Latent Assimilation with Implicit Neural Representations for Unknown Dynamics

Zhuoyuan Li

zy.li@stu.pku.edu.cn

1 Introduction

Data assimilation (DA) has increasingly become an indispensable tool across a multitude of disciplines, including weather forecasting, climate modeling, oceanography, ecology, and even economics. The core idea of data assimilation is the integration of the information derived from disparate sources, such as in situ observations and satellites data, with physical models that describe the underlying dynamics of the system. The objective of DA is to obtain the best estimate of the system states, in line with both observation data and physical models, while considering their inherent uncertainties.

From a mathematical standpoint, DA algorithms are efficient and practical solutions to a particular inverse problem

$$\boldsymbol{y} = \mathcal{F}(\boldsymbol{x}) + \boldsymbol{\eta}.$$

Here, $\boldsymbol{x}$ signifies system states of interest, with $\boldsymbol{y}$ representing observation data, $\mathcal{F}$ being the observational operator, and $\boldsymbol{\eta}$ standing for an additive noise. For instance, within the context of modern numerical weather prediction (NWP), $\boldsymbol{x}$ may consist of all atmospheric variables at a single or multiple time steps, with DA framworks employed to determine optimal initial conditions. Such problem is often addressed by Bayesian point of view, where the posterior probability $p(\boldsymbol{x} \mid \boldsymbol{y})$ is evaluated as

$$p(\boldsymbol{x} \mid \boldsymbol{y}) = \frac{p(\boldsymbol{y} \mid \boldsymbol{x})p(\boldsymbol{y})}{p(\boldsymbol{y})} \propto p(\boldsymbol{y} \mid \boldsymbol{x})p(\boldsymbol{x})$$

and subsequently maximized. The likelihood term $p(\boldsymbol{y} \mid \boldsymbol{x})$ and the prior term $p(\boldsymbol{x})$ correspond to the observation model and the system's underlying dynamic model, respectively. Hence, maximizing the posterior probability effectively equates to adjusting the state $\boldsymbol{x}$ to best align with both the observation data and the physical models.

Current DA frameworks largely draw upon either three/four-dimensional variational (3/4d-Var) methods or Kalman filter methods. The former employs a variational optimization-based approach that assimilates data over a fixed time window, while the latter utilizes a recursive estimation algorithm that operates sequentially in real-time. Nonetheless, both these are faced with challenges linked to system dimensionality. As the dimension of the state $\boldsymbol{x}$ increases dramatically in practical scenarios, the optimization processes for variational methods become inevitably expensive, and the direct storage and updates required for the covariance matrices in Kalman filters turn infeasible. Although the ensemble methods can alleviate these issues to some extent, they confine the assimilated states within a low-dimensional subspace, potentially harming the overall performance. Another significant limitation is the restricted handling of nonlinearity. Consequently, the development of advanced DA frameworks capable of efficiently managing high-dimensional systems is an area of considerable interest.

1.1 Leveraging Reduced-Order-Models for Latent Assimilation

One promising approach to address the issues related to high-dimensionality is the use of Reduced-Order-Models (ROMs) as approximations for the system. At the core of these methods is the assumption that a low-dimensional submanifold $\mathcal{X}'$ exists within the system state space $\mathcal{X}$, where all system states $\boldsymbol{x}$ reside. By defining a suitable parameterization $\{\boldsymbol{x}_{\boldsymbol{z}}\}_{\boldsymbol{z}\in\mathcal{Z}}$ of $\mathcal{X}'$, a one-to-one correspondence between the parameterized (or latent) space $\mathcal{Z}$ and the submanifold $\mathcal{X}'$ can be established. With the explicit derivation of the dynamics on $\mathcal{Z}$ and the bijective relation between $\mathcal{Z}$ and $\mathcal{X}'$, the DA algorithms can be transitioned to operate within the latent space. We refer to such DA process within the latent space as latent assimilation (LA). Given that the dimension of the latent space typically satisfies $\dim \mathcal{Z} \ll \dim \mathcal{X}'$, both the storage and computational cost of LA are reduced in comparison with the original DA.

method	encoder-decoder	latent dynamics	scalability	efficiency	mesh-free
ETKF-Q-L[37]	fully-connected	ReZero	low	high	no
RNN-ETKF[36]	RNN	RC	medium	high	no
NIROM-DA[35]	POD/PCA	LSTM	low	medium	no
GLA [13]	POD/PCA + Conv1d	LSTM	low	medium	no
LAINR (ours)	INR	Neural ODE	high	medium	yes

Table 1: Comparison of different methods for latent assimilation. The method names are borrowed from the original papers.

Classical ROMs are usually linear which make use of proper orthogonal decomposition (POD) method or wavelet decomposition to extract the system's dominant modes a collection of history snapshots. Under these circumstances, $\mathcal{X}'$ is defined as the optimal low-dimensional subspace of $\mathcal{X}$, spanned by the dominant modes, which minimizes the reconstruction error. The corresponding parameterizations $\boldsymbol{z} \in \mathcal{Z}$ for each state $\boldsymbol{x}$ are the projection coefficient vectors onto each mode. These methods are not only straightforward to implement, but also offer simple linear bijective mappings between $\mathcal{Z}$ and $\mathcal{X}'$, which simplify the derivation of the latent dynamics on $\mathcal{Z}$. However, the reconstruction accuracy is heavily dependent on the number of dominant modes, and as the system becomes more complicate, much more dominate modes are required to reconstruct the small-scale structures, which decreases the effectiveness of reduction of dimensionality.

1.2 Related works and contributions

In recent years, the explosive growth of Machine Learning (ML) and, specifically, Deep Learning (DL) methods has made remarkable strides in a wide array of disciplines, including image restoration, natural language processing, and generative modeling. The application of DL in solving complex physical problems, such as solving differential equations [38, 46, 43], discovering underlying physical laws [29, 6, 45], and simulating spatio-temporal dynamics [30, 28], has attracted significant attention and achieved promising results.

In the realm of atmospheric sciences, DL-based approaches have great success in handling a variety of tasks, including weather forecasting, post-processing as well as data assimilation. The positive trend has dispelled doubts [42] about DL's capability to manage intricate dynamics, with notable models like FourCastNet [34], followed by Pangu-Weather [5], GraphCast [27] and FengWu [8] focusing on weather forecasting tasks.

The idea of ROMs, which utilize dimensionality reduction methods to represent high-dimensional space with lower-dimensional ones, has been well developed in the field of DL, and some early works [37, 36, 35, 13] have been proposed in the context of data assimilation. By transforming system state variables into lower-dimensional latent codes using encoder-decoder network structures, surrogate models for latent dynamics are trained simultaneously. However, existing ROMs based on fully-connected networks, Recurrent Neural Networks (RNNs), and Long Short-Term Memory (LSTM) networks face challenges regarding scalability, computational efficiency, and applicability to fixed-grid input data. Table 1 summarizes the comparative strengths and weaknesses of these methods.

