

ON THE EFFECT OF NEURAL FIELD REPARAMETERIZATION FOR 4DVAR

JAEMIN OH*

Abstract. Four-dimensional variational data assimilation (4DVAR) is a cornerstone of numerical weather prediction, yet it remains computationally intensive and sensitive to initialization due to the non-convexity of its objective function. We propose a neural field-based reformulation of 4DVAR in which the spatiotemporal state is represented as a continuous function parameterized by a neural network. We demonstrate that optimizing in parameter space leverages the spectral bias of neural fields, acting as an implicit regularizer that stabilizes state estimation and suppresses spurious high-frequency oscillations without requiring explicit background error covariance information. Furthermore, by parameterizing the full spatiotemporal trajectory, our framework enables parallel-in-time optimization and incorporates physical constraints directly through physics-informed losses. Evaluations on chaotic benchmarks, including 2D Kolmogorov flow and 3D Taylor-Green vortices, show that neural reparameterization produces more accurate initial conditions than classical 4DVAR. When combined with separable neural architectures (SPINNs), the method achieves substantial speedups. Unlike many machine learning approaches, this framework requires no ground-truth training data, offering a robust and scalable alternative for operational data assimilation.

Key words. Data assimilation, 4D-Var, Neural fields, Spectral bias

MSC codes. 49M41, 65K10, 68T07

1. Introduction. Numerical weather prediction (NWP) has been a critical scientific and societal capability since Richardson’s early vision of rational forecasting [43]. Modern NWP systems integrate high-dimensional dynamical models with heterogeneous and incomplete observations to estimate the evolving atmospheric state. Because the governing equations are chaotic [40], even small errors in the initial condition can amplify rapidly, making accurate state estimation indispensable for reliable forecasts. Data assimilation addresses this challenge by systematically combining model dynamics with observational information. Among many existing approaches [19, 26, 38], the four-dimensional variational method (4DVAR) [35, 48] remains a cornerstone of operational NWP systems. Classical 4DVAR incorporates a background state, observational data, and a dynamical model to infer an optimal initial condition. Despite its success, however, 4DVAR faces limitations that constrain both accuracy and scalability when naively implemented, as we revisit in section 4.

From a mathematical perspective, 4DVAR is a partial differential equation (PDE)-constrained optimization problem. While the computational cost of repeatedly solving the governing PDE is a well-known bottleneck, an equally important challenge arises from nonlinearity. For nonlinear PDEs, the resulting objective function is highly non-convex and may admit multiple local minima with comparable cost values. Some of these solutions can satisfy the optimization objective yet remain physically implausible. Consequently, convergence behavior and solution quality depend critically on initialization strategies and regularization mechanisms.

Recent studies have reported an intriguing phenomenon in PDE-constrained optimization: when the solution is parameterized by a neural network, and optimization is performed in the parameter space, the resulting solutions tend to be simpler than those obtained by classical discretization-based approaches. In particular, Hoyer et al. [24] and Lu et al. [42] observed that neural parameterizations often favor smoother

*Division of Applied Mathematics, Brown University, Providence, RI (jaeminoh.math@gmail.com, <https://jaeminoh.github.io>).

and simpler solutions. This behavior is commonly attributed to the spectral bias of neural networks [49], whereby low-frequency components are learned preferentially, and high-frequency components are captured later during training. In this sense, spectral bias acts as an implicit regularizer that guides the optimization toward simpler solutions.

In this work, we investigate how such implicit regularization manifests in the context of 4DVAR. We study two benchmark problems – a two-dimensional Kolmogorov flow and a three-dimensional Taylor–Green vortex – to systematically compare classical formulations with neural reparameterization strategies. In the two-dimensional setting, we show that neural parameterization can recover accurate initial conditions without explicit background error covariance information. Building on the accuracy obtained through implicit regularization, we further consider enforcing physical constraints via physics-informed loss functions [28], thereby avoiding repeated time integration of the governing equations. When combined with efficient neural field architectures, this physics-informed approach yields competitive accuracy and substantial improvements in computational efficiency, particularly in the three-dimensional setting.

We close this section with a brief overview of related work. A growing body of literature explores machine learning-based approaches to data assimilation. Notable examples include learned inverse observation operators for 4DVAR [20] and diffusion-based methods [52, 25]. Ensemble-free neural filters have also been proposed [5, 47]. Several recent studies introduce latent-space data assimilation frameworks to reduce computational cost [37, 57]. However, these approaches typically assume access to ground-truth states or high-quality reanalysis datasets such as ERA5 [23], which limits direct comparison with our setting, where such information is unavailable. Bao et al. [3] proposed a related score-based filtering approach that relies solely on historical observations, whereas 4DVAR assimilates observations over a future time window. Du et al. [16] compared classical 4DVAR with physics-informed neural networks (PINNs); while closely related, their work does not address the regularizing effects of neural reparameterization, which constitute a central theme of this study.

The remainder of this paper is organized as follows. In section 2, we review the classical 4DVAR formulation, neural fields, and physics-informed neural networks. Section 3 describes the experimental setup for studying neural reparameterizations of 4DVAR. In section 4, we analyze the regularizing effects of neural parameterization. Parallel-in-time formulations, including weak-constraint 4DVAR and PINN-based methods, are examined in section 5. Section 6 presents comprehensive numerical results for the two-dimensional Kolmogorov flow at higher resolution and the three-dimensional Taylor–Green vortex. Finally, section 7 concludes the paper.

2. Background. This section introduces the notation, dynamical model, and classical 4DVAR framework used throughout the paper. We also review neural fields and physics-informed neural networks, which will serve as alternative representations of states and trajectories in later sections.

2.1. Dynamical model and observations. We consider a physical system governed by a time-dependent partial differential equation

$$(2.1a) \quad \frac{\partial u}{\partial t} = \mathcal{F}(u), \quad (t, x) \in (0, T) \times \Omega,$$

$$(2.1b) \quad u(0, x) = u_0(x), \quad x \in \Omega,$$

where $\Omega \subset \mathbb{R}^d$ is an open and bounded spatial domain, $u : [0, T] \times \Omega \rightarrow \mathbb{R}^{d_u}$ denotes the state variables (e.g., velocity or temperature fields), and \mathcal{F} is a (possibly nonlinear) differential operator encoding the system dynamics.

Observations are modeled as

$$(2.2) \quad y_k = H(u_k) + \varepsilon_k,$$

where H is an observation operator, $u_k = u(t_k, \cdot)$ is the state at time t_k , and ε_k represents measurement noise.

In practice, (2.1) is discretized in space and integrated forward in time to yield a discrete state-space model. When no ambiguity arises, we use the same notation u to denote both continuous and discretized states.

2.2. Four-dimensional variational assimilation. The classical 4DVAR estimates an optimal initial condition by minimizing the objective

$$(2.3) \quad J(u_0) = \|u_0 - u_b\|_B^2 + \sum_{k=1}^K \|H(u_k) - y_k\|_{R_k}^2,$$

where u_b is a background (prior) state, and B and R_k denote the background and observation error covariance matrices, respectively. Here, each u_k is obtained by numerically integrating (2.1a) forward in time from u_0 . For any symmetric positive definite matrix A , the Mahalanobis norm is defined as

$$\|u\|_A^2 = u^\top A^{-1} u.$$

The first term in (2.3) penalizes deviations from the background state, while the second term penalizes mismatches between the model trajectory and observations over the assimilation window. The minimizer u_0^* is typically computed using iterative gradient-based optimization, which requires repeated evaluations of the forward and adjoint models [18].

2.3. Neural fields. To explore alternative parameterizations of states and trajectories, we employ neural fields – coordinate-based neural networks that represent functions as continuous mappings

$$x \mapsto u^\theta(x),$$

where θ denotes the network parameters. Neural fields have been successfully applied in graphics and scientific computing to represent signals implicitly and at arbitrary resolution [54, 44, 36]. Their mesh-free nature and ability to evaluate solutions at arbitrary spatial locations make them particularly well suited for data assimilation problems with sparse, irregular, or multiscale observations.

2.4. Physics-informed neural networks. An alternative way to impose physical constraints is through physics-informed neural networks (PINNs). PINNs approximate the solution u to (2.1) by a neural field u^θ and determine the parameters θ by minimizing

$$L(\theta) = \int_{(0,T) \times \Omega} \left| \frac{\partial u^\theta(t,x)}{\partial t} - \mathcal{F}(u^\theta)(t,x) \right|^2 dt dx + \lambda \int_{\Omega} |u^\theta(0,x) - u_0(x)|^2 dx,$$

where $\lambda > 0$ balances the PDE residual and enforcement of the initial condition. In practice, the integrals are approximated using Monte Carlo sampling, and automatic

differentiation is employed to compute residuals and gradients. The PINN framework was originally proposed in [34] and later popularized by [50] as a general approach for solving PDE-constrained inverse problems.

In subsequent sections, we use neural fields to parameterize either (i) the initial condition or (ii) the full spatiotemporal state. This perspective allows us to examine how different representations affect the optimization landscape and computational properties of 4DVAR.

3. Experimental setup. This section describes the dynamical system, numerical discretization, and data generation procedure used throughout our experiments. Our goal is to investigate the effect of neural parameterization on the performance of 4DVAR using a well-established chaotic benchmark.

3.1. Kolmogorov flow. We consider the two-dimensional incompressible Navier-Stokes equations with Kolmogorov forcing [6, 10], a widely used testbed for studying chaotic and turbulent flows. Following the setup of [20], the governing equations in velocity form are

$$(3.1a) \quad \frac{\partial \mathbf{u}}{\partial t} + (\mathbf{u} \cdot \nabla) \mathbf{u} + \frac{1}{\rho} \nabla p = \nu \nabla^2 \mathbf{u} + \mathbf{f},$$

$$(3.1b) \quad \nabla \cdot \mathbf{u} = 0,$$

for $(t, x, y) \in (0, T) \times [0, 2\pi)^2$, where the density is $\rho = 1$, the viscosity is $\nu = 10^{-2}$, and periodic boundary conditions are imposed. The forcing is given by

$$(3.2) \quad \mathbf{f}(x, y) = -0.1 \mathbf{u} + \sin(4y) \hat{\mathbf{x}},$$

with $\hat{\mathbf{x}} = [1, 0]^\top$. The incompressibility constraint (3.1b) allows reformulation in terms of the vorticity $\omega = \nabla \times \mathbf{u}$. Taking the curl of (3.1a) yields

$$(3.3) \quad \frac{\partial \omega}{\partial t} + (\mathbf{u} \cdot \nabla) \omega = \nu \nabla^2 \omega - 0.1 \omega - 4 \cos(4y).$$

A detailed derivation is provided in Appendix G.

