predict-correct paradigm for dynamical systems, combining a neural one-step predictor with a physics-based solver at inference. Across diverse ODEs and a 2D heat equation, solver corrections (trapezoid or CN) significantly stabilized rollouts and restored reference-level accuracy, outperforming NN-only approaches. Our results highlight the importance of aligning the predictor's training target with the intended correction method: RK4 targets suit chaotic ODEs with trapezoidal correctors, while CN targets work best for parabolic PDEs. Even when NN-only predictions drift, an appropriate corrector can dramatically reduce error. Practically, moderate step sizes and, if required, multiple corrector iterations are recommended for challenging systems. While our analysis is limited to fixed-step integrators and a small set of examples, directions for future work include adaptive strategies, more complex systems, and richer predictor architectures. In summary, matching neural predictors with compatible solver corrections provides a robust, unified approach for accurate and stable surrogates in ODE and PDE modeling.

References

U. M. Ascher and L. R. Petzold. *Computer Methods for Ordinary Differential Equations and Differential-Algebraic Equations*. Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA, 1998. doi: 10.1137/1.9781611971392.

Y. Bar-Sinai, S. Hoyer, J. Hickey, and M. P. Brenner. Learning data-driven discretizations for partial differential equations. *Proceedings of the National Academy of Sciences*, 116(31):15344–15349, 2019. doi: 10.1073/pnas.1814058116.

J. C. Butcher. *Numerical Methods for Ordinary Differential Equations*. John Wiley & Sons, Chichester, UK, 2nd edition, 2008. ISBN 978-0-470-72335-7. doi: 10.1002/9780470753767.

R. T. Q. Chen, Y. Rubanova, J. Bettencourt, and D. Duvenaud. Neural ordinary differential equations. *Advances in Neural Information Processing Systems*, 2018.

J. Crank and P. Nicolson. A practical method for numerical evaluation of solutions of partial differential equations of the heat-conduction type. *Mathematical Proceedings of the Cambridge Philosophical Society*, 43(1):50–67, 1947a.

J. Crank and P. Nicolson. A practical method for numerical evaluation of solutions of partial differential equations of the heat-conduction type. *Mathematical Proceedings of the Cambridge Philosophical Society*, 43(1):50–67, 1947b. doi: 10.1017/S0305004100023197.

R. Fablet, B. Chapron, L. Drumetz, E. Mémin, O. Pannekoucke, and F. Rousseau. Learning variational data assimilation models and solvers. *Journal of Advances in Modeling Earth Systems*, 13(10): e2021MS002572, 2021. doi: 10.1029/2021MS002572.

E. Hairer, S. P. Nørsett, and G. Wanner. *Solving Ordinary Differential Equations I: Nonstiff Problems*, volume 8 of *Springer Series in Computational Mathematics*. Springer, Berlin, Heidelberg, 2nd rev. ed. edition, 2009. doi: 10.1007/978-3-540-78862-1.

M. R. Hestenes and E. Stiefel. Methods of conjugate gradients for solving linear systems. *Journal of Research of the National Bureau of Standards*, 49(6):409–436, 1952. doi: 10.6028/JRES.049.044.

D. Kochkov, J. A. Smith, A. Alieva, Q. Wang, M. P. Brenner, and S. Hoyer. Machine learning–accelerated computational fluid dynamics. *Proceedings of the National Academy of Sciences*, 118 (21):e2101784118, 2021. doi: 10.1073/pnas.2101784118.

R. J. LeVeque. *Finite Difference Methods for Ordinary and Partial Differential Equations: Steady-State and Time-Dependent Problems*. Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA, 2007. ISBN 978-0-89871-629-0. doi: 10.1137/1.9780898717839.

Z. Li, N. B. Kovachki, K. Azizzadenesheli, B. Liu, K. Bhattacharya, A. M. Stuart, and A. Anandkumar. Fourier neural operator for parametric partial differential equations. In *International Conference on Learning Representations (ICLR)*, 2021. URL `https://openreview.net/pdf?id=roNqYL0_XP`.

E. N. Lorenz. Deterministic nonperiodic flow. *Journal of the Atmospheric Sciences*, 20(2):130–141, 1963.

E. N. Lorenz. Predictability: A problem partly solved. *Proceedings of the Seminar on Predictability*, 1:1–18, 1996.

E. N. Lorenz and K. A. Emanuel. Optimal sites for supplementary weather observations: Simulation with a small model. *Journal of the Atmospheric Sciences*, 55(3):399–414, 1998. doi: 10.1175/1520-0469(1998)055<0399:OSFSWO>2.0.CO;2.

L. Lu, P. Jin, G. Pang, Z. Zhang, and G. E. Karniadakis. Learning nonlinear operators via DeepONet based on the universal approximation theorem of operators. *Nature Machine Intelligence*, 3(3): 218–229, 2021. doi: 10.1038/s42256-021-00302-5.

T. Pfaff, M. Fortunato, A. Sanchez-Gonzalez, and P. W. Battaglia. Learning mesh-based simulation with graph networks. In *International Conference on Learning Representations (ICLR)*, 2021. URL `https://arxiv.org/abs/2010.03409`.

M. Raissi, P. Perdikaris, and G. 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, 2019a. doi: 10.1016/j.jcp.2018.10.045.