The work of [37] selects trivial fully-connected deep neural networks for both the encoder and the decoder, and the latent dynamics is approximated via ReZero[3] as a variant of ResNet[22] to enhance the stability. In spite of the unawareness of explicit dynamics, the performance is comparable even with classical DA with known dynamics. Besides, the forward propagation is accelerated by latent embedding. Unfortunately, the fully-connected architecture can hardly be applied to multi-dimensional cases since the storage and training for the huge number of network parameters are unaffordable. Recurrent Neural Networks (RNNs) can serve as an alternative way for evolving latent dynamics as proposed in [36], which merges the encoder-decoder structure into RNNs. The training of RNNs takes the advantage of Reservoir Computing (RC) [31] to make the forward propagation faster, but as suggested in [1, 36], the RC approaches are more suitable for short-term prediction tasks. Meanwhile, creating patches when scaling to high-dimensional problems can probably result in disagreement between neighborhoods. To effectively capture long-term dependencies in sequencial data, Long Short-Term Memory (LSTM) networks have been employed in [35, 13], but they face the same issues as those of [37] that the model is not scalable to high-dimensional problems. In addition, the LSTM-based models slowen the forward propagation process, which needs to be called frequently in assimilation algorithms. Finally, all the the architectures mentioned above require the input data lying on certain fixed grids unless interpolations probably resulting in additional errors are applied.

In this work, we present the Latent Assimilation with Implicit Neural Representations (LAINR) framework, which is to our best knowledge, the first mesh-free DA method designed for multi-dimensional unknown dynamics. By proposing the LAINR framework, we aim to make a meaningful contribution to the ongoing exploration of DL and ML techniques in data assimilation and atmospheric sciences. The primary contributions of our work are:

- We introduce a mesh-free encoder-decoder structure for LA, employing Implicit Neural Representations (INR) to approximate system states. This structure is scalable and applicable to multi-dimensional problems.
- We validate the effectiveness of our proposed method through experiments on the shallow water equations, demonstrating the high accuracy, stability, and robustness of the LAINR framework.

2 Latent-space embedding and dynamics learning

Latent Assimilation (LA) is essentially a form of Data Assimilation (DA) methodology where the model space is characterized by a latent space, denoted by $\mathcal{Z}$, rather than the physical state space $\mathcal{X}$. Similar to conventional DA methods, the construction of an effective LA model necessitates the specification of the forward propagation on the latent space $\mathcal{Z}$ and the associated observation operator that bridges $\mathcal{Z}$ to the observational space $\mathcal{Y}$ (Figure 1). Additionally, uncertainty considerations brought by noise must be incorporated into these mappings. Our work aims to explore the choice of latent-space embeddings and surrogate models for latent dynamics under this LA context.

$$\begin{tikzcd}
\cdots \arrow[r, "\mathcal{M}"] & \mathcal{X} \arrow[r, "\mathcal{M}"] \arrow[d, "\mathcal{H}"'] & \mathcal{X} \arrow[r, "\mathcal{M}"] \arrow[d, "\mathcal{H}"] & \cdots \\
 & \mathcal{Y} & \mathcal{Y} &
\end{tikzcd}$$

(a) classical DA framework

$$\begin{tikzcd}
\cdots \arrow[r, "\mathcal{G}"] & \mathcal{Z} \arrow[r, "\mathcal{G}"] \arrow[d, bend left, "\mathcal{D}"] & \mathcal{Z} \arrow[r, "\mathcal{G}"] \arrow[d, bend left, "\mathcal{D}"] & \cdots \\
\cdots \arrow[r, "\mathcal{M}"] & \mathcal{X} \arrow[r, "\mathcal{M}"] \arrow[u, bend left, "\mathcal{E}"] \arrow[d, "\mathcal{H}"'] & \mathcal{X} \arrow[r, "\mathcal{M}"] \arrow[u, bend left, "\mathcal{E}"] \arrow[d, "\mathcal{H}"] & \cdots \\
 & \mathcal{Y} & \mathcal{Y} &
\end{tikzcd}$$

(b) latent assimilation (LA) framework

Figure 1: Comparison between classical DA and LA frameworks.

Our work operates under the assumption that a low-dimensional parameterization exists for the high-dimensional system states, thus, we present detailed discussions about the latent embedding and learning of latent dynamics. Let $\mathcal{X} = \mathbb{R}^n$ and $\mathcal{Z} = \mathbb{R}^m$ denote the system state space and the latent space, respectively, where n is considerably larger than m. The goal of latent embedding is to determine a pair of encoder $\mathcal{E} : \mathcal{X}' \to \mathcal{Z}$ and decoder $\mathcal{D} : \mathcal{Z} \to \mathcal{X}'$ that are essentially inverse mappings between the low-dimensional submanifold $\mathcal{X}'$ that encapsulates all possible system states, and the latent space $\mathcal{Z}$.

Let's assume that the system state variable we aim to assimilate is $\boldsymbol{u} : \mathbb{R}^d \to \mathbb{R}^c$, where d signifies the inherent dimension of the system (usually 2 or 3), and c refers to the number of features of interest (such as horizontal wind, temperature, humidity, and so forth). Given a fixed sampling grid $G \subseteq \mathbb{R}^d$, we explore various approaches to build a reduced-order model for the system state $\boldsymbol{x} = \{\boldsymbol{u}(\boldsymbol{p})\}_{\boldsymbol{p}\in G} \in \mathbb{R}^{|G|\times c} = \mathbb{R}^n$. Table 2 offers a summary of distinct choices for the encoder $\mathcal{E}$ and the decoder $\mathcal{D}$.

method	encoder $\mathcal{E}$	decoder $\mathcal{D}$	error to be minimized
POD	$\boldsymbol{x} \mapsto A^\mathsf{T}\boldsymbol{x}$	$\boldsymbol{z} \mapsto A\boldsymbol{z}$	$\left\|\left(I_n - A^\mathsf{T}A\right)\boldsymbol{X}\right\|_F^2$ subject to $A^\mathsf{T}A = I_m$
AE	$\boldsymbol{x} \mapsto f_\theta(\boldsymbol{x})$	$\boldsymbol{z} \mapsto g_\varphi(\boldsymbol{z})$	$\mathcal{L}(\boldsymbol{x}, g_\varphi(f_\theta(\boldsymbol{x})))$
INR	$\boldsymbol{x} \mapsto \operatorname{argmin}_{\boldsymbol{z}} \|I(\cdot;\boldsymbol{z},\boldsymbol{\varphi}) - \boldsymbol{u}\|_{\mu(G)}$	$\boldsymbol{z} \mapsto \{I(\boldsymbol{p};\boldsymbol{z},\boldsymbol{\varphi})\}_{\boldsymbol{p}\in G}$	$\|I(\cdot;\boldsymbol{z},\boldsymbol{\varphi}) - \boldsymbol{u}\|_{L^p(\mu)}$

Table 2: Comparison of different methods for latent embedding.