We solve (3.3) using a Fourier collocation method with a resolution 64^2 for spatial discretization and an implicit–explicit (IMEX) scheme for time integration with a step size 10^{-3} . Additional implementation details are given in Appendix B.1.

3.2. Benchmark initial condition. To construct a representative ground-truth initial state, we follow the protocol of [20]. Specifically, we sample a divergence-free random velocity field, apply spectral filtering with a peak wavenumber of 4, and scale the field so that the maximum velocity magnitude is approximately 7. The resulting field is then integrated forward to $t = 10$ using the vorticity solver described above. The velocity field at $t = 10$ is taken as the ground-truth initial condition for all experiments.

3.3. Observation model and sparsity. Synthetic observations are generated by integrating the system from the ground-truth initial condition up to time $t = 0.5$ and recording velocity snapshots at

$$t \in \{0, 0.05, \dots, 0.5\}.$$

Although the vorticity equation (3.3) is solved numerically, data assimilation is performed on the velocity field.

To control observation sparsity, we apply a uniform subsampling operator. For a sparsity level of k^2 , we retain indices $\{kn + 1 : n \in \mathbb{Z}_{\geq 0}\}$ along both spatial directions. This procedure yields regularly spaced pointwise observations that are consistent across experiments.

Figure 1 summarizes the data generation process. Panel (A) shows the ground-truth initial vorticity field, with yellow markers indicating the spatial locations at which observations are collected. Panel (B) displays the corresponding sparse observations obtained by applying the subsampling operator H . Panel (C) presents an interpolated reconstruction of the sparse observations, which we use as an initial guess for subsequent 4DVAR-based methods.

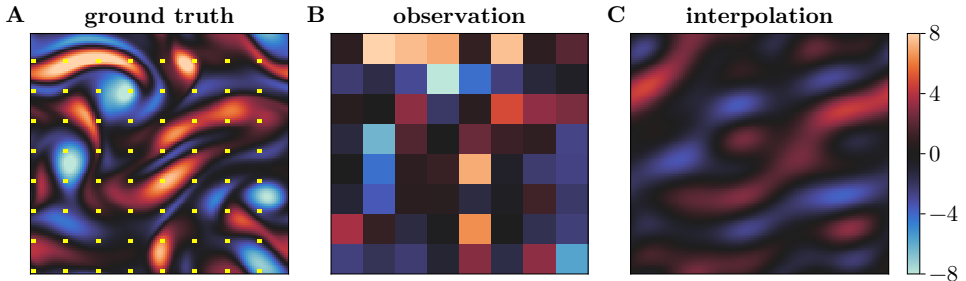


Fig. 1: Illustration of the data generation process: (A) ground-truth initial vorticity field with yellow markers indicating observation locations; (B) sparse observations produced by applying the subsampling operator; and (C) interpolated vorticity field reconstructed from the sparse measurements, used as an initial guess for 4DVAR.

3.4. Simplified 4DVAR objective. To isolate the effect of state representation and avoid interactions with prior information, we consider a simplified 4DVAR cost function

$$(3.4) \quad J_{\text{vanilla}}(u_0) = \sum_{k=0}^K \|H(u_k) - y_k\|^2,$$

which omits the background term $\|u_0 - u_b\|_B^2$ present in the classical formulation. In operational data assimilation, the background state u_b is typically a short-term forecast from the previous analysis cycle and plays an important regularizing role. Here, however, our objective is to compare different representations of the control variable (e.g., discretized states versus neural fields) under otherwise identical conditions. We therefore omit the background term and interpret (3.4) as a pure data-fitting objective constrained solely by the governing dynamics.

Throughout the remainder of this paper, we refer to the method that estimates the initial condition by minimizing (3.4) as vanilla-4DVAR.

4. Parameterizing the initial state with neural fields. Our first strategy represents the initial condition $u_0(x)$ by a neural field $u_0^\theta(x)$ with trainable parameters θ . Under this reparameterization, the simplified 4DVAR objective (3.4) becomes

$$(4.1) \quad J_{\text{neural}}(\theta) = J_{\text{vanilla}}(u_0^\theta),$$

and the optimization is performed directly in parameter space. We refer to this formulation as neural-4DVAR.

This change of variables replaces the high-dimensional state vector u_0 with the network parameters θ , which can substantially alter the geometry of the optimization landscape. Although no theoretical guarantees exist that parameter-space optimization is inherently easier, empirical evidence from related settings suggests that neural reparameterization can improve convergence robustness and reconstruction quality [24, 33]. The primary trade-off is a modest increase in per-iteration cost due to evaluating the neural field.

4.1. Effect of neural reparameterization. We first examine the impact of neural reparameterization by comparing the classical objective J_{vanilla} with its neural counterpart J_{neural} . While both objectives measure the same observation misfit, they are optimized over different spaces: J_{vanilla} operates on the discretized state u_0 , whereas J_{neural} operates on the network parameters θ .

The resulting behaviors are summarized in Figure 2. Panel (A) shows that both formulations successfully reduce the observation misfit during optimization. However, panel (B) reveals a crucial difference: vanilla-4DVAR reduces the misfit without improving the accuracy of the reconstructed initial condition, whereas neural-4DVAR consistently reduces both the misfit and the relative L^1 error.

To further understand this behavior, we examine the energy spectra of the estimated initial conditions. Panel (C) compares the spectra of the assimilated and ground-truth velocities. During vanilla-4DVAR, high-wavenumber energy grows rapidly, producing the small-scale artifacts visible in Figure 3. In contrast, neural-4DVAR maintains a stable spectrum throughout optimization. This behavior reflects the spectral bias of neural fields [49]: low-frequency components are learned preferentially, while high-frequency modes are incorporated more gradually. While spectral bias is frequently cited as a bottleneck in high-resolution reconstruction, it acts as a potent implicit regularizer within the data assimilation framework. By naturally attenuating spurious high-frequency modes, this bias stabilizes the optimization trajectory and prevents the model from fitting non-physical noise. A quantitative comparison between this neural inductive bias and explicit Fourier mode truncation is detailed in Appendix E.

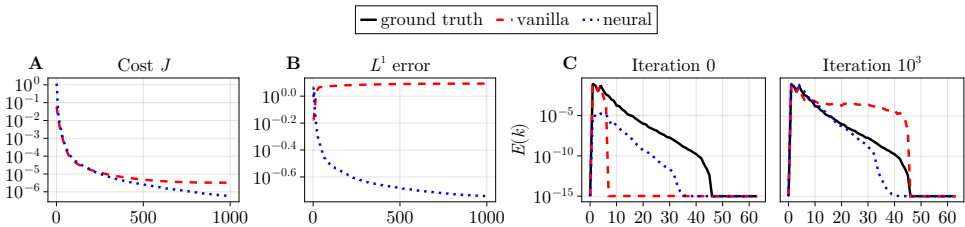


Fig. 2: Comparison of vanilla-4DVAR and neural-4DVAR. (A) Cost decay during optimization. (B) Relative L^1 error between assimilated and ground-truth initial conditions. (C) Energy spectra at iteration 0 and 10^3 .

4.2. Neural reparameterization and the control variable transform. The stabilizing effect of neural reparameterization resembles that of the classical control variable transform (CVT), which regularizes 4DVAR through the background error

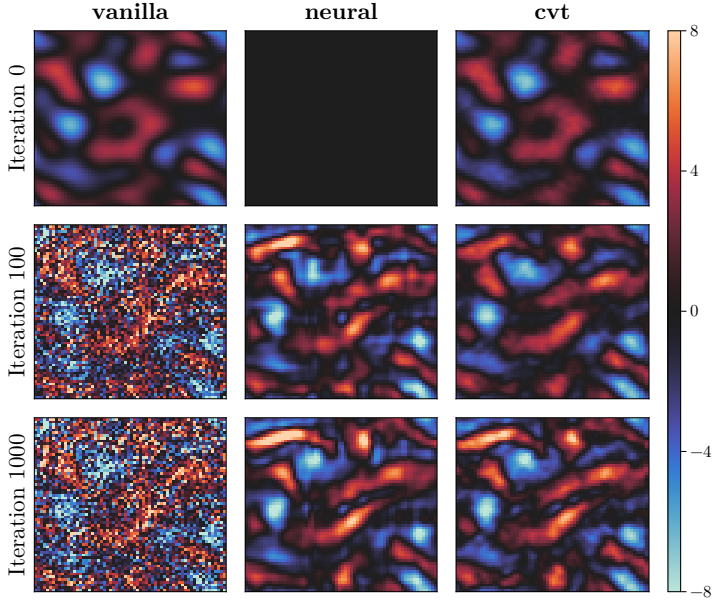


Fig. 3: Estimated initial conditions at iterations 0, 100, and 1000.

covariance matrix B [1]. CVT introduces the transformed variable

$$\zeta = B^{-1/2}u,$$

where $B^{-1/2} = \Lambda^{-1/2}Q^\top$ corresponds to the eigendecomposition $B = Q\Lambda Q^\top$. The coefficients ζ represent orthogonal modes scaled by $1/\sqrt{\lambda_i}$, yielding balanced variables with comparable magnitudes. Observation misfits are evaluated by reconstructing

$$u = B^{1/2}\zeta = Q\Lambda^{1/2}\zeta,$$

which leads to the transformed objective

$$J_{\text{CVT}}(\zeta) = J_{\text{vanilla}}(Q\Lambda^{1/2}\zeta).$$

To assess the analogy between CVT and neural reparameterization, we estimate the background error covariance B from N independently generated initial conditions following the protocol described in subsection 3.2. The empirical covariance is computed, and eigenvalues below 10^{-6} are truncated to ensure positive semidefiniteness.

As shown in Table 1, CVT substantially improves over vanilla-4DVAR, underscoring the importance of regularization in 4DVAR optimization. However, CVT performance depends sensitively on the quality of the covariance estimate. With only 10^2 ensemble members, the assimilation error remains large (6.83×10^{-1}), whereas increasing the ensemble to 10^3 yields a significant improvement (3.92×10^{-1}). Interestingly, further increasing the ensemble to 10^4 degrades performance (5.07×10^{-1}), highlighting the delicacy of empirical covariance estimation and the sensitivity of CVT to sampling variability.