M. Raissi, P. Perdikaris, and G. 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, 2019b.

Y. Saad. *Iterative Methods for Sparse Linear Systems*. Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA, 2 edition, 2003. ISBN 978-0-89871-534-7. doi: 10.1137/1.9780898718003.

A. Sanchez-Gonzalez, J. Godwin, T. Pfaff, R. Ying, J. Leskovec, and P. W. Battaglia. Learning to simulate complex physics with graph networks. In *Proceedings of the 37th International Conference on Machine Learning (ICML)*, pages 8459–8468, 2020. URL `https://proceedings.mlr.press/v119/sanchez-gonzalez20a.html`.

J. C. Strikwerda. *Finite Difference Schemes and Partial Differential Equations*. Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA, 2 edition, 2004. ISBN 0-89871-567-9. doi: 10.1137/1.9780898717938.

A Overview

The appendix details the experimental setup and methods, including task definitions, model architectures, training targets, correction strategies, and mathematical formulations for all ODE and PDE systems. It also contains extended quantitative and visual results for both ODE and PDE experiments, complementing the main text with additional figures and analyses.

B Extended Setup and Methods

Tasks and dynamics. We study both ODEs and PDEs under a unified predict-correct framework.

- **ODEs:** (i) scalar decay $y' = -y$; (ii) Lorenz-63; (iii) Lorenz-96 with $N = 20$.
- **PDE:** 2D forced heat equation on $[0, 1]^2$ with homogeneous Dirichlet boundaries, $\partial_t u = \alpha(u_{xx} + u_{yy}) + S(x, y, t)$, with forcing $S(x, y, t) = \cos(\omega t) \sin(\pi x) \sin(\pi y)$.

Neural predictor. A compact CNN ingests the current field/state and scalar channels (time t, step Δt, spatial steps $\Delta x, \Delta y$, coefficients such as α or ω), and outputs a one-step forecast. Dirichlet zeros are enforced on the network output along the PDE boundary indices. For ODEs we use a small MLP or sequence of fully connected layers (implicitly represented in our scripts); for the PDE we use several 3×3 conv layers with Tanh activations and a final 1-channel head.

Training targets for the predictor. We compare two supervised one-step targets:

- **RK4 (mostly ODEs):** y_{n+1}^{RK4} from the exact RHS f; the loss is $\|\Phi_\theta(y_n, h) - y_{n+1}^{\mathrm{RK4}}\|^2$.
- **Trapezoid/CN-aligned:** targets that mirror the inference-time corrector. For ODEs we use a trapezoid with stop-gradient through the predicted $\hat{y}_{n+1}$. For the PDE we use Crank-Nicolson (CN):

$$(I - \tfrac{\Delta t\, \alpha}{2} L)\, u_{n+1} = (I + \tfrac{\Delta t\, \alpha}{2} L)\, u_n + \tfrac{\Delta t}{2}(S_n + S_{n+1}), \tag{1}$$

  with L the 5-point Laplacian and boundary rows set to Dirichlet zeros.

Inference-time correction. At rollout we report both: (i) NN-only (iterate the predictor), and (ii) predictor+corrector. For ODEs the corrector is trapezoid; for the PDE we solve the CN system every step using conjugate gradients (CG), warm-started by the neural output. We enforce Dirichlet zeros after each CG iterate.

Discretization and stability (PDE). We use second-order central differences with $N_x \times N_y$ grid, spacings $\Delta x, \Delta y$, and a fixed Δt. Although CN is A-stable, we follow conservative Δt choices for direct comparison to explicit baselines. All PDE boundaries are set to $u = 0$ at each step.

Dataset generation. For each ODE/PDE system we sample parameters (e.g., α, ω, horizon T, amplitude A_0 for the PDE initial condition) and generate fixed-step trajectories. For the PDE, we draw Δt from a range proportional to a stability bound $\propto 1/(\alpha(\Delta x^{-2} + \Delta y^{-2}))$, and simulate with CN for target generation.

Optimization. We train with Adam (initial lr 3×10^{-3}) and a ReduceLROnPlateau scheduler on the validation loss. Batch sizes of 16 (PDE) and comparable sizes for ODEs are used, with 5-10 epochs for the compact demonstrations. We seed the runs and save the best checkpoint by validation loss. All models are trained/evaluated on CPU or single-GPU depending on availability; PDE CN solves benefit from GPU for tensor ops, while CG remains inexpensive at the grid sizes considered.

Metrics. We report maximum absolute error over the grid/time for the PDE; for ODEs, we report max absolute error over time (scalar, Lorenz-63 dimensions, or over all $N = 20$ dimensions for Lorenz-96). We also visualize rollouts, error trajectories, and heatmaps where appropriate.

B.1 Mathematical formulation of the training targets

Intuition. Oracle RK4 supervision trains the network to approximate the continuous-time flow map directly. In contrast, the trapezoid/CN-aligned targets are defined by the same discrete implicit equation that the corrector enforces at inference. The training loss therefore measures how well the predictor satisfies this implicit update, i.e., the residual of the corresponding trapezoid or CN step. When the corrector is a convergent solver for this implicit equation, starting from a prediction with small residual means that only a small adjustment is required at inference, which helps explain the stable corrected rollouts observed in our experiments.