2.1 Classical linear ROMs

In the domain of classical ROMs, we typically encounter linear methods like Proper Orthogonal Decomposition (POD) or wavelet decomposition that provide a low-dimensional representation of the system. We introduce such linear ROMs by take the method of POD as an example. POD aims to identify a set of orthogonal basis vectors $\boldsymbol{\phi}_j{}_{j=1}^{m} \subseteq \mathbb{R}^n$, known as dominant modes, that minimize the projection error on expectation. Formally, given a series of historical ground truths or assimilated data $\boldsymbol{x}_k{}_{k=1}^{K}$, the objective of POD is to determine the dominant modes such that

$$\{\boldsymbol{\phi}_j\}_{j=1}^{m} = \underset{\boldsymbol{\phi}_j}{\operatorname{argmin}} \underset{\boldsymbol{x}\sim\mathscr{D}}{\mathbb{E}} \|\boldsymbol{x} - \widehat{\boldsymbol{x}}\|^2,$$

where $\mathscr{D}$ denotes the distribution of system states $\boldsymbol{x} \in \mathcal{X}$, and $\widehat{\boldsymbol{x}}$ is the projection of $\boldsymbol{x}$ onto the linear subspace spanned by $\{\boldsymbol{\phi}_j\}_{j=1}^{m}$. m is the number of dominant modes usually fixed in advance. Since the explicit form of $\mathscr{D}$ is often difficult to derive in practice, the expectation is approximated empirically, i.e.,

$$\{\boldsymbol{\phi}_j\}_{j=1}^{m} = \underset{\boldsymbol{\phi}_j}{\operatorname{argmin}} \frac{1}{K}\sum_{k=1}^{K} \|\boldsymbol{x}_k - \widehat{\boldsymbol{x}_k}\|^2.$$

It is worth noting that the projections $\widehat{\boldsymbol{x}_k}$ can be explicitly expressed as

$$\widehat{\boldsymbol{x}_k} = \sum_{j=1}^{m} \langle \boldsymbol{x}_k, \boldsymbol{\phi}_j \rangle \boldsymbol{\phi}_j,$$

where $\langle\cdot,\cdot\rangle$ denotes the inner product in $\mathcal{X} = \mathbb{R}^n$. Consequently, the optimization problem can be rewritten in the form

$$A := (\boldsymbol{\phi}_1, \cdots, \boldsymbol{\phi}_m)^{\mathsf{T}} = \underset{A}{\operatorname{argmin}} \frac{1}{K}\sum_{k=1}^{K} \left\|\boldsymbol{x}_k - A^{\mathsf{T}}A\boldsymbol{x}_k\right\|^2 = \underset{A}{\operatorname{argmin}} \sum_{k=1}^{K} \left\|\left(I_n - A^{\mathsf{T}}A\right)\boldsymbol{x}_k\right\|^2 \text{ s.t. } AA^{\mathsf{T}} = I_m,$$

which can be solved by the singular value decomposition (SVD) of the empirical covariance matrix $\boldsymbol{X}\boldsymbol{X}^{\mathsf{T}}$. More specifically, the target function is minimized when the rows of A are the first m left singular vectors of the matrix $\boldsymbol{X} = (\boldsymbol{x}_1, \cdots, \boldsymbol{x}_K) \in \mathbb{R}^{n\times K}$ (See Prop. A.1 for a detailed proof). Such operations are often referred to as the Principal Component Analysis (PCA).

Overall, the key benefit of POD in this context is that the dominant modes $\boldsymbol{\phi}_j j = 1^m$ can be obtained directly without training, offering an efficient and direct way to generate a linear reduced-order model, and the encoder-decoder mappings are defined as $\mathcal{E}^{\text{POD}} : \boldsymbol{x} \mapsto A\boldsymbol{x}$ and $\mathcal{D}^{\text{POD}} : \boldsymbol{z} \mapsto A^{\mathsf{T}}\boldsymbol{z}$, respectively.

2.2 Autoencoders: non-linear representations and limitations

Autoencoders, as a common neural network architecture in deep learning, introduce a non-linear dimensionality reduction approach that expands upon the concept of latent space. With an encoder $\mathcal{E}^{\text{AE}} = f_\theta$ and a decoder $\mathcal{D}^{\text{AE}} = g_\varphi$, this architecture can skillfully navigate high-dimensional input data, extracting a lower-dimensional latent space representation in the process. The encoder progressively decreases the data dimension with its multi-layered structure. By mirroring the encoder's architecture, the decoder takes the encoded representations and then reconstruct the original input data.

An autoencoder's objective is to minimize the empirical reconstruction error by employing a loss function to measure the divergence between the input and reconstructed data. The optimization process for

$$\theta, \varphi = \underset{\theta,\varphi}{\operatorname{argmin}} \sum_{k=1}^{K} \mathcal{L}(\boldsymbol{x}_k, g_\varphi(f_\theta(\boldsymbol{x}_k)))$$

promotes the learning of compressed meaningful data representation.

Compared with POD, autoencoders are capable of capturing non-linear relationships in the data, which is crucial for the latent dynamics learning since it introduces the possibility of simplifying a complex dynamical system in the physical space into a more manageable latent dynamics. Moreover, autoencoders bring flexibility due to their various structures. For instance, autoencoders can be designed to focus on local correlations with the help of convolutional networks. With structures like Recurrent Neural Networks (RNN), such as LSTM [24] and GRU [14], they can even process sequential data naturally.

Despite the promise of autoencoders, there still exist lots of limitations. Unlike linear ROMs like PCA, they do not ensure an optimal representation even in the empirical sense since the minimizer is not guaranteed to be obtained. Besides, the training process is considerably more time-consuming as the selections of network architectures and hyperparameters need to be determined carefully. Furthermore, autoencoders may face scalability challenges when dealing with high-dimensional problems as the number of grid points will increase exponentially with respect to the state dimension and resolutions. Such complexity could lead to cumbersome and resource-intensive training processes, which is where implicit neural representations (INRs) as our main focus may offer a viable alternative. The scalability and reduced complexity of INRs render them more conducive to high-dimensional problems, ensuring efficient training and implementation.

2.3 Implicit neural representations

Implicit Neural Representations (INRs) adopt a distinct approach to data modeling that deviates from traditional grid-vased or mesh-maséd methods such as the autoencoder described earlier. Rather than operating on state features on discrete positions, INRs ingest coordinate information and model signals or fields as continuous mappings, which provides a substantial advantage when working with continuous underlying signals or fields that are only available at discrete grid points or mesh intersections.

The core idea of INR is to use a parameterized neural network $I(\cdot;\boldsymbol{z},\boldsymbol{\varphi})$ to approximate the system state of interest $\boldsymbol{u}:\mathcal{X}\to\mathbb{R}^c$. The network aims to minimize the disparity between $I(\cdot;\boldsymbol{z},\boldsymbol{\varphi})$ and $\boldsymbol{u}$, measured with an appropriatem measure μ defined on a fixed grid G. The parameter $\boldsymbol{\varphi}$ captures consistent field properties and thus remains invariant across the dataset. On the other hand, the latent vector $\boldsymbol{z}$ containing the physical features shifts with different time steps. As an example, one may choose the L^2 norm with μ being a sum of Dirac delta functions $\mu = |G|^{-1}\sum_{\boldsymbol{p}\in G}\delta_{\boldsymbol{p}}$, which results in a minimization problem for

$$\|I(\cdot;\boldsymbol{z},\boldsymbol{\varphi})-\boldsymbol{u}\|_{L^2(\mu)} = \frac{1}{|G|}\sum_{\boldsymbol{p}\in G}|I(\boldsymbol{p};\boldsymbol{z},\boldsymbol{\varphi})-\boldsymbol{u}(\boldsymbol{p})|^2.$$

INRs present multiple advantages. Their independence of the explicit form of underlying grids or spatial resolutions promotes the flexibility. Meanwhile, similar to autoencoders, the non-linearity inherent to neural networks enables them to capture the intricate non-linear correlations, which outperforms classical linear ROMs like POD. Such advantages have been leveraged firstly in the field of shape modeling which have demostrate the superiority over grid-based and mesh-based archtectures [12, 33]. Since then, INRs have been applied to a variety of fields including image processing [44, 4, 17] and spatio-temporal dynamics [25, 9, 49].