In principle, sampling error decays at a rate $O(N^{-1/2})$, implying that accurate covariance estimation requires very large ensembles – often impractical when each

method	time	cost	error
vanilla	325	3.41E-5	1.94E0
CVT (10 ²)	324	5.22E-2	6.83E-1
CVT (10 ³)	309	1.03E-3	<u>3.92E-1</u>
CVT (10 ⁴)	310	9.55E-4	5.07E-1
neural	309	9.96E-5	2.73E-1

Table 1: Comparison of vanilla-4DVAR, CVT-4DVAR, and neural-4DVAR (time, cost, and final error). Numbers in parentheses denote ensemble size used to estimate B .

forward model evaluation is costly. Standard remedies such as covariance inflation and localization can mitigate sampling error but require substantial tuning [12]. Moreover, in operational settings the background error covariance B cannot be formed explicitly: for a discretized state $u \in \mathbb{R}^{2 \times 64 \times 64}$, the matrix $B \in \mathbb{R}^{8192 \times 8192}$ already requires approximately 512 MB of storage, while real-world NWP systems involve more than millions of degrees of freedom. As a result, practical systems rely on approximate covariance models rather than explicit matrices [2].

In this context, it is notable that neural-4DVAR achieves competitive – and in this experiment, superior – accuracy without access to any background error covariance information.¹

5. Parameterizing the full state with neural fields. While initial-condition parameterization improves stability, it does not leverage the full expressive power of neural fields. We therefore consider parameterizing the entire spatiotemporal field.

Our second strategy parameterizes the full spatiotemporal field $u(t, x)$ with a neural field $u^\theta(t, x)$ defined for all $(t, x) \in [0, T] \times \Omega$. This representation enables assimilation in function space rather than through discrete time stepping. To enforce physical constraint, we consider two complementary approaches: a weak-constraint 4DVAR formulation [46] and a physics-informed neural network formulation [50]; see subsection 5.1 and subsection 5.2.

5.1. Weak-constraint 4DVAR. The weak-constraint formulation extends the vanilla cost (3.4) by relaxing the assumption that the model (2.1a) is perfect [46]. The resulting objective takes the form

$$(5.1) \quad J_{\text{weak-vanilla}}(u_0, \dots, u_K) = \sum_{k=0}^K \|y_k - H(u_k)\|^2 + \underbrace{\sum_{k=0}^{K-1} \|u_{k+1} - \hat{u}_{k+1}\|^2}_{\text{physics constraint}},$$

where \hat{u}_{k+1} denotes the forecast obtained by advancing u_k through the numerical model. The second term penalizes deviations from model dynamics and thus incorporates physical constraints directly into the assimilation process.

As done in section 4, we also consider a neural reparameterization and control variable transform using background error covariance. First, we consider the neural reparameterization in which the entire spatiotemporal field is represented by a single

¹A systematic comparison with operational covariance models is an important direction for future work, but lies beyond the scope of this study.

neural field $u^\theta(t, x)$. Evaluating u^θ at discrete times and spatial grids yields the states $\{u_k^\theta\}_{k=0}^K$, leading to

$$J_{\text{weak-neural}}(\theta) = J_{\text{weak-vanilla}}(u_0^\theta, \dots, u_K^\theta).$$

Second, we also apply the CVT to balance scales across state variables. Let $\zeta_k = \Lambda^{-1/2} Q^\top u_k$, where $B = Q\Lambda Q^\top$ denotes the background error covariance. The weak-constraint cost in CVT coordinates becomes

$$J_{\text{weak-CVT}}(\zeta_0, \dots, \zeta_K) = J_{\text{weak-vanilla}}(Q\Lambda^{1/2}\zeta_0, \dots, Q\Lambda^{1/2}\zeta_K).$$

As shown in section 4, these approaches may regularize the optimization process, thereby enhancing the stability over the vanilla counterpart.

Although originally introduced to model uncertainty, weak-constraint 4DVAR admits a parallel-in-time structure – the dynamical consistency terms couple only adjacent time slices. This feature suggests the potential for substantial speedups in practice.

5.2. PINN-4DVAR. In contrast to weak-constraint 4DVAR, which enforces dynamics at discrete times, the PINN framework imposes the governing equations continuously over space–time. We consider the following physics-informed loss function

$$(5.2) \quad L_{\text{PINN}}(\theta) = \int_{(0,T) \times \Omega} \left(\frac{\partial \omega^\theta}{\partial t} + (\mathbf{u}^\theta \cdot \nabla) \omega^\theta - \nu \Delta \omega^\theta - \mathbf{f} \right)^2 dt d\mathbf{x}$$

$$(5.3) \quad + \lambda_{\text{divergence}} \int_{(0,T) \times \Omega} (\nabla \cdot \mathbf{u}^\theta)^2 dt d\mathbf{x},$$

where we directly parameterize the velocity field with a neural field and then compute the vorticity ω^θ via automatic differentiation.² Because the incompressibility constraint $\nabla \cdot \mathbf{u} = 0$ is not automatically enforced, we add a divergence-penalty term weighted by $\lambda_{\text{divergence}} = 5 \times 10^3$ following [11]. To enforce the periodic boundary condition, we adopt a positional encoding strategy [44]

$$\xi \mapsto (1, \cos(\xi), \sin(\xi), \dots, \cos(k\xi), \sin(k\xi))$$

for the spatial coordinate x and y , separately.

To assimilate observations, we augment this with an observation misfit:

$$(5.4) \quad J_{\text{PINN}}(\theta) = L_{\text{PINN}}(\theta) + \lambda_{\text{data}} \sum_{k=0}^K |H(u_k^\theta) - y_k|^2,$$

where λ_{data} balances physics and data fidelity. Minimizing (5.4) defines the PINN-4DVAR method. Appendix D.2 provides more details on the physics-informed formulation.

For completeness, Figure 4 summarizes the workflows of vanilla-, neural-, and PINN-4DVAR.

²Other parameterizations are possible. For example, [51] imposes incompressibility by construction; parameterizing vorticity would require solving a Poisson equation on a grid, and parameterizing the streamfunction introduces higher-order derivatives – both of which compromise the mesh-free nature of neural fields.

Remark 5.1. Despite their generality, PINNs are known to suffer from training stiffness [32], temporal causality violations [55], or slow convergence [56]. Interestingly, in the data assimilation setting, these issues are mitigated: observational constraints stabilize training and reduce causality violations, helping PINNs remain effective even for chaotic dynamics.

Separable physics-informed neural networks. A standard choice for u^θ is a multi-layer perceptron (MLP). However, MLP-based PINNs often become slow when high-order derivatives are required. To address this, we adopt separable physics-informed neural networks (SPINNs), which exploit low-rank tensor-product structure on regular grids [11]. For a function $u : \mathbb{R}^d \rightarrow \mathbb{R}$, SPINNs approximate

$$u_{\text{SPINN}}(x_1, \dots, x_d; \theta_1, \dots, \theta_d) = \sum_{r=1}^R \prod_{i=1}^d \text{MLP}_r(x_i; \theta_i),$$

where each $\text{MLP}_r : \mathbb{R} \rightarrow \mathbb{R}^R$ produces the r -th component. The separable structure reduces evaluation on tensor grids from $O(\prod_i N_i)$ operations to $O(\sum_i N_i)$, giving substantial speedups in high resolution. We refer the reader to Appendix D.1 for hyperparameters.

5.3. Parallel-in-time efficiency. Having introduced the full-state parameterization strategies – weak-neural, PINN (MLP), and PINN (SPINN) – we now compare their computational efficiency. Both weak-constraint and PINN-based formulations admit parallel-in-time optimization. Observation misfits can be evaluated independently across time levels, and the neural representation decouples optimization from end-to-end sequential time stepping. This stands in sharp contrast to classical strong-constraint 4DVAR, which requires serial forward and adjoint integrations over the entire assimilation window. Empirical runtime comparisons are reported in Table 2.

$$\begin{array}{ccc} \left\| H\left(\boxed{u_0} \xrightarrow{F^{(k)}} \boxed{\text{stack}(u_k)}\right) - \boxed{\text{stack}(y_k)} \right\| & \left\| H\left(\theta \mapsto \boxed{u_0} \xrightarrow{F^{(k)}} \boxed{\text{stack}(u_k)}\right) - \boxed{\text{stack}(y_k)} \right\| & \left\| H\left(\theta \mapsto \boxed{\text{stack}(u_k)}\right) - \boxed{\text{stack}(y_k)} \right\| + L_{\text{PINN}} \\ \text{(a) vanilla-4DVAR} & \text{(b) neural-4DVAR} & \text{(c) PINN-4DVAR} \end{array}$$

Fig. 4: Schematics of three 4DVAR cost functions. (a) vanilla-4DVAR estimates the initial condition by minimizing observation misfits. (b) neural-4DVAR reparameterizes the initial condition with a neural network. (c) PINN-4DVAR parameterizes the full spatiotemporal state $u^\theta(t, x)$ and enforces governing dynamics through a physics-informed loss L_{PINN} .

Overall, the weak-constraint variants (weak-vanilla, weak-CVT, and weak-neural) achieve approximately an $8\times$ speedup relative to the classical CVT baseline, primarily due to the parallel-in-time structure inherent in weak-constraint 4DVAR (5.1). The PINN formulation based on a standard MLP obtains a more modest $2\times$ speedup; however, its relatively high final cost suggests difficulty in simultaneously minimizing the physics-informed term L_{PINN} and the observation misfit. Additional optimization steps would likely improve accuracy, but at the expense of runtime. This limited speedup reflects the computational overhead associated with evaluating PDE residuals via automatic differentiation. By contrast, incorporating the SPINN architecture dramatically reduces the computational burden – the total runtime drops to only 4

method	iterations	time	cost	error
CVT (10^3)	10^3	309	1.03E-3	3.92E-1
weak-vanilla	10^3	35	1.07E-7	1.46E0
weak-CVT (10^3)	10^3	36	1.10E-5	5.65E-1
weak-neural	10^3	35	3.66E-5	3.31E-1
PINN (MLP)	10^3	148	1.12E-2	5.80E-1
PINN (SPINN)	10^3	4	1.77E-4	4.43E-1
PINN (SPINN)	10^4	32	1.62E-5	3.86E-1

Table 2: Comparison of full-state parameterization methods. The table reports wall-clock time (seconds), final objective value, and the relative L^1 error of the recovered initial condition. Weak-constraint formulations and PINN-based variants are substantially faster than classical CVT-based 4DVAR, with SPINN-4DVAR achieving the lowest runtime and competitive accuracy.

seconds (a $77\times$ speedup), while maintaining accuracy comparable to the other full-state parameterization methods. After increasing the total number of iterations to 10^4 , the error was decreased by 13%.

5.4. Sensitivity of weak-constraint formulations. A notable observation is that the performance of CVT degrades when transitioning from the classical to the weak-constraint formulation: the assimilation error increases from 3.92×10^{-1} to 5.65×10^{-1} . A plausible explanation lies in the treatment of the background error covariance. As discussed in [1, Section 3.5], the background error covariance should ideally evolve under the dynamical model to remain consistent with the evolving state.³

More precisely, if $u_k \sim P$ and $\mathcal{M} : u_k \mapsto u_{k+1}$ denotes the forward map, then

$$u_{k+1} \sim \mathcal{M}_*(P),$$

where \mathcal{M}_* denotes pushforward. For example, if $u_0 \sim N(u, B)$ and $\mathcal{M} = M$ is linear. Then $u_1 \sim N(Mu, MBM^\top)$. In our weak-constraint experiments, however, the same static covariance is reused across all sub-windows. This mismatch between the evolving state and a fixed background error covariance likely contributes to the observed degradation in performance.