Notation. We write an ODE as $y' = f(t, y;\, \mu)$ (state $y \in \mathbb{R}^d$, optional parameters μ), with time grid $t_n = t_0 + nh$. The neural predictor (one-step map) is

$$\hat{y}_{n+1} = \Phi_\theta(t_n, y_n, h, \mu, \dots), \tag{2}$$

where θ are trainable weights. We use the operator $\mathrm{sg}(\cdot)$ to denote stop-gradient: in forward pass $\mathrm{sg}(z) = z$, while in backward pass $\partial\ \mathrm{sg}(z)/\partial z = 0$.

ODE: Oracle RK4 one-step supervision. Given (t_n, y_n, h) and parameters μ, define

$$k_1 = f(t_n, y_n;\, \mu), \tag{3}$$

$$k_2 = f\big(t_n + \tfrac{h}{2},\, y_n + \tfrac{h}{2}k_1;\, \mu\big), \tag{4}$$

$$k_3 = f\big(t_n + \tfrac{h}{2},\, y_n + \tfrac{h}{2}k_2;\, \mu\big), \tag{5}$$

$$k_4 = f\big(t_n + h,\, y_n + hk_3;\, \mu\big), \tag{6}$$

$$y_{n+1}^{\mathrm{RK4}} = y_n + \tfrac{h}{6}(k_1 + 2k_2 + 2k_3 + k_4). \tag{7}$$

The supervised loss is

$$\mathcal{L}_{\mathrm{RK4}}(\theta) = \big\|\hat{y}_{n+1} - y_{n+1}^{\mathrm{RK4}}\big\|_2^2. \tag{8}$$

Usage in code: simple decay ODE and Lorenz-63/L96 with parameter channels.

ODE: Trapezoid-aligned (PC-in-training) with stop-gradient. Form the target by a single trapezoid application whose second RHS is evaluated at the predicted next state with stopped gradients:

$$y_{n+1}^{\mathrm{trap}}(\theta) = y_n + \tfrac{h}{2}\Big(f(t_n, y_n;\, \mu) + f\big(t_{n+1},\, \mathrm{sg}(\hat{y}_{n+1});\, \mu\big)\Big). \tag{9}$$

The training loss is

$$\mathcal{L}_{\mathrm{trap}}(\theta) = \big\|\hat{y}_{n+1} - y_{n+1}^{\mathrm{trap}}(\theta)\big\|_2^2. \tag{10}$$

This mirrors the inference-time corrector while keeping supervision explicit in Φ_θ.

PDE model. We consider the linear parabolic PDE on $\Omega \subset \mathbb{R}^2$ with homogeneous Dirichlet data,

$$u_t = \alpha\, \Delta u + s(t, \boldsymbol{x}), \qquad u|_{\partial\Omega} = 0. \tag{11}$$

After spatial discretization (finite differences or finite elements), this becomes an ODE system $\dot{\boldsymbol{u}} = F(\boldsymbol{u}, t)$ of size M.

FD (5-point Laplacian) Crank-Nicolson target. Let $A_h \in \mathbb{R}^{M\times M}$ be the discrete Laplacian with Dirichlet rows enforcing boundary zeros, and let $\boldsymbol{b}_n$ be the sampled forcing at t_n. The semi-discrete ODE is $\dot{\boldsymbol{u}} = \alpha A_h \boldsymbol{u} + \boldsymbol{b}(t)$. One CN step with Δt reads

$$\big(I - \tfrac{\Delta t\,\alpha}{2}A_h\big)\boldsymbol{u}_{n+1}^{\mathrm{CN}} = \big(I + \tfrac{\Delta t\,\alpha}{2}A_h\big)\boldsymbol{u}_n + \tfrac{\Delta t}{2}\,(\boldsymbol{b}_{n+1} + \boldsymbol{b}_n). \tag{12}$$

Given the predictor output $\hat{\boldsymbol{u}}_{n+1} = \Phi_\theta(\boldsymbol{u}_n, \Delta t, \text{grid meta})$, the supervised CN loss is

$$\mathcal{L}_{\mathrm{CN,FD}}(\theta) = \big\|\hat{\boldsymbol{u}}_{n+1} - \boldsymbol{u}_{n+1}^{\mathrm{CN}}\big\|_2^2. \tag{13}$$

For a semilinear variation $\dot{\boldsymbol{u}} = \alpha A_h \boldsymbol{u} + \boldsymbol{r}(\boldsymbol{u}, t) + \boldsymbol{s}(t)$, the PC-in-training target is

$$\big(I - \tfrac{\Delta t\,\alpha}{2}A_h\big)\boldsymbol{u}_{n+1}^{\mathrm{tar}} = \big(I + \tfrac{\Delta t\,\alpha}{2}A_h\big)\boldsymbol{u}_n \tag{14}$$

$$+ \tfrac{\Delta t}{2}\Big(\boldsymbol{r}(\boldsymbol{u}_n, t_n) + \boldsymbol{r}(\mathrm{sg}(\hat{\boldsymbol{u}}_{n+1}), t_{n+1})\Big) + \tfrac{\Delta t}{2}\big(\boldsymbol{s}_{n+1} + \boldsymbol{s}_n\big), \tag{15}$$

$$\mathcal{L}_{\mathrm{PC,FD}}(\theta) = \big\|\hat{\boldsymbol{u}}_{n+1} - \boldsymbol{u}_{n+1}^{\mathrm{tar}}\big\|_2^2. \tag{16}$$

For our linear heat equation (no $\boldsymbol{r}$, $\boldsymbol{s}$ independent of $\boldsymbol{u}$), (14) reduces to (12).