In the context of INRs, the encoder involves solving an optimization problem

$$\mathcal{E}^{\mathrm{INR}}\left(\{\boldsymbol{u}(\boldsymbol{p})\}_{\boldsymbol{p}\in G}\right) = \boldsymbol{z} = \underset{\boldsymbol{z}}{\operatorname{argmin}}\,\|I(\cdot;\boldsymbol{z},\boldsymbol{\varphi})-\boldsymbol{u}\|_{L^p(\mu)}.$$

The high cost for the optimization problem is a major concern in most INR-based methodologies, but as explained later, the LAINR framework remains efficient since the optimization needs to be executed only at the initialization stage. Conversely, given the latent representation $\boldsymbol{z}$, the system state is directly recovered by $I(\cdot;\boldsymbol{z},\boldsymbol{\varphi})$ as a continuous signal or field.

2.4 Evolution of latent dynamics

Establishing a suitable surrogate model for the evolution of latent dynamics forms another crucial part for the LA framework, which not only addresses the non-linearity of the system dynamics but also integrates the auto-regressive property for the evolution of latent dynamics, regardless of whether or not the explicit form of the physical dynamics are provided. For simplicity, we assume that the latent dynamics are time-invariant and thus approximated by a parameterized surrogate model $\mathcal{G}_{\boldsymbol{\psi}}$ such that the differences between $\boldsymbol{z}_{k+1}$ and $\mathcal{G}_{\boldsymbol{\psi}}(\boldsymbol{z}_k)$ are minimized.

The choices for the structure of $\mathcal{G}_{\boldsymbol{\psi}}$ can be far more diverse, which strongly depends on the specific characteristics of the system state's temporal dynamics. For instance, Long Short-Term Memory (LSTM)[24] units and Gated Recurrent Units (GRUs)[14] are specially designed for sequential data, whose structure enable the network to remenber or forget previous states based on the context. Another kinds of modelling the sequential data are motivated by the classical numerical integrators for differential equations. Deep Residual Nets (ResNets)[22] have been promoted to accelerate the training process of deep networks at first, which add skip connections to mitigate the problem of vanishing and exploding gradients. However, ResNet models are essentially a discretization of the continuous-time system and are subject to issues related to step size and temporal resolution. On the contrary, Neural ODEs, introduced by [11, 10], consider a neural network as a continuous-time dynamical system. In the current work, we do not dive deeply into the optimal structure of the surrogate model, but the experimental performances suggest that using a network implemented by NeuralODE [11, 10] have shown superiority over ResNet. Furthermore, compared with ResNet, NeuralODE is more suitable for comtinuous-time dynamical system and more flexible as it allows variable lengths of time steps.

3 Latent assimilation framework

Classical data assimilation methods require observed data $\{\boldsymbol{y}_k\}_k$, observation operators $\{\mathcal{H}_k\}_k$, forward propagation operators $\{\mathcal{M}_k\}_k$ describing the system dynamics as well as a background estimate $\boldsymbol{x}_0$ for start-up. Meanwhile, the noise modeling is indispensable to help track the evolution of the error and then derive the assimilated state. Once the encoder-decoder mappings $\mathcal{E}$, $\mathcal{D}$ and the surrogate model $\mathcal{G}$ have been obtained via either deterministic algorithm such as PCA and wavelet decomposition, or training process with the aid of existing datasets, an analogous assimilation routine can be taken on the latent spaces.

First, denote the background estimate for the latent representation as $\boldsymbol{z}^b$, which is obtained by setting $\boldsymbol{z}^b = \mathcal{E}(\boldsymbol{x}^b)$. The latent dynamics is directly approximated by the surrogate model $\mathcal{G}$ after an offline training process, and the observation operator from the latent space to the physical observation space becomes the composed mapping $\mathcal{H}' = \mathcal{H} \circ \mathcal{D}$. The overall framework can be outlined as follows:

- Establish the encoder $\mathcal{E} : \mathcal{X}' \to \mathcal{Z}$ and the decoder $\mathcal{D} : \mathcal{Z} \to \mathcal{X}'$, together with the latent surrogate model $\mathcal{G} : \mathcal{Z} \to \mathcal{Z}$ on the latent space such that

$$\mathcal{D} \circ \mathcal{E} \approx \text{identity mapping on } \mathcal{X}'; \qquad \mathcal{D} \circ \mathcal{G} \circ \mathcal{E} \approx \mathcal{M}.$$

- Define the inputs for the assimilation algorithm on the latent space. $\boldsymbol{z}^b = \mathcal{E}(\boldsymbol{x}^b)$, $\mathcal{G}$ serves as the latent forward propagation operator and $\mathcal{H}' = \mathcal{H} \circ \mathcal{D}$ operates as the observation vector. The observation noise of $\mathcal{H}' = \mathcal{H} \circ \mathcal{D}$ remains the same as that of $\mathcal{H}$, and the model noise of $\mathcal{G}$ is empirically tuned by trials.
- Implement the assimilation process on the latent space with an existing well-developed DA algorithm.
- Extract the assimilated system states $\{\boldsymbol{x}_k^a\}_k$ from the corresponding latent assimilated representations via the decoder $\boldsymbol{x}_k^a = \mathcal{D}(\boldsymbol{z}_k^a)$.

The main advantage of this framework lies in its generalizability. It allows for integration of any deterministic or data-driven encoder-decoder pairs and surrogate models. Moreover, no explicit formula for the physical dynamics is required since the latent surrogate model is data-driven and trained out of scratch. Therefore, the framework supports the application to a wide range of problems, and provides versatility in assimilating complex system states.

4 Experiments

To study the scalability of the proposed LAINR framework, we choose multi-dimensional partial differential equations (PDEs) rather than chaotic ordinary differential equations (ODEs) such as the Lorenz-96 model adopted in [37, 36]. The reason is that the dimension of the state space of ODEs is usually much smaller compared with that of the state space of PDEs. The shallow-water model as well as the more realistic reanalysis data have been established as the test cases in this paper.

4.1 Test cases

4.1.1 the shallow-water model

Consider the following 3D spherical shallow-water equations on the sphere

$$\begin{aligned}\frac{\mathrm{d}\boldsymbol{u}}{\mathrm{d}t} &= -f\boldsymbol{k} \times \boldsymbol{u} - g\nabla h + \nu\Delta\boldsymbol{u},\\ \frac{\mathrm{d}h}{\mathrm{d}t} &= -h\nabla \cdot \boldsymbol{u} + \nu\Delta h,\end{aligned}$$

with the material derivative denoted by $\frac{\mathrm{d}}{\mathrm{d}t}$. The vorticity field $w = \nabla \times \boldsymbol{u}$ together with the thickness of the fluid layer becomes the system states to be assimilated. The parameters f, g, ν, Ω are all fixed consistent with the real earth surface as in [20]. The model is set up by initialize two symmetric zonal flow representing typical mid-latitude tropospheric jets, and the initial thickness h involves solving a boundary-value problem (balanced equation) [20]. The maximal zonal wind varies in order to create different trajectories for training and testing purposes. All the trajectories in the have been generated and recorded within a 128×64 regular lat-lon grid with $\Delta t \approx 1$ real hour. Readers may refer to Appendix B for detailed configurations.