By contrast, neural reparameterization does not rely on an explicit background covariance; instead, it leverages implicit regularization effects such as spectral bias. Consequently, one would expect reduced sensitivity to the weak-constraint setting. To isolate this effect from optimization artifacts, we conducted a controlled comparison using double precision arithmetic and the limited-memory Broyden–Fletcher–Goldfarb–Shanno (L-BFGS) optimizer [39]. The results, reported in Table 3, confirm this hypothesis. While CVT exhibits approximately a twofold increase in error under the weak formulation ($1.39 \times 10^{-1} \rightarrow 3.24 \times 10^{-1}$), neural reparameterization shows the opposite trend: the weak-neural variant achieves lower error than its classical counterpart ($2.20 \times 10^{-1} \rightarrow 1.81 \times 10^{-1}$). These results support the view that

³From a statistical learning perspective, the opposite trends in cost and error indicate the possibility of overfitting due to the increased number of free variables. We have tried to impose larger penalty weights on the physics constraint term of (5.1), yet there was no improvement in the accuracy.

method	time	cost	error
CVT (10^4)	531	1.25E-6	1.39E-1
weak-CVT (10^4)	60	4.62E-7	3.24E-1
neural	506	2.06E-5	2.20E-1
weak-neural	67	5.68E-7	1.81E-1

Table 3: Comparison of CVT and neural reparameterization under the classical and weak-constraint formulations, evaluated in double precision with the L-BFGS optimizer to minimize optimization error. While CVT doubles the assimilation error when moving to the weak formulation ($1.39 \times 10^{-1} \rightarrow 3.24 \times 10^{-1}$), the neural approach shows the opposite trend, achieving improved accuracy ($2.20 \times 10^{-1} \rightarrow 1.81 \times 10^{-1}$).

neural parameterizations are inherently more robust to the structural limitations of weak-constraint formulations.

5.5. Effect of the physics-informed loss. A natural question is whether the performance gains of PINN-4DVAR stem primarily from the zero-shot super-resolution capability of neural fields [53], rather than from the explicit enforcement of physical constraints. To disentangle these effects, we compare PINN-4DVAR with a regression-only baseline,

$$J_{\text{regression}}(\theta) = \sum_{k=0}^K \|H(u_k^\theta) - y_k\|^2,$$

which fits a neural field solely to the observations and omits the physics-informed loss.

Table 4 reports relative L^1 errors for both methods across varying levels of observation noise and sparsity. Under dense, noise-free observations, regression can achieve competitive accuracy, and in a small number of cases, it marginally outperforms PINN-4DVAR. However, as either observation noise or sparsity increases, the regression baseline deteriorates rapidly, often by more than an order of magnitude. In contrast, PINN-4DVAR maintains consistently lower errors across nearly all tested regimes.⁴

These results indicate that interpolation capability alone is insufficient for reliable data assimilation in underdetermined or noisy settings. Enforcing physical consistency through the physics-informed loss substantially improves robustness, stabilizes optimization, and yields accurate reconstructions even when observational information is limited.

6. Overall comparison. In this section, we provide a collective comparison between vanilla-4DVAR and its weak and neural variants,⁵ in order to investigate the accuracy-efficiency trade-off. We first compare the methods in the 2D Kolmogorov with an increased spatial resolution (256^2) to explore how sparsity and noise levels of observations impact accuracy. Finally, we consider a 3D Taylor–Green vortex, a computationally demanding benchmark to evaluate computational efficiency and

⁴For $k^2 = 4^2$ and $\sigma \in \{0.2, 0.25\}$, the PINN-based initial condition estimates still exhibit better accuracy in vorticity gradients than the regression baseline ($0.9 \sim 1.3$ for regression and $0.83 \sim 0.96$ for PINN), owing to the physics-informed loss.

⁵We do not consider J_{CVT} here, since the full background error covariance matrices require more than 17GB.

	Sparsity k^2			
	2^2	4^2	8^2	
σ	0%	(8.73E-2 / 6.97E-2)	(2.98E-1 / 1.88E-1)	(7.96E-1 / 3.86E-1)
	5%	(5.06E-1 / 2.01E-1)	(3.32E-1 / 2.68E-1)	(8.00E-1 / 3.97E-1)
	10%	(7.01E-1 / 2.86E-1)	(3.75E-1 / 3.13E-1)	(8.04E-1 / 6.64E-1)
	15%	(9.41E-1 / 5.54E-1)	(4.20E-1 / 3.55E-1)	(8.08E-1 / 4.51E-1)
	20%	(1.17E0 / 6.30E-1)	(4.84E-1 / 6.41E-1)	(8.55E-1 / 7.11E-1)
	25%	(1.37E0 / 9.13E-1)	(5.51E-1 / 6.91E-1)	(8.39E-1 / 5.81E-1)

Table 4: Ablation study evaluating the effect of the physics-informed loss. Rows correspond to noise levels and columns to sparsity levels. Reported values are relative L^1 error for (regression / PINN-4DVAR). While regression without physics constraints performs well under low noise and dense observations, PINN-4DVAR outperforms it as noise and sparsity increase, except for the two cases.

scalability of the methods. To reduce the impact of rounding-off error, we use double precision. We use SPINN as the neural field to parameterize either the initial condition or the entire state. Further hyperparameters are provided in Appendix D.

6.1. 2D Kolmogorov flow on a higher resolution. We consider the same governing equation as (3.1) and (3.3), but employ 256^2 grid points for spatial discretization.

Accuracy test with varying sparsity. We first investigate the effect of observation sparsity on the accuracy of the estimated initial states. The sparsity level is varied from 2^2 to 32^2 , corresponding to approximately 25% down to 0.1% of the full state being observed. Table 5 reports the relative L^1 errors between the ground-truth initial vorticity and the estimates produced by different methods. As an additional baseline, we include “interp,” which applies either bicubic or Fourier interpolation to the observations at $t = 0$. As expected, reconstruction accuracy degrades as observations become sparser; however, the rate of degradation varies markedly across methods. Nevertheless, the neural-parameterized variants consistently achieve lower errors than both interpolation and vanilla-4DVAR baselines when observations are sparse, demonstrating enhanced robustness to limited observational coverage.

Accuracy test with varying noise levels. To evaluate robustness against noise in observations, we compare the methods under additive Gaussian noise $N(0, \sigma^2 I)$. Noise is injected after applying the subsampling operator with sparsity level 32^2 .

Table 6 presents relative L^1 errors between estimated initial conditions and the true initial condition, with noisy observation data. Notably, PINN-4DVAR maintains the lowest reconstruction error across all tested noise levels. In particular, adding 5% noise to observations significantly deteriorates initial estimates from all methods except for Fourier interpolation, weak-neural, and PINN.

Rollout test. Recall that observations were assimilated over the time interval $[0, 0.5]$.⁶ In many applications, however, the central objective is the quality of fore-

⁶We also investigated the sensitivity of the methods to the assimilation window length. Extending the window from $t = 0.5$ to $t = 1$ (with a proportional increase in observations) maintained the superior performance of the neural method, though PINN-based variants showed slight performance degradation, likely due to representational capacity limits over longer temporal horizons.

init	method	Sparsity k^2			
		4^2 (6.25%)	8^2 (1.56%)	16^2 (0.39%)	32^2 (0.1%)
bicubic	interp	5.31E-2	2.70E-1	8.44E-1	1.18E0
	vanilla	6.74E-2	4.43E-1	2.09E0	2.67E0
	weak-vanilla	5.46E-2	5.78E-1	2.80E0	2.46E0
Fourier	interp	1.36E-3	3.47E-2	2.02E-1	6.07E-1
	vanilla	1.80E-2	1.01E-1	4.17E-1	1.58E0
	weak-vanilla	4.55E-4	3.00E-2	4.59E-1	1.31E0
N/A	neural	2.52E-2	7.80E-2	1.56E-1	2.66E-1
	weak-neural	1.91E-2	4.08E-2	9.63E-2	3.30E-1
	PINN	8.54E-2	8.74E-2	1.56E-1	3.64E-1

Table 5: Relative L^1 errors between estimated and true initial vorticities under varying levels of observation sparsity. The *init* column denotes interpolation schemes for the initial guess. As expected, the accuracy of the three baselines degrades as observations become sparser. Neural and weak-neural methods maintain better performance overall. PINN-4DVAR achieves competitive accuracy at high sparsity.

init	method	Noise (σ)				
		5%	10%	15%	20%	25%
bicubic	interp	1.18E0	1.17E0	1.18E0	1.19E0	1.20E0
	vanilla	3.93E0	5.70E0	7.49E0	9.24E0	1.09E1
	weak-vanilla	2.64E0	2.89E0	3.16E0	3.41E0	3.65E0
Fourier	interp	6.71E-1	6.75E-1	6.82E-1	6.92E-1	7.06E-1
	vanilla	3.45E0	5.68E0	7.74E0	9.62E0	1.13E1
	weak-vanilla	1.59E0	2.00E0	2.32E0	2.61E0	2.85E0
N/A	neural	1.91E0	5.32E0	1.05E1	5.32E0	1.64E1
	weak-neural	4.45E-1	6.84E-1	9.33E-1	1.11E0	1.31E0
	PINN	3.82E-1	4.08E-1	4.44E-1	4.74E-1	4.97E-1

Table 6: Relative L^1 errors between estimated and true initial vorticities under varying levels of observation noise, and at a sparsity level 32^2 . The *init* column denotes interpolation schemes for the initial guess. PINN-4DVAR consistently achieves the lowest error across all noise levels, indicating robustness of the physics-informed formulation.

casts generated from the assimilated initial condition. To assess this, we roll out the estimated initial conditions with the numerical solver up to $T = 5$, which is $10\times$ of the assimilation horizon.

Figure 5 shows the resulting forecasts. The PINN-4DVAR solution preserves key vorticity features of the ground-truth state up to $t = 2.5$, in contrast to the vanilla method. Remarkably, even a simple neural reparameterization of the initial condition – without the full physics-informed framework – achieves the best overall accuracy, underscoring the value of neural parameterization for improving forecast accuracy.