FE (P1 Galerkin) Crank-Nicolson target. On a conforming triangulation with nodal basis $\{\varphi_i\}_{i=1}^{M}$, define the mass and stiffness matrices

$$M_{ij} = \int_\Omega \varphi_i \varphi_j \, \mathrm{d}x, \qquad K_{ij} = \int_\Omega \nabla\varphi_i \cdot \nabla\varphi_j \, \mathrm{d}x, \tag{17}$$

with appropriate Dirichlet treatment (rows/cols modified or condensed). The semi-discrete system is

$$M\dot{\boldsymbol{u}} + \alpha K \boldsymbol{u} = \boldsymbol{f}(t). \tag{18}$$

One CN step gives

$$\big(M + \tfrac{\Delta t\,\alpha}{2} K\big)\boldsymbol{u}_{n+1}^{\mathrm{CN}} = \big(M - \tfrac{\Delta t\,\alpha}{2} K\big)\boldsymbol{u}_n + \tfrac{\Delta t}{2}\big(\boldsymbol{f}_{n+1} + \boldsymbol{f}_n\big). \tag{19}$$

The supervised loss can be the Euclidean norm or the M-weighted norm $\|\boldsymbol{v}\|_M^2 = \boldsymbol{v}^\top M \boldsymbol{v}$:

$$\mathcal{L}_{\mathrm{CN,FE}}(\theta) = \big\|\hat{\boldsymbol{u}}_{n+1} - \boldsymbol{u}_{n+1}^{\mathrm{CN}}\big\|_2^2 \quad \text{or} \quad \|\hat{\boldsymbol{u}}_{n+1} - \boldsymbol{u}_{n+1}^{\mathrm{CN}}\|_M^2. \tag{20}$$

For a semilinear FE system $M\dot{\boldsymbol{u}} + \alpha K \boldsymbol{u} = \boldsymbol{g}(\boldsymbol{u}, t) + \boldsymbol{f}(t)$, the PC-in-training target reads

$$\big(M + \tfrac{\Delta t\,\alpha}{2} K\big)\boldsymbol{u}_{n+1}^{\mathrm{tar}} = \big(M - \tfrac{\Delta t\,\alpha}{2} K\big)\boldsymbol{u}_n \tag{21}$$

$$+ \tfrac{\Delta t}{2}\Big(\boldsymbol{g}(\boldsymbol{u}_n, t_n) + \boldsymbol{g}(\mathrm{sg}(\hat{\boldsymbol{u}}_{n+1}), t_{n+1})\Big) + \tfrac{\Delta t}{2}\big(\boldsymbol{f}_{n+1} + \boldsymbol{f}_n\big), \tag{22}$$

$$\mathcal{L}_{\mathrm{PC,FE}}(\theta) = \big\|\hat{\boldsymbol{u}}_{n+1} - \boldsymbol{u}_{n+1}^{\mathrm{tar}}\big\|_2^2 \text{ (or } M\text{-norm)}. \tag{23}$$

Inference-time correctors (for completeness). For ODEs we apply one trapezoid update initialized at $\hat{y}_{n+1}$:

$$y_{n+1}^{\mathrm{corr}} = y_n + \tfrac{h}{2}\Big(f(t_n, y_n;\, \mu) + f\big(t_{n+1}, \hat{y}_{n+1};\, \mu\big)\Big), \tag{24}$$

optionally iterating a few fixed-point or Newton steps when f is stiff/nonlinear. For PDEs we solve the linear CN system (12)/(19) each step, typically with CG or a sparse direct solver, warm-started at $\hat{\boldsymbol{u}}_{n+1}$.

Remarks. (i) The trapezoid/CN-aligned targets make the training loss state the implicit equation the corrector would solve at inference, improving train-test alignment. (ii) In our linear heat setting, the semilinear PC-in-training formulas reduce exactly to CN supervision since there is no nonlinearity in u on the RHS. (iii) For Lorenz/L96 we pass parameters μ as additional inputs to Φ_θ and use the same formulas above, with f the standard Lorenz/L96 RHS.

C Extended PDE Results (2D Heat, CN)

These supplementary figures extend the main results by providing additional visualizations and quantitative comparisons for each method discussed in the main text. This allows for a more detailed assessment and supports the claims made about error, drift, and the effectiveness of corrections. The following figures (see, e.g., Figures 4 and 5) provide detailed visualizations and quantitative results for the 2D heat equation experiments.

D Extended ODE Results

These extended ODE results complement the main text by providing additional quantitative and visual detail for all systems and correction schemes. These results support the generality and robustness of the claims made in the main text. The following figures and tables (see, e.g., Figures 6–17 and Table 1) present extended quantitative and visual results for all ODE systems examined.