4.1.2 ERA5 datasets

The ERA5 datasets [23] are the latest global atmospheric reanalysis datasets produced by the European Centre for Medium-Range Weather Forecasts (ECMWF), which are generated by assimilating observations from across the world into a numerical weather prediction model. In our experiments, we do not make comparison with the DA algorithms utilized in the generation of the ERA5 datasets, but treat the datasets the ground truth to provide a more realistic model than the ideal shallow-water model.

We extract both the Z500 (geopotential at 500 hPa) and the T850 (temperature at 850 hPa) fields from the WeatherBench [39] dataset, which is a resampled version of the ERA5 datasets specially designed for DL training. The choice of Z500 and T850 aligns with those of the early attempts [15, 41, 48, 40] of DL-based weather forecasting. Similar to the configurations of the shallow-water model, the spatial resolution of the Z500 and T850 fields are 128×64 ($2.8125° \times 2.8125°$) with $\Delta t = 1$ hour. The data are collected from 1979 to 2018 (40 years in total).

4.2 network architectures and training process

4.2.1 autoencoder

In previous work [37], the encoder-decoder mappings have been implemented with fully-connected neural networks. Nevertheless, such design choice raises potential scalability issues for our two-dimensional test case, considering the prohibitively large parameter volume. Directly applying fully-connected networks could lead to a parameter volume of roughly $(2 \times 128 \times 64)^2 \times d = \mathcal{O}(10^9)$ for a network of depth d, which is impractical, particularly as this volume grows with the scale of the real deployment.

To make a systematical comparison with the autoencoder-based LA framework and circumvent the scalability concerns, we have opted to leverage convolutional neural networks (CNNs). The decision aligns with the natural suitability of CNNs for tasks that exhibit local spatial or temporal correlations. The fully-connected network in the network structure of [37] has been substituted with the state-of-the-art data compression model named AEflow [21] in the field of turbulent flow simulation.

4.2.2 implicit neural representations

Similar to the approaches in [49], we have selected FourierNet for our INR implementation, which is currently a state-of-the-art architecture within the class pf Multiplicative Filter Network (MFN) [18]. Given a latent vector $\boldsymbol{z} \in \mathcal{Z}$, the system state $\boldsymbol{u} = I(\cdot; \boldsymbol{z}, \boldsymbol{\varphi})$ is decoded as a continuous field. Recall that we denote the coordinate by $\boldsymbol{p} \in \mathcal{X}$, then the evaluation at each point $\boldsymbol{p}$ is given as

$$\begin{aligned} \boldsymbol{z}^{(0)} &= \boldsymbol{s}_0, \\ \boldsymbol{z}^{(l)} &= (\boldsymbol{W}^{(l-1)}\boldsymbol{z}^{(l-1)} + \boldsymbol{b}^{(l-1)} + \boldsymbol{\mu}^{(l-1)}) \odot \boldsymbol{s}_l, \\ \boldsymbol{u}(\boldsymbol{p}) = I(\boldsymbol{p}; \boldsymbol{z}, \boldsymbol{\varphi}) = \boldsymbol{z}^L &= \boldsymbol{W}^{(L-1)}\boldsymbol{z}^{(L-1)} + \boldsymbol{b}^{(L-1)}, \end{aligned}$$

where $\odot$ stands for element-wise multiplication and $\boldsymbol{s}_l$ is defined as the concatenation of $\cos\omega_l \boldsymbol{p}$ and $\sin\omega_l \boldsymbol{p}$. We also introduce a latent shift $\boldsymbol{\mu}^{(l)}$ at each layer, defined as $\boldsymbol{W}'^{(l)}\boldsymbol{z} + \boldsymbol{b}'^{(l)}$ to induce amplitude modulation [16]. The weights and the biases $\boldsymbol{W}^{(l)}$, $\boldsymbol{W}'^{(l)}$, $\boldsymbol{b}^{(l)}$, $\boldsymbol{b}'^{(l)}$, along with the scalar ω_l at each layer, are encapsulated within $\boldsymbol{\varphi}$ as trainable parameters.

4.2.3 surrogate model for latent dynamics

By embedding the physical states into a latent space, the physical dynamics naturally leads to an unknown latent dynamics, which can be written as

$$\frac{\mathrm{d}\boldsymbol{z}}{\mathrm{d}t} = \boldsymbol{F}(\boldsymbol{z}),$$

where we omit the time-dependence of $\boldsymbol{F}$ for simplicity. Generally there are two major kinds of surrogate models for latent dynamics implemented via neural networks. One way is to model the latent dynamics as a discrete-time dynamical system, which is usually evolved with a forward Euler integrator

$$\boldsymbol{z}_{k+1} = \boldsymbol{z}_k + f_{\boldsymbol{\psi}}(\boldsymbol{z}_k)\Delta t =: \mathcal{G}_{\boldsymbol{\psi}}(\boldsymbol{z}_k).$$

The function $f_{\boldsymbol{\psi}}$ is implemented with a neural network and the trainable parameters are denoted by $\boldsymbol{\psi}$. The work of [37] has followed such an approaches and used the ReZero[3] structure to enhance the performance. The other way is to model the latent dynamics as a continuous-time dynamical system

$$\frac{\mathrm{d}\boldsymbol{z}}{\mathrm{d}t} = f_{\boldsymbol{\psi}}(\boldsymbol{z})$$

with the trainable parameter $\boldsymbol{\psi}$, which is suggested in our proposed framework, and we have chosen the Neural ODE structure for the implementation. It's worth noting that such structure are well suited to handle time-series data, as they offer a data-efficient way to parameterize a trajectory over time and naturally handle arbitrary time steps. These advantages make Neural ODEs a particularly fitting choice for modeling the latent dynamics in our LAINR framework.