Figure 6 illustrates forecast accuracy through the evolution of relative L^1 errors over time. The initial error decay reflects the dissipative nature of the benchmark

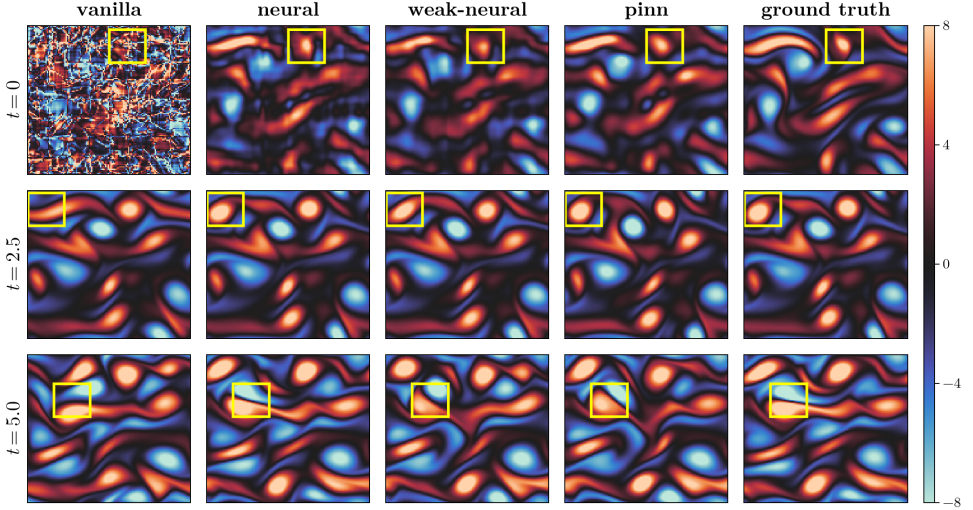


Fig. 5: Rollout test results from assimilated initial conditions. Rows correspond to time snapshots at $t \in \{0, 2.5, 5\}$. The first four columns show forecasts obtained using different assimilation methods, and the last column shows the ground-truth states. The vanilla-4DVAR initialization exhibits a spurious high-frequency perturbation that decays quickly during the forecast, whereas neural reparameterized variants produce smoother and more stable rollouts. This phenomenon is further examined in the energy spectrum analysis of Subsection 4.1.

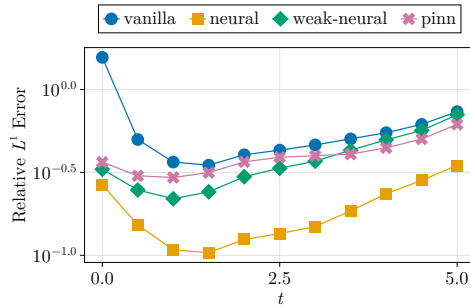


Fig. 6: Relative L^1 errors during the rollout test. All solver-dependent methods' errors decay rapidly once forecasting begins, yielding a more accurate long-term prediction at the final time. However, such behavior is less prominent in PINN. This reflects the alignment between 4DVAR and the numerical solver used for forecasting.

problem, where drag and viscosity rapidly damp small perturbations [20]. While the weak-neural method and neural-4DVAR both achieve lower initial assimilation errors than PINN-4DVAR ($3.30\text{E-}1$ and $2.66\text{E-}1$, respectively, vs. $3.64\text{E-}1$), neural-4DVAR demonstrates superior accuracy over longer horizons. This is because neural-4DVAR is optimized using the same numerical solver employed for forecasting, providing a

Method	iterations	time (s)	memory (MB)
vanilla (w/o checkpointing)	10^3	351	17102
vanilla (w/ checkpointing)	10^3	441	1290
weak-vanilla (w/ checkpointing)	10^3	106	1666
neural (w/ checkpointing)	10^3	429	1282
weak-neural (w/ checkpointing)	10^3	107	1402
PINN	10^4	48	2294

Table 7: Wall-clock time and memory usage for optimization. Results highlight the comparable memory requirements and the significant runtime speedup of PINN-4DVAR enabled by its parallel-in-time formulation.

structural advantage during rollout. In contrast, PINN-4DVAR enforces governing dynamics via a physics-informed loss; though this may result in slightly higher short-term errors, it may offer greater robustness when the forecasting model deviates from the assimilation model or when observations are sparse.

Computational efficiency. Table 7 reports wall-clock time and memory usage for 4DVAR methods under the comparison. Despite parameterizing the full spatiotemporal state, the memory requirements of PINN-4DVAR remain comparable to those of vanilla-4DVAR. Further reductions may be possible using neural compression techniques [17].

The most notable difference lies in the runtime. vanilla- and neural-4DVAR require sequential forward and backward passes through time, creating a strong bottleneck that limits scalability, especially on modern parallel hardware. By contrast, the parallel-in-time formulation of weak-constraint formulations yields approximately $3\times$ to $4\times$ speedup in overall runtime. Moreover, PINN-4DVAR yields an approximately $8\times$ speedup in overall runtime. While the advantage of SPINN may diminish for irregular observation grids, enforcing physical constraints via $L_{\text{PINN}}(\theta)$ remains computationally inexpensive so long as the domain retains a tensor-product structure.

6.2. 3D Taylor–Green vortex. We consider the incompressible Navier–Stokes equation (3.1a) with $\mathbf{f} = 0$ defined on a three-dimensional periodic domain $[0, 2\pi)^3$. Appendix B.2 provides details for the numerical solver.

Following [45, section 4.2], we consider $\rho = 1$, $\nu = 1/1600$, and

$$\mathbf{u}_0(x, y, z) = (\sin(x) \cos(y) \cos(z), -\cos(x) \sin(y) \cos(z), 0)$$

for the initial condition.

For J_{PINN} , we consider the vorticity form

$$(6.1) \quad \frac{\partial \omega}{\partial t} + (\mathbf{u} \cdot \nabla) \omega = (\omega \cdot \nabla) \mathbf{u} + \nu \Delta \omega.$$

A detailed derivation is provided in Appendix G.

We considered 64^3 Fourier grids for spatial discretization and 8^3 for observations. However, initialization with Fourier interpolation reconstructs the initial condition exactly; we instead consider initialization with a constant vector $(0.5, \dots, 0.5)$ for J_{vanilla} .

Results. Table 8 summarizes the results for the three-dimensional Taylor–Green vortex test. As in the two-dimensional case, minimizing J_{vanilla} and $J_{\text{weak-vanilla}}$ fails

to recover an accurate initial condition, whereas methods based on neural parameterization successfully approximate the ground truth.

Notably, the computational speedup achieved by transitioning from the strong-constraint objective J_{vanilla} to the weak-constraint formulation $J_{\text{weak-vanilla}}$ is less than $2\times$, in contrast to the $> 4\times$ speedup observed in the two-dimensional experiments. This reduced gain can be attributed to the substantially higher cost of spatial operators in three dimensions, which dominate the overall runtime and limit the effectiveness of parallel-in-time optimization. In this regime, alleviating temporal dependencies provides only a marginal benefit once the computational bottleneck shifts to spatial operators. In contrast, the PINN-based approach achieves comparable reconstruction accuracy in only 9 seconds. This highlights the advantage of full spatiotemporal neural parameterization, which bypasses explicit time stepping and expensive spatial discretization, yielding more favorable scaling behavior in three-dimensional settings.

method	cost	error	time (s)	memory (MB)
vanilla	1.48E-2	3.07E0	381	17284
weak-vanilla	6.12E-2	3.04E0	210	18070
neural	3.82E-4	1.39E-1	379	17300
weak-neural	7.54E-4	1.86E-1	211	17560
PINN	2.90E-3	2.68E-1	10	3722
PINN (10^4)	1.64E-4	7.66E-2	102	3722

Table 8: Test results for the 3D Taylor–Green vortex.

7. Conclusion. In this work, we introduced a neural field-based parameterization for 4DVAR and demonstrated its effectiveness on both two- and three-dimensional Navier–Stokes equations. Our results indicate that neural parameterizations act as an implicit regularization of the optimization, achieving performance competitive with classical control variable transforms based on background error covariance matrices. We also explored an alternative strategy for imposing physical constraints through physics-informed formulations. Taken together, these results suggest that neural parameterizations have the potential to improve both the accuracy and computational efficiency of classical 4DVAR formulations. More broadly, neural parameterizations of 4DVAR provide a promising foundation for hybrid approaches that combine the theoretical rigor of variational data assimilation with the flexibility of modern machine learning, particularly in the context of operational numerical weather prediction.

Several important research directions remain open. Extending the proposed framework to more realistic and operationally relevant settings – including the assimilation of satellite observations, globe-like geometries, explicit treatment of model uncertainty, and coupling with oceanic or land-surface models – will be essential for practical impact. In addition, recent advances in neural field representations and physics-informed neural networks, such as compact parameterizations [29, 27], offer promising opportunities to further enhance both the scalability and performance of neural parameterizations of 4DVAR.

Appendix A. Computing environment and software. All numerical experiments were performed on a single NVIDIA H200 graphics processing unit paired with an Intel Xeon Platinum 8480C central processing unit.

All implementations were carried out using JAX [7], which provides automatic differentiation and just-in-time compilation while adhering to a NumPy-compatible API [22]. Optimization routines were implemented using `optax` [15] and `jaxopt` [4]. For the two-dimensional Kolmogorov flow experiments, we employed the JAX-CFD package [31], which provides differentiable spectral solvers for incompressible flows. For the neural field parameterization, we used the original SPINN implementation [11]. All visualizations were generated using `Makie.jl` [14].

The code for numerical experiments is available at <https://github.com/jaeminoh/neural-4dvar>.

Appendix B. Numerical solvers. This appendix describes the numerical solvers used throughout this work.

B.1. Two-dimensional Kolmogorov flow. We employed a JAX-based pseudo-spectral solver implemented in the JAX-CFD package [31]. Spatial discretization was performed using the Fourier collocation method on a uniform grid with resolutions $N_x^2 \in \{64^2, 256^2\}$. All spatial derivatives were computed in Fourier space.

The governing equations were solved in vorticity form (3.3). The velocity field was recovered from the vorticity by solving the Poisson equation

$$-\Delta\psi = \omega,$$

with the velocity obtained as $\mathbf{u} = (\partial_y\psi, -\partial_x\psi)$, following [58]. This inversion is straightforward in Fourier space, where the Laplacian operator is diagonal.

Time integration was carried out using a low-storage Runge–Kutta scheme of order (5,4) [9], which employs five stages to achieve fourth-order accuracy. Linear terms were treated implicitly using a Crank–Nicolson scheme [8, Appendix D.3]. The time step size was constrained by the Courant–Friedrichs–Lewy (CFL) condition [13],

$$\Delta t \leq \frac{C \Delta x}{v_{\max}},$$

where $v_{\max} = 7$, $C = 0.5$, and $\Delta x = 2\pi/N_x$. This yields upper bounds of $\Delta t \leq 7 \times 10^{-3}$ for $N_x = 64$ and $\Delta t \leq 1.75 \times 10^{-3}$ for $N_x = 256$. Since the diffusion term is treated implicitly, this stability constraint is independent of the viscosity ν .