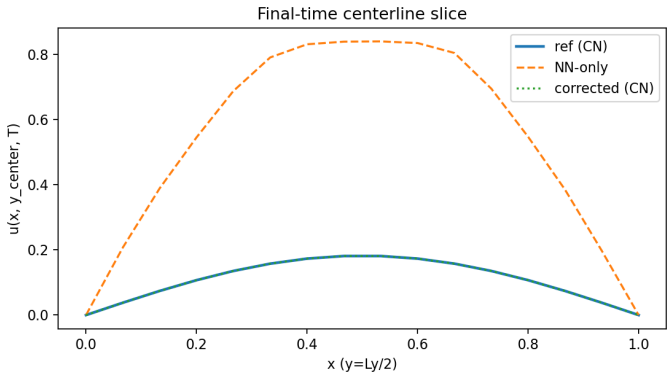


Figure 4: 2D heat (Dirichlet): final-time centerline slice. We plot $u(x, y_{\text{center}}, T)$ at the final time for the reference (CN), NN-only, and CN-corrected models. The CN-corrected solution overlays the reference, while NN-only drifts and overshoots, consistent with main-text findings.

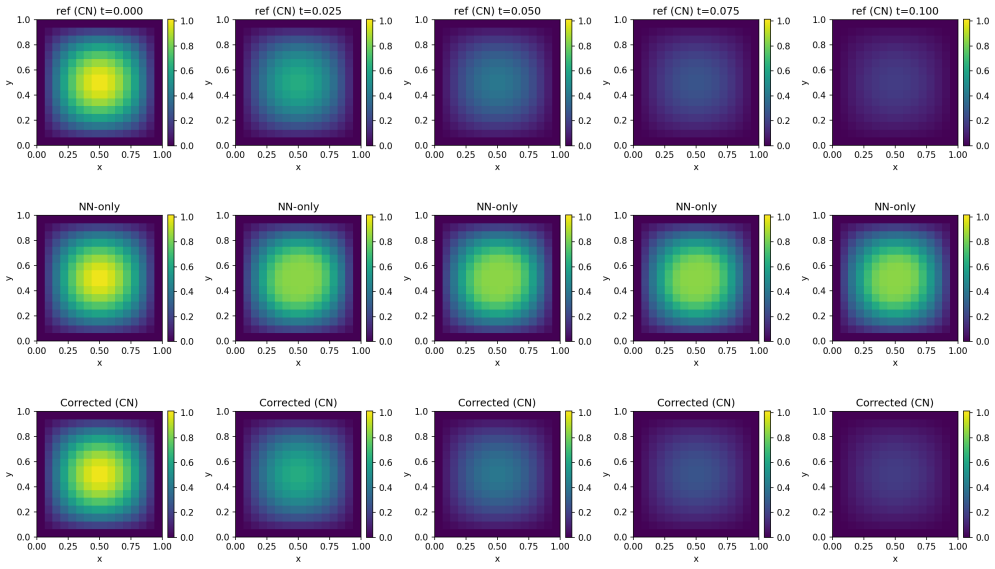
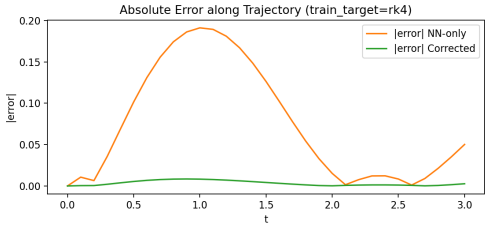


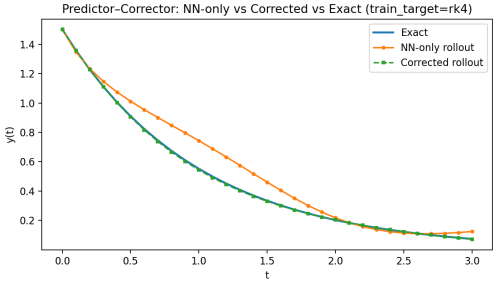
Figure 5: 2D heat (Dirichlet): time-snapshots (top: reference/CN; middle: NN-only; bottom: CN-corrected). CN-corrected predictions remain close to the reference while NN-only diverges, in line with the main text.

Table 1: Ablation study: maximum absolute error across ODE systems (lower is better). PC correction reduces maximum error for simple ODE and Lorenz-63. For Lorenz-96, rare spikes persist, as discussed in the main text.

System	Target	NN-only (max $\lvert\text{err}\rvert$)	PC corrected (max $\lvert\text{err}\rvert$)
Simple $y' = -y$	RK4	0.1911	0.0084
Simple $y' = -y$	Trapezoid	0.1860	0.0082
Lorenz-63	RK4	28.385	7.179
Lorenz-63	Trapezoid	31.923	11.887
Lorenz-96 (N=20)	RK4	23.487	25.442
Lorenz-96 (N=20)	Trapezoid	20.518	23.485



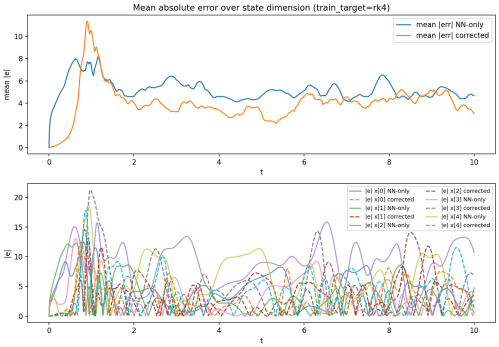
(a) Simple ODE: error over time (RK4 target). The predictor+corrector (PC) model maintains lower errors versus NN-only, consistent with reference-following in rollouts.