4.2.4 training process

To show the advantages of our LAINR framework, we choose three different sets of methods for our experiments, which are detailed in Table 3. Note that all the methods are pre-computed (PCA-LinReg) or trained (AEflow-ReZero and INR-NeuralODE) with the same training dataset and then evaluated on a separated testing dataset. Since the performances are usually quantified by the rooted-mean-square error $\mathcal{L}^{\text{skip}}$ on the quasi-uniform skipped latitude-longitude grid G_{skip} [47] or the weighted rooted-mean-square error $\mathcal{L}^{\text{weighted}}$ with reepect to the cosine values of latitudes, formulated as

$$\mathcal{L}^{\text{skip}}(\boldsymbol{x}, \boldsymbol{x}') = \sqrt{\frac{1}{|G \cap G_{\text{skip}}|} \sum_{\boldsymbol{p} \in G \cap G_{\text{skip}}} \|\boldsymbol{x}(\boldsymbol{p}) - \boldsymbol{x}'(\boldsymbol{p})\|^2},$$

$$\mathcal{L}^{\text{weighted}}(\boldsymbol{x}, \boldsymbol{x}') = \sqrt{\frac{1}{\sum_{\boldsymbol{p} \in G} \cos\phi(\boldsymbol{p})} \sum_{\boldsymbol{p} \in G} \|\boldsymbol{x}(\boldsymbol{p}) - \boldsymbol{x}'(\boldsymbol{p})\|^2 \cos\phi(\boldsymbol{p})},$$

where $\phi(\boldsymbol{p})$ is evaluated as the latitude of the position $\boldsymbol{p}$, the networks parameters are trained to minimize the same corresponding discrepancy. No significant differences between these two metrics have been found throughout our experiments (not shown), so we fix the metric for both training and evaluation as the weighted rooted-mean-square error $\mathcal{L}^{\text{weighted}}$ in that it is more flexible when applied to non-regular meshes. The trainable parameters for both the AEflow and the INR encoders are optimized by the Adam optimizer [26], which aim to minimize the reconstruction loss

$$L_{\text{rec}}(\boldsymbol{x}) = \mathcal{L}^{\text{weighted}}(\boldsymbol{x}, \mathcal{D} \circ \mathcal{E}(\boldsymbol{x}))$$

for each snapshot $\boldsymbol{x}$. As for the dynamics learning, the loss function for the ReZero model is defined as the one-step-ahead prediction loss

$$L^{(1)}_{\text{pred}}(\boldsymbol{x}_k) = \mathcal{L}^{\text{weighted}}(\boldsymbol{x}_{k+1}, \mathcal{D} \circ \mathcal{G} \circ \mathcal{E}(\boldsymbol{x}_k)),$$

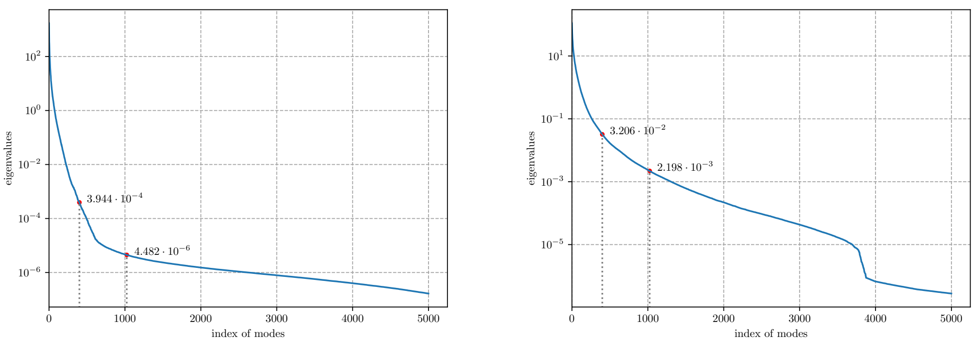
which is the same as that of [37], and the subscript k indicates the index for time steps. On the contrary, for the NeuralODE method, we use an RK4 integrator via TorchDiffEq [11] and apply apply exponential Scheduled Sampling with the multi-step-ahead prediction loss

$$L^{(s)}_{\text{pred}}(\boldsymbol{x}_k) = \mathcal{L}^{\text{weighted}}(\boldsymbol{x}_{k+s}, \mathcal{D} \circ \mathcal{G}^s \circ \mathcal{E}(\boldsymbol{x}_k)),$$

where s is the number of steps ahead with a maximum 10. The learning process for both the encoder-decoder mappings and the latent surrogate models are performed in an alternating manner with distinct learning rates.

method name	encoder/decoder	latent dynamics	latent dimension
PCA-LinReg	PCA (Sec. 2.1)	linear regression	400 or 1024
AEflow-ReZero	AEflow[21]	ReZero[3]	1024
INR-NeuralODE (ours)	FourierNet[18]	Neural ODE[11, 10]	400

Table 3: Three different methods for comparison in the experiments.



(a) shallow-water model

(b) ERA5 data

Figure 2: Distributions of the 5000 largest eigenvalues of the covariance matrix. The 400-th and the 1024-th eigenvalues are marked on the graph.

4.3 Performances

4.3.1 exploring low-dimensional representations

Our experiments start with an exploration of the underlying low-dimensional structure within the datasets, followed by a comparative analysis of the reconstruction capabilities of the three distinct ROMs: PCA, AEflow and INR described above. The assessment has been conducted independently of the surrogated model assigned to handle latent dynamics, which means only the encoder-decoder mappings are computed or trained.

First, after computing the empirical covariance matrix for all the snapshots within the training dataset, we calculate all the eigenvalues and corresponding eigenvectors for the matrix and then sort the eigenvalues in descending order. As illustrated in Figure 2, PCA-based methods are able to reconstruct the physical states for the shallow-water model well as long as more than 1000 dominant modes are kept. However, since real physical states are often much more complex, the ERA5 data cannot be recovered accurately with about or less than 1000 PCA modes. We decide to set the latent dimension for our LAINR framework as 400 (See Table 3), which present a more challenging setup to explore its generalizability.

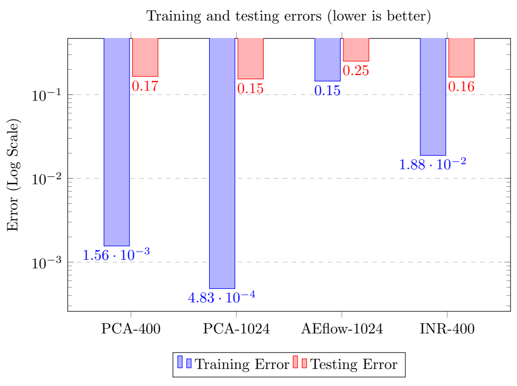
For each method, we have recorded the reconstruction error on the training dataset and subsequently evaluated its generalization ability on a separate testing dataset. We adapt the following averaged reconstruction error

$$L_{\text{rec}} = \frac{1}{N}\sum_{k=1}^{N} \mathcal{L}^{\text{weighted}}(\boldsymbol{x}_k, \mathcal{D} \circ \mathcal{E}(\boldsymbol{x}_k))$$

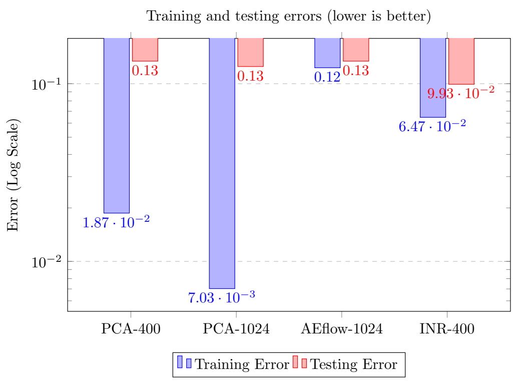
between the system states and the recovered states, where k refers to the index of each snapshot and N is the total amount. The reconstruction errors for both the training and testing processes are summarized in Figure 3, where the numbers of latent dimensions are appended to the method names.