The convergence behavior of the numerical solver is summarized in Figure 7. Spatial convergence was assessed at a fixed time step $\Delta t = 10^{-4}$, with grid spacings $\Delta x \in \{2\pi/2^8, \dots, 2\pi/2^4\}$. The results exhibit exponential convergence in space. Temporal convergence was evaluated at a fixed spatial resolution $\Delta x = 2\pi/2^8$, using time steps $\Delta t \in \{10^{-4}, 2 \times 10^{-4}, \dots, 2^4 \times 10^{-3}\}$. The observed algebraic convergence rate agrees with the formal order of the time-stepping scheme.

B.2. Three-dimensional Taylor–Green vortex. For the three-dimensional Taylor–Green vortex considered in subsection 6.2, we implemented a differentiable pseudo-spectral solver in JAX following the approach described in [45]. The solver directly advances the velocity–pressure formulation (3.1) using the Fourier collocation method on a uniform grid with resolution $N_x^3 = 64^3$.

Time integration was performed using the classical fourth-order Runge–Kutta method with a fixed time step $\Delta t = 10^{-3}$.

Appendix C. Assimilation tests. Throughout this work, we did not observe successful reconstructions using the vanilla 4DVAR formulation under the benchmark settings considered in the main text. To validate the assimilation setup in a simplified

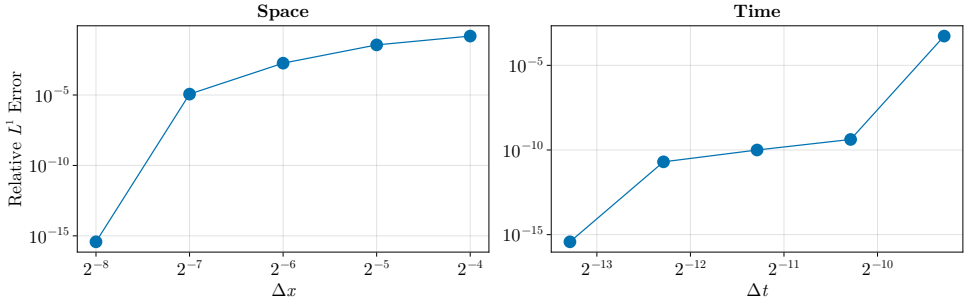


Fig. 7: Convergence tests of the numerical solver. Log-log plots of relative L^1 error versus grid spacing. The left panel shows exponential spatial convergence at a fixed time step size $\Delta t = 10^{-4}$. The right panel shows algebraic temporal convergence at the fixed spatial resolution $\Delta x = 2\pi/2^8$.

regime, we therefore examined the two-dimensional Kolmogorov flow at resolution $N_x^2 = 256^2$, as in subsection 6.1, but with a shorter assimilation window $t \in [0, 10^{-2}]$ and denser observations corresponding to a sparsity level of 2^2 .

The initial condition was obtained via bicubic interpolation. Under these conditions, optimization using the L-BFGS algorithm successfully converged to the true initial state, achieving a relative reconstruction error of 2.64×10^{-5} (see Figure 8). This result confirms the correctness of the numerical and assimilation setup in a regime with sufficient observational coverage and a short assimilation horizon.

As the observation sparsity increased beyond 2^2 , the optimization increasingly became trapped in local minima, resulting in reconstruction errors on the order of 10^{-2} or larger. This behavior highlights the sensitivity of the vanilla-4DVAR formulation to observation density and further motivates the use of classical control variable transforms or neural parameterizations explored in the main text.

Appendix D. Optimization and training details. All optimization problems were solved using the Adam optimizer [30]. A cosine decay learning-rate schedule [41] was employed throughout. For the vanilla- and neural-4DVAR formulations, the initial learning rate was set to 10^{-2} , while a smaller initial learning rate of 10^{-3} was used for PINN-4DVAR to improve training stability.

D.1. Neural network hyperparameters. Unless otherwise stated, all neural networks were instantiated with a width of 64 and a depth of 3. For the SPINN architecture, we used a separation rank $R = 128$ and five Fourier features. Network parameters were initialized using the Glorot initialization scheme [21]. The precise architectural details and implementation follow the original SPINN formulation and are available in the accompanying open-source repository [11].

D.2. Physics-informed formulation. For the physics-informed loss, spatiotemporal collocation points were resampled at each optimization iteration. Time coordinates were drawn from the uniform distribution $\text{Uniform}[0, 0.5]$, while spatial coordinates were sampled from $\text{Uniform}[0, 2\pi]$ in each spatial dimension.

For the two-dimensional Kolmogorov flow, we used $N_t = N_x = N_y = 128$ collocation points. For the three-dimensional incompressible Navier–Stokes equations, we used $N_t = N_x = N_y = N_z = 32$ collocation points. The data fidelity weight in the

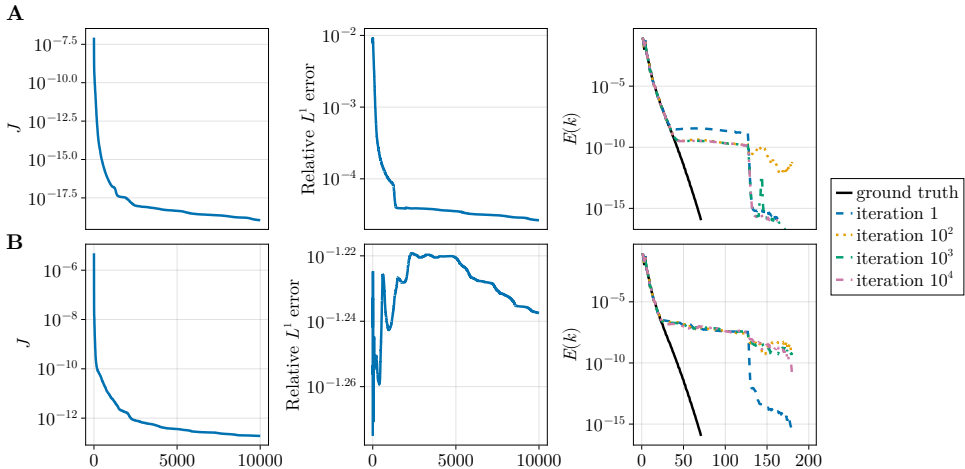


Fig. 8: Assimilation test results for vanilla-4DVAR. Left: decay of the cost function during L-BFGS optimization. Center: evolution of the relative L^1 error, demonstrating consistency between objective reduction and state recovery. Right: energy spectra of the assimilated and ground-truth initial conditions. Results are shown for two observation sparsity levels: **(A)** 2^2 , for which successful convergence is achieved, and **(B)** 4^2 , where the optimization begins to diverge.

physics-informed loss was chosen as $\lambda_{\text{data}} = 1/\sigma^2$, where σ denotes the standard deviation of the observation noise. In the noise-free case ($\sigma = 0$), we set $\lambda_{\text{data}} = 5 \times 10^3$ to prioritize ensuring the neural network fits the experimental data before strictly enforcing governing physics. As presented in Table 9, we empirically found that sufficiently large λ_{data} and $\lambda_{\text{divergence}}$ are essential to guide the optimization process toward the correct solution.

λ_{data}	5×10^0	5×10^1	5×10^2	5×10^3	5×10^4
error	0.993	0.438	0.389	0.386	0.379
$\lambda_{\text{divergence}}$	5×10^{-1}	5×10^0	5×10^1	5×10^2	5×10^3
error	0.472	0.401	0.393	0.397	0.386

Table 9: Relative L^1 errors for varying penalty weights. The experimental setup is identical to subsection 5.2.

Appendix E. Comparison with Fourier-based spectral truncation. To further contextualize the performance of neural reparameterization, we introduce an additional baseline defined by:

$$J_{\text{Fourier},\kappa}(u_0) = J_{\text{vanilla}}(f_{\kappa}(u_0)),$$

where f_{κ} denotes a smoothing operator that subsamples every κ -th grid point in physical space and reconstructs the full field via Fourier interpolation. This operator effectively imposes a low-pass filter, restricting the solution to lower-wavenumber components while suppressing high-frequency noise.

Empirically, setting $\kappa = 4$ provides the optimal trade-off between regularization and resolution, achieving a relative L^1 reconstruction error of 2.78×10^{-1} . This performance is nearly on par with the neural reparameterization result (2.73×10^{-1}), demonstrating that low-wavenumber constraints alone can provide a significant regularization benefit.

However, a spectral analysis reveals a clear distinction between these approaches. As illustrated in Figure 9, the Fourier truncation method ($\kappa = 4$) inherently fails to recover the high-wavenumber portion of the energy spectrum relative to the ground truth. In contrast, both the neural reparameterization and CVT approaches successfully reconstruct the energy spectrum across a broader range of scales, including higher-frequency components. These results suggest that while Fourier-based smoothing mimics the noise-suppression aspect of our approach, neural reparameterization provides a more sophisticated inductive bias. It allows the model to preserve physically meaningful fine-scale structures that are lost under simple spectral truncation.

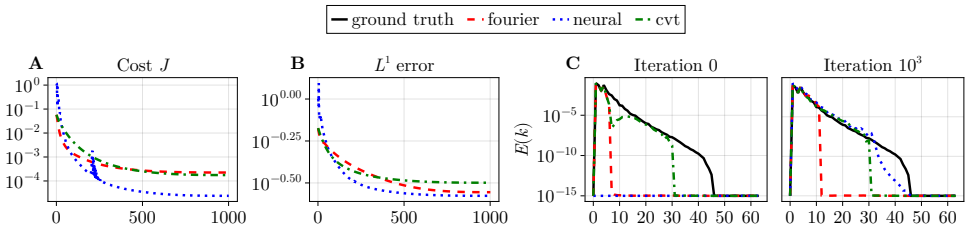


Fig. 9: Comparison of energy spectra for Fourier smoothing, neural reparameterization, and CVT methods against the ground truth. The experimental setup is identical to subsection 4.1.

Appendix F. Hyperparameter sensitivity analysis. Building on the default configuration described in Appendix D, we examine the sensitivity of neural- and PINN-4DVAR to key hyperparameters. All experiments were conducted using the same benchmark configuration as in subsection 6.1. While both methods exhibited robust performance across a wide range of settings, we observed failure cases when the number of Fourier features was large and the initial learning rate was high.