(b) Simple ODE: rollout comparison (RK4 target). The PC-corrected trajectory overlays the reference, whereas NN-only drifts, echoing error-curve findings.

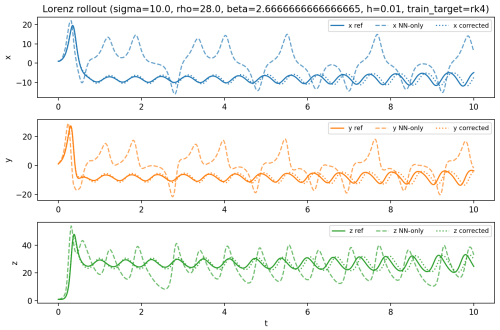
Figure 6: Simple ODE: RK4-target results. PC correction sharply reduces both one-step and rollout error compared to NN-only. Corrected rollouts overlay the reference, supporting the main text's claim about improved stability and accuracy.

(a) Lorenz-63: error over time (RK4 target). PC-corrected errors remain small, whereas NN-only exhibits intermittent spikes.

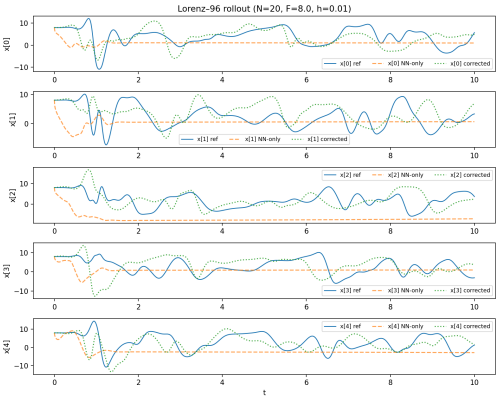


(b) Lorenz-96 ($N = 20$): maximum error (RK4 target), averaged over all dimensions. Corrections lower the error overall, especially after the initial transient.

Figure 7: Chaotic ODEs: Error comparisons under RK4 supervision. Across systems, PC correction reduces errors and suppresses outlier spikes versus NN-only, supporting claims in the main text.

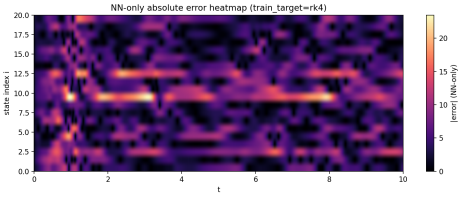


(a) Lorenz-63: rollout comparison. PC-corrected trajectory tracks the reference closely, while NN-only diverges, especially in chaotic intervals.

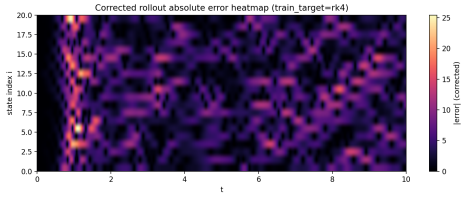


(b) Lorenz-96: rollout comparisons over selected dimensions. The PC-corrected model reduces drift relative to NN-only.

Figure 8: RK4: Rollout overlays for Lorenz-63 and Lorenz-96. PC-corrected rollouts track the reference, reducing visible drift compared to NN-only across all systems.



(a) Lorenz-96 (N=20), RK4: absolute error heatmap for NN-only. Pronounced error bands are visible.



(b) Lorenz-96 (N=20), RK4: absolute error heatmap for PC-corrected. Errors are substantially suppressed and localized.

Figure 9: Lorenz-96 (N=20), RK4: absolute error heatmaps. PC correction localizes and reduces errors across coordinates and time with respect to NN-only.

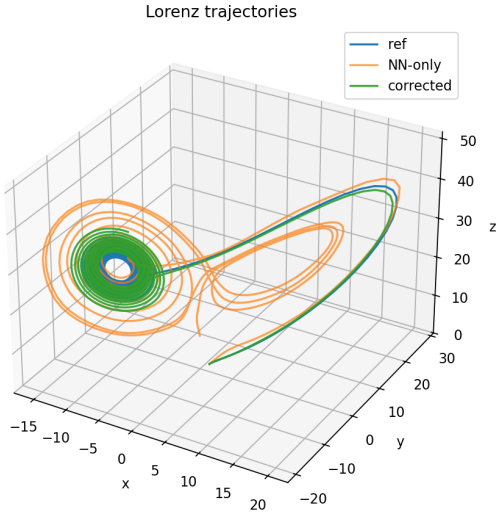
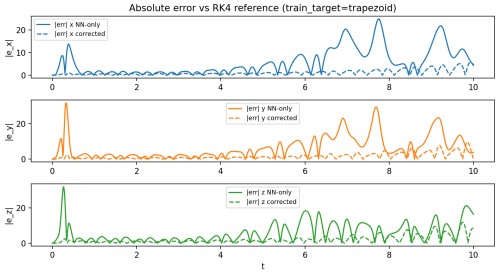
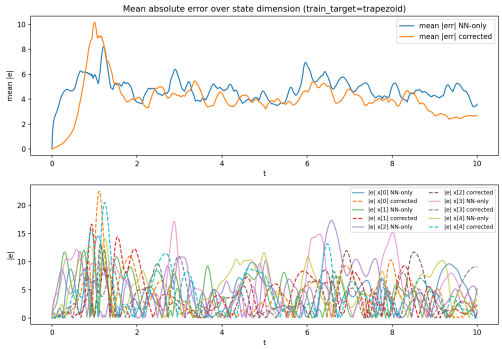