PCA-400 and PCA-1024 uses 400 and 1024 dominant modes, respectively. The lower training and testing errors with PCA-1024, as compared to PCA-400, signify the advantage of including more modes in reconstructing the system state, which is, however, at the cost of increased complexity and computational burden. Additionally, in spite of the low reconstruction errors on the training dataset, the testing errors are almost the same as those of AEflow and INR, which implies the PCA model tends to overfit.

When comparing to AEflow, it's important to note that the method of AEflow uses 1024 dimensions in its latent space. Even with a higher latent dimension, the AEflow method has shown relatively larger reconstruction errors in both training and testing stages, indicating its inefficiency in capturing and restoring the physical features when the latent dimensionality is limited. Contrastingly, our ROM implemented via INR utilises a



(a) shallow-water model



(b) ERA5 data

Figure 3: Comparison of different ROMs

lower-dimensional latent space with only 400 dimensions, which is less than half of that in AEflow and less than the number of modes used in PCA-1024. Despite this reduction in complexity, INR achieved competitive performance for the shallow-water model and even better performance for the ERA5 data than the other methods.

Upon observing the compelling performance of the encoder-decoder mappings implemented via INR, it becomes clear that our choice of employing INR as the ROM for the LA framework is well-justified. INR does not compromise the accuracy of state reconstruction and achieves competitive performance against the larger dimensional PCA-1024 and AEflow models. This not only speaks to the efficiency of INR in capturing the physical features but also points to its potential in managing the computational resources better, which means it is well suited for real-world data assimilation tasks where computational cost is a crucial factor.

4.3.2 embedding the physical dynamics

Embedding a spatio-temporal dynamical system into a latent space involves not only an accurate encoding of the snapshot at each time step, but also the discovery of the latent dynamics, which is even more crucial since the latent repreasentations will be fed into a LA framework. To evaluate the consistency as well as the stability of the surrogate models for latent dynamics, the multi-step prediction error

$$L_{\text{pred}}^{(s)} = \frac{1}{N} \sum_{k=1}^{N} \mathcal{L}^{\text{weighted}}(\boldsymbol{x}_{k+s}, \mathcal{D} \circ \mathcal{G}^s \circ \mathcal{E}(\boldsymbol{x}_k))$$

used for training is employed, where k is the time index for the state $\boldsymbol{x}$ and s indicates the number of predicting steps. The encoder-decoder mappings $\mathcal{E}$ and $\mathcal{D}$ are trained together with the surrogate model $\mathcal{G}$ for the latent dynamics in the training process for all of the DL-based methods. The increment of the prediction error with respect to the number of predicting steps s is recorded in Figure 4.

4.3.3 compatibility (not finished yet)

Intuitively, good performances of DA algorithms rely on the quality of the forward integrator as well as the observation operator, but an accurate surrogate model does not necessarily lead to better results since the model noise is intractable in the LA framework, and the non-linear behavior may be incompatible with the assimilation algorithm. Consequently, we check the compatibility of our proposed methods when combined with existing assimilation algorithms within the LA framework.

We choose several variants of the Kalman filter method for testing, and we would like to emphasize that similar experiments can be done with other assimilation algorithms even including the variational methods. Full algorithms for the asociated Kalman filter methods (EnKF, SEnKF, DEnKF, EnSRKF, ETKF, ETKF-Q) that involve in our experiments are all exhibited in Appendix C.

First, we fix the observation operator as a random sampling on the grid $128 \times 64 \times 2$ with the number of the observed grids $n_{\text{obs}} \in \{1024, 2048, 4096\}$. The observation noise σ^o is set as 0.001, and the model noise σ^m is tuned empirically. The initial condition is set as a perturbation of the ground truth with a Gaussian noise $\sigma^b = 0.1$. See Table 4 and Table 5 for full results of AEflow-ReZero and INR-NeuralODE, respectively.

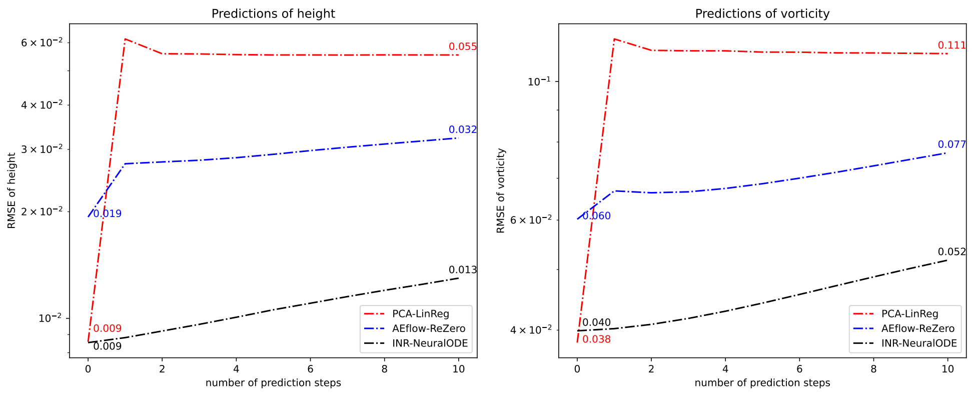


Figure 4: Comparison of multi-step prediction error for different methods on the testing dataset

By fixing the best configurations for the two models, we have plotted the evolution of assimilated system states along with the ground truth and the differences. See Figure 5 and Figure 6 for the evolution, and Figure 7 for the corresponding assimilation error.

5 Conclusion and future work

In this paper, we have proposed a new latent assimilation framewark called Latent Assimilation with Implicit Neural Representations (LAINR), which offers an innovative integration of machine learning techniques and data assimilation concepts. Besides, our LAINR framework opens up new possibilities for processing and assimilating complex data systems by using the reduce-order technique. In comparison to classical reduced-order models (ROMs) and autoencoder-based models, experiments have shown that LAINR surpasses the previously proposed models in terms of not just theoretically appliability but also practical performance.

As a DL-based approach, LAINR employs an encoding model using Implicit Neural Representations (INR) to map the physical state into a lower-dimensional parameterization space. Simultaneously, a surrogate model implemented through Neural Ordinary Differential Equations (Neural ODEs) approximates the latent dynamics. These components together provide a powerful, high-capacity data assimilation framework that transcends many of the limitations of conventional approaches.

Non-linear embedding LAINR's capability for non-linear embedding introduces the possibility that the complex dynamics of physical states is transformed into a much simpler latent dynamics. This feature stands in contrast with classical linear ROMs including the PCA-LinReg method as tested in our experiments, which may inadequently represent complex system dynamics.

Continuous mapping Unlike classical linear ROMs or autoencoder-based models that treat the state field as a discrete grid of state features, LAINR considers the state field as a continuous map, and the special perspective provides a more accurate description of physical features, aiding in the learning of latent representations.

Flexibility of inputs While classical ROMs and autoencoder architectures requires fully provided input data for embedding process, LAINR is able to handle inputs from partially observed or even irregular grids, which becomes more suitable under potential hardware limitations when complete observations are not always guaranteed such as satellite or radar observations, and as a consequence, its utility scope is wider in real-world scenarios.

Scalability LAINR exhibits a clear edge in scalability over previous autoencoder-based models. Since the architecture of LAINR uses coordinates as inputs, it effortlessly accommodates increasing dimensionality and resolution, unlike classical models that encounter increased computational costs due to their reliance on a grid of features as inputs.