F.1. Hyperparameter sweep for neural-4DVAR. We performed a hyperparameter sweep for neural-4DVAR by varying the network width $\{2^5, 2^6, 2^7\}$, depth $\{2, 3, 4, 5\}$, initial learning rate $\{10^{-2}, 10^{-3}, 10^{-4}\}$, and the number of Fourier features $\{1, 3, 5, 7\}$. The results are summarized in Table 10. A notable observation is the mismatch between objective value reduction and reconstruction accuracy. While the failure case attains the lowest final cost, it simultaneously produces the largest reconstruction error. This behavior suggests overfitting to spurious high-frequency modes, underscoring the importance of balancing representational capacity and optimization stability in neural parameterizations of 4DVAR.

F.2. Hyperparameter sweep for PINN-4DVAR. We conducted a hyperparameter sweep for PINN-4DVAR by varying the network width $\{2^4, 2^5, 2^6, 2^7\}$, depth $\{2, 3, 4, 5\}$, initial learning rate $\{10^{-2}, 10^{-3}, 10^{-4}\}$, number of Fourier modes $\{1, 3, 5, 7\}$, and the number of training epochs $\{10^3, 5 \times 10^3, 10^4\}$. The top five performing configurations, ranked by reconstruction accuracy, together with two repre-

	width	depth	lr ₀	#FF	error	cost
Top 5	128	5	10 ⁻²	3	3.519E-1	5.962E-6
	128	4	10 ⁻²	3	<u>3.749E-1</u>	<u>4.884E-6</u>
	128	5	10 ⁻³	3	3.901E-1	2.728E-5
	64	5	10 ⁻²	3	4.042E-1	6.920E-6
	64	5	10 ⁻²	3	4.100E-1	7.413E-6
Failure case	64	5	10 ⁻²	7	9.163E-1	4.346E-6

Table 10: Hyperparameter sweep results for neural-4DVAR. #FF denotes the number of Fourier features. Boldface and underline indicate the best and the second best in each column. The top five configurations (ranked by accuracy) and the failure case show that lower error does not necessarily coincide with lower cost.

sentative failure cases, are reported in Table 11.

We observe that large networks are prone to divergence when trained with a high initial learning rate of 10⁻² (see Failure cases 1 and 2 in Table 11). In contrast, smaller learning rates ($\leq 10^{-3}$) consistently stabilize training and yield relative L^1 errors below 0.5. Notably, a compact SPINN architecture with width 16, depth 2, and five Fourier modes already achieves a relative error below 0.4, indicating that moderate model capacity is sufficient for this benchmark when combined with physics-informed regularization.

	width	depth	lr ₀	#FF	epochs	error
Top 5	16	2	10 ⁻²	5	10 ⁴	3.641E-1
	32	2	10 ⁻³	7	10 ⁴	3.606E-1
	32	3	10 ⁻³	7	10 ⁴	3.605E-1
	32	4	10 ⁻³	3	10 ⁴	<u>3.585E-1</u>
	32	5	10 ⁻³	5	10 ⁴	3.578E-1
Failure case 1	128	3	10 ⁻²	7	5 × 10 ³	3.864E-1
					10 ⁴	8.146E2
Failure case 2	128	4	10 ⁻²	5	5 × 10 ³	3.826E-1
					10 ⁴	1.398E2

Table 11: Hyperparameter sweep results for PINN-4DVAR. #FF denotes the number of Fourier features. The failure cases indicate that too large an initial learning rate 10⁻² causes divergence after 5 × 10³ steps.

Appendix G. Derivation of the vorticity form. In this appendix, we derive the vorticity form of the incompressible Navier–Stokes equations. Starting from the velocity-pressure form

$$\frac{\partial \mathbf{u}}{\partial t} + (\mathbf{u} \cdot \nabla) \mathbf{u} + \frac{1}{\rho} \nabla p = \nu \nabla^2 \mathbf{u} + \mathbf{f}$$

$$\nabla \cdot \mathbf{u} = 0,$$

we take the curl of the momentum equation:

$$\nabla \times \left(\frac{\partial \mathbf{u}}{\partial t} + (\mathbf{u} \cdot \nabla) \mathbf{u} + \frac{1}{\rho} \nabla p \right) = \nabla \times (\nu \nabla^2 \mathbf{u} + \mathbf{f}).$$

The pressure gradient term vanishes after taking the curl, and introducing the vorticity $\omega = \nabla \times \mathbf{u}$ gives

$$\frac{\partial \omega}{\partial t} + \nabla \times [(\mathbf{u} \cdot \nabla) \mathbf{u}] = \nu \nabla^2 \omega + \nabla \times \mathbf{f}.$$

To simplify the nonlinear term $(\mathbf{u} \cdot \nabla) \mathbf{u}$, we use the vector identity

$$(\mathbf{u} \cdot \nabla) \mathbf{u} = \nabla \left(\frac{1}{2} |\mathbf{u}|^2 \right) - \mathbf{u} \times \omega.$$

Taking the curl yields

$$\begin{aligned} \nabla \times [(\mathbf{u} \cdot \nabla) \mathbf{u}] &= \nabla \times \left[\nabla \left(\frac{1}{2} |\mathbf{u}|^2 \right) - \mathbf{u} \times \omega \right] \\ &= -\nabla \times (\mathbf{u} \times \omega), \end{aligned}$$

since the curl of a gradient is zero. Expanding the remaining term,

$$-\nabla \times (\mathbf{u} \times \omega) = \omega(\nabla \cdot \mathbf{u}) - (\omega \cdot \nabla) \mathbf{u} + (\mathbf{u} \cdot \nabla) \omega.$$

By incompressibility ($\nabla \cdot \mathbf{u} = 0$), the term $\omega(\nabla \cdot \mathbf{u})$ is zero. Collecting terms, the vorticity equation becomes

$$\frac{\partial \omega}{\partial t} + (\mathbf{u} \cdot \nabla) \omega = (\omega \cdot \nabla) \mathbf{u} + \nu \nabla^2 \omega + \nabla \times \mathbf{f}.$$

For the two-dimensional Kolmogorov flow, the vortex stretching term $(\omega \cdot \nabla) \mathbf{u}$ vanishes since the curl is two-dimensional and thus satisfies $\omega \cdot \nabla = 0$. The curl of the Kolmogorov forcing (3.2) is given by

$$\nabla \times \mathbf{f} = -0.1 \omega - 4 \cos(4y).$$

Substituting this expression into the vorticity equation yields (3.3).

For the three-dimensional incompressible Navier–Stokes equations with $\mathbf{f} = 0$, as considered in subsection 6.2, the vortex stretching term $(\omega \cdot \nabla) \mathbf{u}$ is nonzero in general. Consequently, the vorticity formulation includes this additional nonlinear contribution, leading to (6.1).

Acknowledgments. The author thanks Jaeyeon Oh and A N M Nafiz Abeer for comments on the presentation, and Byung-Jun Yoon for supporting computing resources. The author also thanks two very knowledgeable anonymous referees for their comments.

REFERENCES

- [1] R. N. BANNISTER, *A review of forecast error covariance statistics in atmospheric variational data assimilation. i: Characteristics and measurements of forecast error covariances*, Quarterly Journal of the Royal Meteorological Society: A journal of the atmospheric sciences, applied meteorology and physical oceanography, 134 (2008), pp. 1951–1970.
- [2] R. N. BANNISTER, *A review of forecast error covariance statistics in atmospheric variational data assimilation. ii: Modelling the forecast error covariance statistics*, Quarterly Journal of the Royal Meteorological Society: A journal of the atmospheric sciences, applied meteorology and physical oceanography, 134 (2008), pp. 1971–1996.

- [3] F. BAO, Z. ZHANG, AND G. ZHANG, *A score-based filter for nonlinear data assimilation*, Journal of Computational Physics, 514 (2024), p. 113207.
- [4] M. BLONDEL, Q. BERTHET, M. CUTURI, R. FROSTIG, S. HOYER, F. LLINARES-LÓPEZ, F. PEDREGOSA, AND J.-P. VERT, *Efficient and modular implicit differentiation*, Advances in neural information processing systems, 35 (2022), pp. 5230–5242.
- [5] M. BOCQUET, A. FARCHI, T. S. FINN, C. DURAND, S. CHENG, Y. CHEN, I. PASMANS, AND A. CARRASSI, *Accurate deep learning-based filtering for chaotic dynamics by identifying instabilities without an ensemble*, Chaos: An Interdisciplinary Journal of Nonlinear Science, 34 (2024).
- [6] G. BOFFETTA AND R. E. ECKE, *Two-dimensional turbulence*, Annual review of fluid mechanics, 44 (2012), pp. 427–451.
- [7] J. BRADBURY, R. FROSTIG, P. HAWKINS, M. J. JOHNSON, C. LEARY, D. MACLAURIN, G. NECULA, A. PASZKE, J. VANDERPLAS, S. WANDERMAN-MILNE, AND Q. ZHANG, *JAX: composable transformations of Python+NumPy programs*, 2018, <http://github.com/google/jax>.
- [8] C. CANUTO, A. QUARTERONI, M. Y. HUSSAINI, AND T. A. ZANG JR, *Spectral methods: evolution to complex geometries and applications to fluid dynamics*, Springer, 2007.
- [9] M. H. CARPENTER AND C. A. KENNEDY, *Fourth-order 2N-storage Runge-Kutta schemes*, tech. report, National Aeronautics and Space Administration Langley Research Center, 1994.
- [10] G. J. CHANDLER AND R. R. KERSWELL, *Invariant recurrent solutions embedded in a turbulent two-dimensional kolmogorov flow*, Journal of Fluid Mechanics, 722 (2013), pp. 554–595.
- [11] J. CHO, S. NAM, H. YANG, S.-B. YUN, Y. HONG, AND E. PARK, *Separable physics-informed neural networks*, Advances in Neural Information Processing Systems, 36 (2024).
- [12] B. CHOI AND Y. LEE, *Sampling error mitigation through spectrum smoothing: First experiments with ensemble transform kalman filters and lorenz models*, Physica D: Nonlinear Phenomena, 472 (2025), p. 134436.
- [13] R. COURANT, K. FRIEDRICHS, AND H. LEWY, *Über die partiellen differenzgleichungen der mathematischen physik*, Mathematische annalen, 100 (1928), pp. 32–74.
- [14] S. DANISCH AND J. KRUMBIEGEL, *Makie.jl: Flexible high-performance data visualization for Julia*, Journal of Open Source Software, 6 (2021), p. 3349, <https://doi.org/10.21105/joss.03349>, <https://doi.org/10.21105/joss.03349>.
- [15] DEEPMIND, I. BABUSCHKIN, K. BAUMLI, A. BELL, S. BHUPATIRAJU, J. BRUCE, P. BUCHLOVSKY, D. BUDDEN, T. CAI, A. CLARK, I. DANIELKA, A. DEDIEU, C. FANTACCI, J. GODWIN, C. JONES, R. HEMSLEY, T. HENNIGAN, M. HESSEL, S. HOU, S. KAPTUROWSKI, T. KECK, I. KEMAEV, M. KING, M. KUNESCH, L. MARTENS, H. MERZIC, V. MIKULIK, T. NORMAN, G. PAPAMAKARIOS, J. QUAN, R. RING, F. RUIZ, A. SANCHEZ, L. SARTRAN, R. SCHNEIDER, E. SEZENER, S. SPENCER, S. SRINIVASAN, M. STANOJEVIĆ, W. STOKOWIEC, L. WANG, G. ZHOU, AND F. VIOLA, *The DeepMind JAX Ecosystem*, 2020, <http://github.com/google-deeppmind>.
- [16] Y. DU, M. WANG, AND T. A. ZAKI, *State estimation in minimal turbulent channel flow: A comparative study of 4DVar and PINN*, International Journal of Heat and Fluid Flow, 99 (2023), p. 109073.
- [17] E. DUPONT, A. GOLINSKI, M. ALIZADEH, Y. W. TEH, AND A. DOUCET, *COIN: COmpression with implicit neural representations*, in Neural Compression: From Information Theory to Applications – Workshop @ ICLR 2021, 2021, <https://openreview.net/forum?id=yekxhcsVi4>.
- [18] R. M. ERRICO, *What is an adjoint model?*, Bulletin of the American Meteorological Society, 78 (1997), pp. 2577–2592.
- [19] G. EVENSEN, *Sequential data assimilation with a nonlinear quasi-geostrophic model using monte carlo methods to forecast error statistics*, Journal of Geophysical Research: Oceans, 99 (1994), pp. 10143–10162.
- [20] T. FRERIX, D. KOCHKOV, J. SMITH, D. CREMERS, M. BRENNER, AND S. HOYER, *Variational data assimilation with a learned inverse observation operator*, in International Conference on Machine Learning, PMLR, 2021, pp. 3449–3458.
- [21] X. GLOROT AND Y. BENGIO, *Understanding the difficulty of training deep feedforward neural networks*, in Proceedings of the thirteenth international conference on artificial intelligence and statistics, JMLR Workshop and Conference Proceedings, 2010, pp. 249–256.
- [22] C. R. HARRIS, K. J. MILLMAN, S. J. VAN DER WALT, R. GOMMERS, P. VIRTANEN, D. COUNAPEAU, E. WIESER, J. TAYLOR, S. BERG, N. J. SMITH, ET AL., *Array programming with NumPy*, nature, 585 (2020), pp. 357–362.
- [23] H. HERSBACH, B. BELL, P. BERRISFORD, S. HIRAHARA, A. HORÁNYI, J. MUÑOZ-SABATER, J. NICOLAS, C. PEUBEY, R. RADU, D. SCHEPERS, ET AL., *The era5 global reanalysis*, Quarterly journal of the royal meteorological society, 146 (2020), pp. 1999–2049.