Figure 10: Lorenz-63, trapezoid target: 3D trajectory overlay. PC-corrected rollouts remain close to reference, as with RK4; NN-only diverges in chaotic phases.



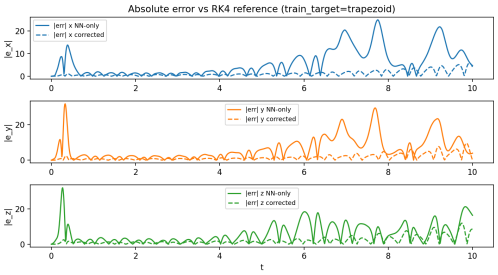
(a) Lorenz-63: error over time (trapezoid target). Correction reduces errors and sharp spikes compared to NN-only.



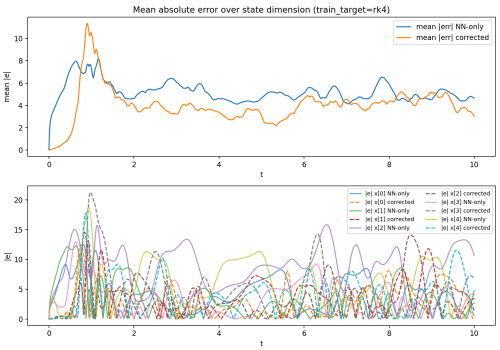
(b) Lorenz-96 ($N = 20$): maximum error (trapezoid target). PC correction lowers typical errors, with rare spikes in aggressive regimes.

Figure 11: Chaotic ODEs: Error reduction via correction (trapezoid target). Correction reduces both typical and peak errors across systems relative to NN-only, in agreement with RK4 findings.

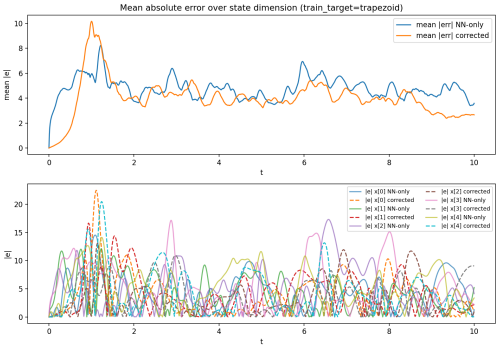
(a) Lorenz-63: error (RK4 target). PC correction yields much lower errors than NN-only.



(b) Lorenz-63: error (trapezoid target). Error reduction is pronounced with PC correction.

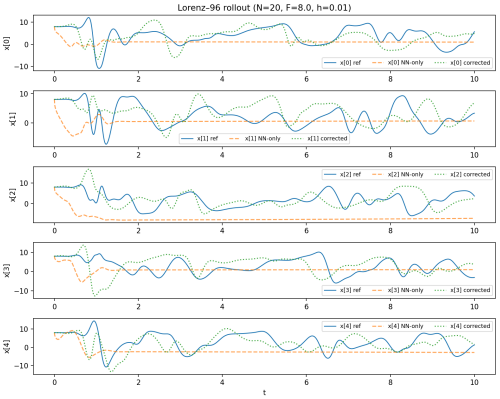


(c) Lorenz-96 (N=20): max error (RK4 target). Maximum error is suppressed by correction except during rare spikes.

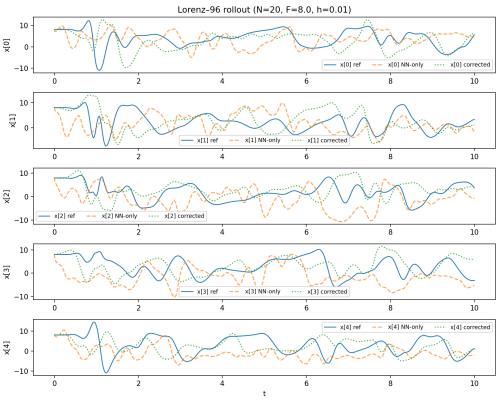


(d) Lorenz-96 (N=20): max error (trapezoid target). Correction lowers typical error levels.

Figure 12: Head-to-head error comparison for predictor targets (RK4 vs trapezoid). Correction is beneficial in all systems, most strongly on Lorenz-63 under RK4, and in typical errors for Lorenz-96.

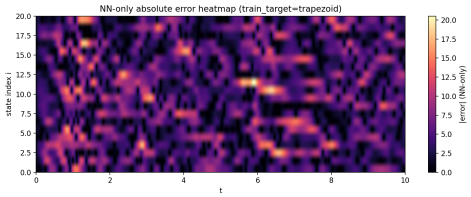


(a) Lorenz-96: rollout (RK4 target) comparison. Correction improves phase alignment and reduces drift.

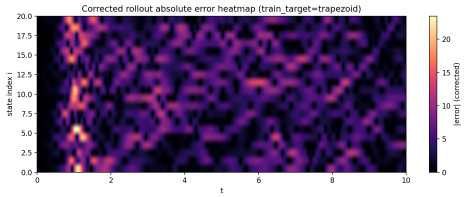


(b) Lorenz-96: rollout (trapezoid target) comparison. PC correction provides comparable improvements in drift reduction.

Figure 13: Lorenz-96: Head-to-head rollout comparisons for RK4 and trapezoid targets. Both benefit from PC correction, showing close tracking of the reference.

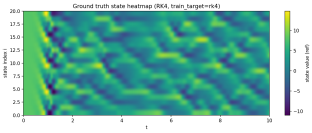


(a) Lorenz-96 (N=20), trapezoid: absolute error heatmap for NN-only. Coherent error bands are observed.

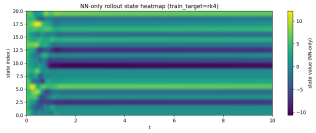


(b) Lorenz-96 (N=20), trapezoid: absolute error heatmap for PC-corrected. Correction reduces and compresses errors.

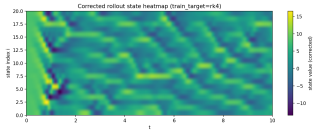
Figure 14: Lorenz-96 (N=20), trapezoid: absolute error heatmaps. Correction compresses error structures and suppresses peak values relative to NN-only.



(a) Lorenz-96: true state evolution (RK4 target).

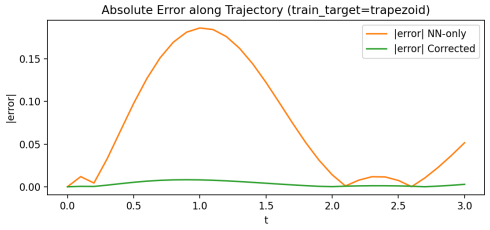


(b) Lorenz-96: NN-only prediction (RK4 target).

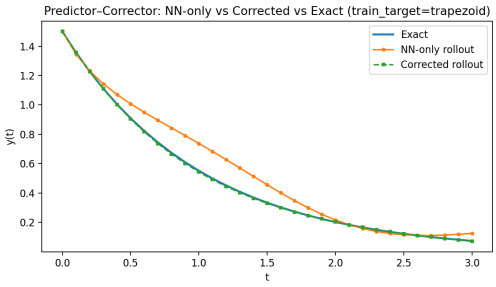


(c) Lorenz-96: PC-corrected prediction (RK4 target).

Figure 15: Lorenz-96 (N=20), RK4: state heatmaps. PC-corrected predictions follow ground truth more closely than NN-only, matching claims about error and phase lag reduction.

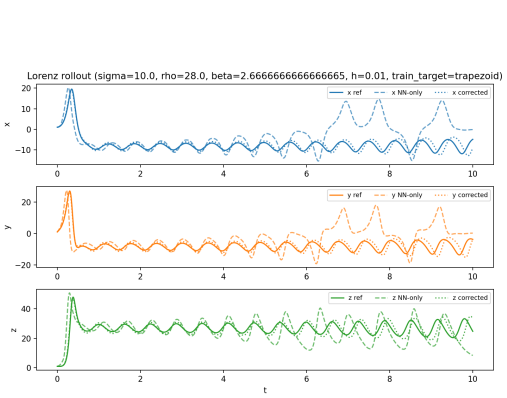


(a) Simple ODE: error over time (trapezoid target). PC correction reduces error to near-reference level, compared to larger NN-only drift.

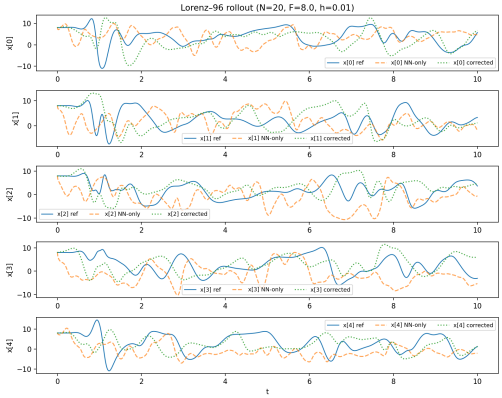


(b) Simple ODE: rollout comparison (trapezoid target). PC-corrected trajectories overlay the reference and leave less drift than NN-only.

Figure 16: Simple ODE (trapezoid target): Correction reduces error and secures close alignment to reference.



(a) Lorenz-63: rollout comparison (trapezoid target). Corrected trajectories reduce divergence versus NN-only.



(b) Lorenz-96: selected dimensions (trapezoid target). PC correction reduces drift in each coordinate, consistent with RK4 observations.

Figure 17: Rollout overlays (trapezoid target): corrected rollouts track reference and reduce both chaotic divergence and phase drift, paralleling RK4 outcomes.