- [24] S. HOYER, J. SOHL-DICKSTEIN, AND S. GREYDANUS, *Neural reparameterization improves structural optimization*, in NeurIPS 2019 Workshop on Solving Inverse Problems with Deep Networks, 2019, <https://openreview.net/forum?id=Bkcc3m3q8B>.
- [25] L. HUANG, L. GIANINAZZI, Y. YU, P. D. DUEBEN, AND T. HOEFLER, *Diffda: a diffusion model for weather-scale data assimilation*, in Proceedings of the 41st International Conference on Machine Learning, 2024, pp. 19798–19815.
- [26] B. R. HUNT, E. J. KOSTELICH, AND I. SZUNYOGH, *Efficient data assimilation for spatiotemporal chaos: A local ensemble transform kalman filter*, *Physica D: Nonlinear Phenomena*, 230 (2007), pp. 112–126.
- [27] N. KANG, J. OH, Y. HONG, AND E. PARK, *PIG: Physics-informed gaussians as adaptive parametric mesh representations*, in The Thirteenth International Conference on Learning Representations, 2025, <https://openreview.net/forum?id=y5B0ca4mjt>.
- [28] G. E. KARNIADAKIS, I. G. KEVREKIDIS, L. LU, P. PERDIKARIS, S. WANG, AND L. YANG, *Physics-informed machine learning*, *Nature Reviews Physics*, 3 (2021), pp. 422–440.
- [29] B. KERBL, G. KOPANAS, T. LEIMKÜHLER, AND G. DRETTAKIS, *3d gaussian splatting for real-time radiance field rendering.*, *ACM Trans. Graph.*, 42 (2023), pp. 139–1.
- [30] D. P. KINGMA AND J. BA, *Adam: A method for stochastic optimization*, arXiv preprint arXiv:1412.6980, (2014).
- [31] 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 (2021), p. e2101784118.
- [32] A. KRISHNAPRIYAN, A. GHOLAMI, S. ZHE, R. KIRBY, AND M. W. MAHONEY, *Characterizing possible failure modes in physics-informed neural networks*, *Advances in neural information processing systems*, 34 (2021), pp. 26548–26560.
- [33] R. K. KRUEGER AND M. WARD, *Jax-rnafold: scalable differentiable folding*, *Bioinformatics*, 41 (2025), p. btaf203.
- [34] I. E. LAGARIS, A. LIKAS, AND D. I. FOTIADIS, *Artificial neural networks for solving ordinary and partial differential equations*, *IEEE transactions on neural networks*, 9 (1998), pp. 987–1000.
- [35] F.-X. LE DIMET AND O. TALAGRAND, *Variational algorithms for analysis and assimilation of meteorological observations: theoretical aspects*, *Tellus A: Dynamic Meteorology and Oceanography*, 38 (1986), pp. 97–110.
- [36] S. L. LI, A. ZHANG, B. CHEN, H. MATUSIK, C. LIU, D. RUS, AND V. SITZMANN, *Controlling diverse robots by inferring jacobian fields with deep networks*, *Nature*, (2025), pp. 1–7.
- [37] Z. LI, B. DONG, AND P. ZHANG, *Latent assimilation with implicit neural representations for unknown dynamics*, *Journal of Computational Physics*, 506 (2024), p. 112953.
- [38] C. LIU, Q. XIAO, AND B. WANG, *An ensemble-based four-dimensional variational data assimilation scheme. part i: Technical formulation and preliminary test*, *Monthly Weather Review*, 136 (2008), pp. 3363–3373.
- [39] D. C. LIU AND J. NOCEDAL, *On the limited memory bfgs method for large scale optimization*, *Mathematical programming*, 45 (1989), pp. 503–528.
- [40] E. N. LORENZ, *THE ESSENCE OF CHAOS*, University of Washington Press, 1993.
- [41] I. LOSCHILOV AND F. HUTTER, *SGDR: Stochastic gradient descent with warm restarts*, in International Conference on Learning Representations, 2017, <https://openreview.net/forum?id=Skq89Scxx>.
- [42] L. LU, R. PESTOURIE, W. YAO, Z. WANG, F. VERDUGO, AND S. G. JOHNSON, *Physics-informed neural networks with hard constraints for inverse design*, *SIAM Journal on Scientific Computing*, 43 (2021), pp. B1105–B1132.
- [43] P. LYNCH, *The origins of computer weather prediction and climate modeling*, *Journal of computational physics*, 227 (2008), pp. 3431–3444.
- [44] B. MILDENHALL, P. P. SRINIVASAN, M. TANCIK, J. T. BARRON, R. RAMAMOORTHY, AND R. NG, *Nerf: Representing scenes as neural radiance fields for view synthesis*, *Communications of the ACM*, 65 (2021), pp. 99–106.
- [45] M. MORTENSEN AND H. P. LANGTANGEN, *High performance python for direct numerical simulations of turbulent flows*, *Computer Physics Communications*, 203 (2016), pp. 53–65.
- [46] H. NGODOCK, M. CARRIER, S. SMITH, AND I. SOUOPGUI, *Weak and strong constraints variational data assimilation with the ncom-4dvar in the agulhas region using the representer method*, *Monthly Weather Review*, 145 (2017), pp. 1755–1764.
- [47] J. OH, J. LEE, AND Y. HONG, *Machine learning-based nonlinear nudging for chaotic dynamical systems*, arXiv preprint arXiv:2508.05778, (2025).
- [48] F. RABIER, J.-N. THÉPAUT, AND P. COURTIER, *Extended assimilation and forecast experiments with a four-dimensional variational assimilation system*, *Quarterly Journal of the Royal*

- Meteorological Society, 124 (1998), pp. 1861–1887.
- [49] N. RAHAMAN, A. BARATIN, D. ARPIT, F. DRAXLER, M. LIN, F. HAMPRECHT, Y. BENGIO, AND A. COURVILLE, *On the spectral bias of neural networks*, in International conference on machine learning, PMLR, 2019, pp. 5301–5310.
 - [50] 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 (2019), pp. 686–707.
 - [51] J. RICHTER-POWELL, Y. LIPMAN, AND R. T. CHEN, *Neural conservation laws: A divergence-free perspective*, Advances in Neural Information Processing Systems, 35 (2022), pp. 38075–38088.
 - [52] F. ROZET AND G. LOUPPE, *Score-based data assimilation*, Advances in Neural Information Processing Systems, 36 (2023), pp. 40521–40541.
 - [53] A. SHOCHER, N. COHEN, AND M. IRANI, *“zero-shot” super-resolution using deep internal learning*, in Proceedings of the IEEE conference on computer vision and pattern recognition, 2018, pp. 3118–3126.
 - [54] V. SITZMANN, J. MARTEL, A. BERGMAN, D. LINDELL, AND G. WETZSTEIN, *Implicit neural representations with periodic activation functions*, Advances in neural information processing systems, 33 (2020), pp. 7462–7473.
 - [55] S. WANG, S. SANKARAN, AND P. PERDIKARIS, *Respecting causality for training physics-informed neural networks*, Computer Methods in Applied Mechanics and Engineering, 421 (2024), p. 116813.
 - [56] S. WANG, Y. TENG, AND P. PERDIKARIS, *Understanding and mitigating gradient flow pathologies in physics-informed neural networks*, SIAM Journal on Scientific Computing, 43 (2021), pp. A3055–A3081.
 - [57] Y. YANG, X. CHENG, D. GILES, S. CHENG, Y. HE, X. XUE, B. CHEN, AND Y. HU, *Tensorvar: Efficient four-dimensional variational data assimilation*, in Forty-second International Conference on Machine Learning, 2025.
 - [58] Z. YIN, H. CLERCX, AND D. MONTGOMERY, *An easily implemented task-based parallel scheme for the Fourier pseudospectral solver applied to 2d Navier–Stokes turbulence*, Computers & fluids, 33 (2004), pp. 509–520.