Experimental results have demonstrated that our LAINR framework is not only theoretically sound but also practically efficient and effective. It has also been shown that LAINR has superior performance in terms of

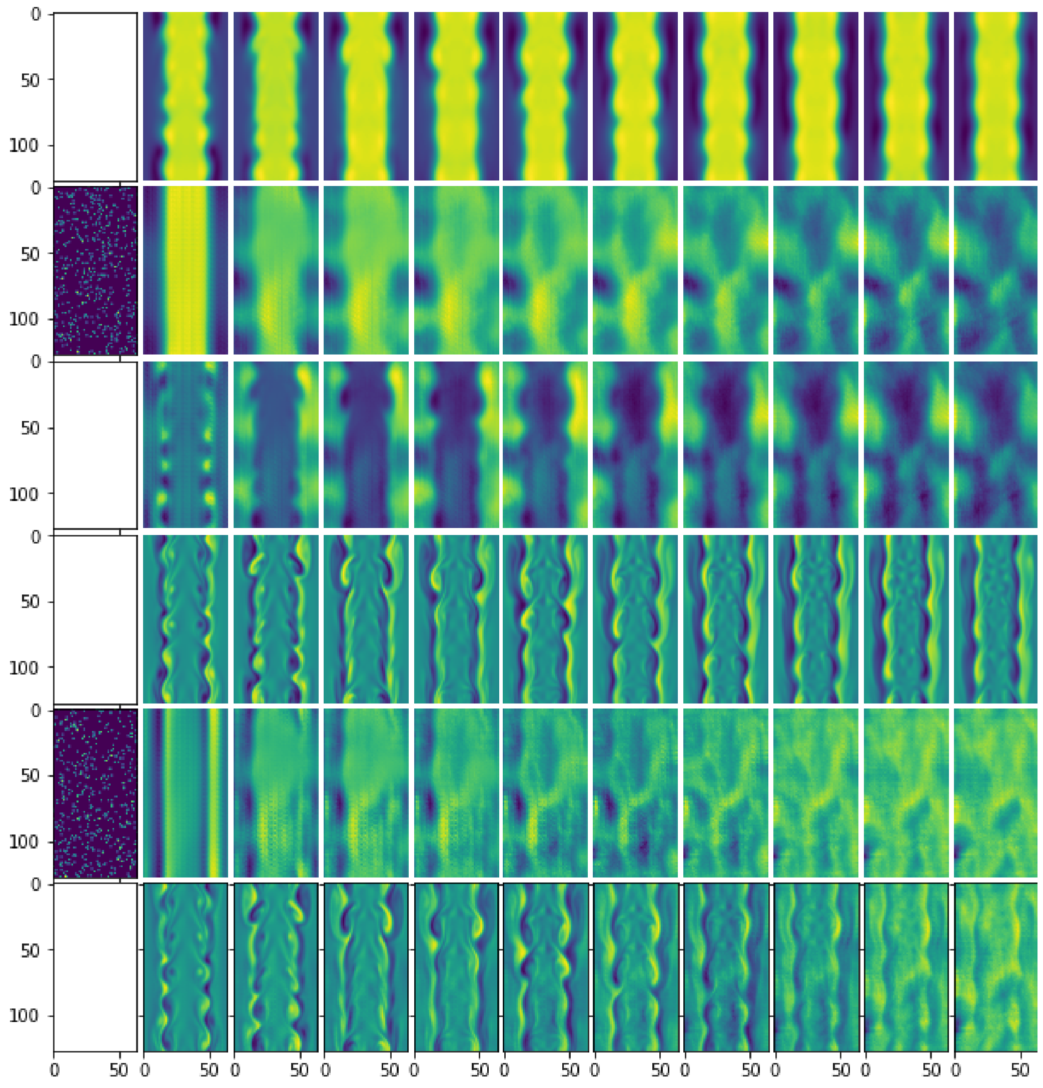


Figure 5: AEflow-ReZero with ETKF-Q, mod_sigma=0.1, $n_{\text{obs}} = 1024$. The first 3 rows correspond to feature 0, while the last 3 rows correspond to feature 1. For each feature, the ground truth, the assimilated field as well as the corresponding differences are displayed sequentially (the frames are recorded every $40\Delta t$).

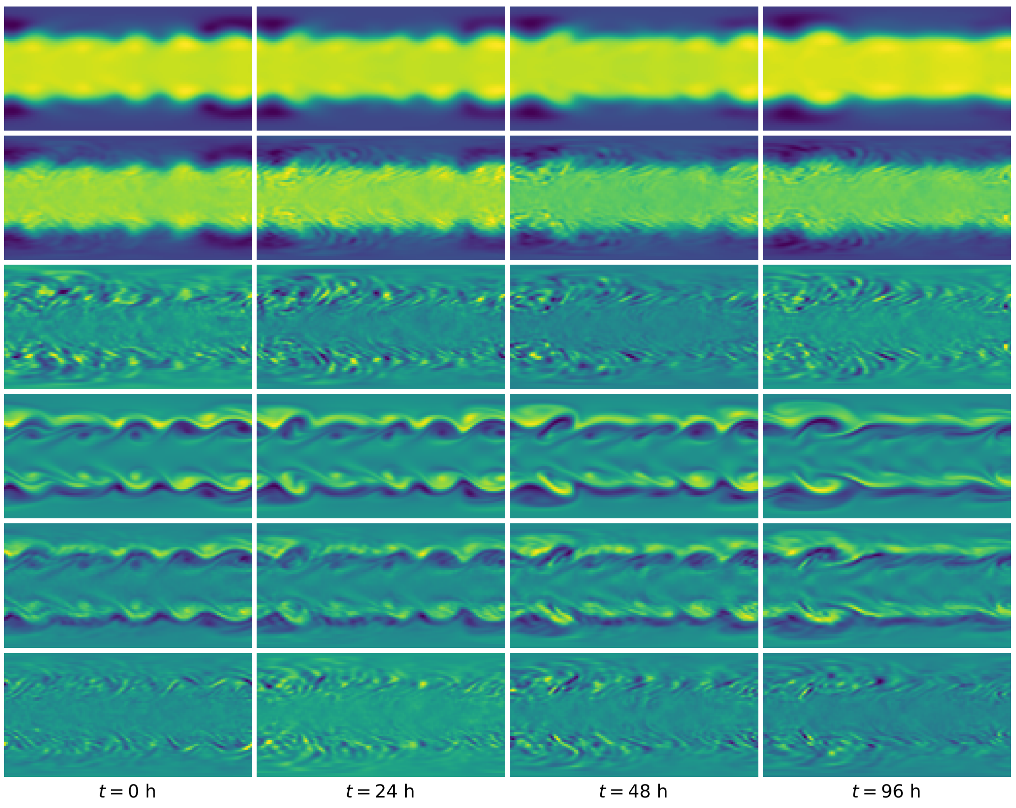


Figure 6: Our model with EnKF, mod_sigma=0.1, $n_{\text{obs}} = 1024$. The first 3 rows and the last 3 rows correspond to the height and the vorticity, respectively. For each feature, the ground truth, the assimilated field as well as the corresponding differences are displayed sequentially.

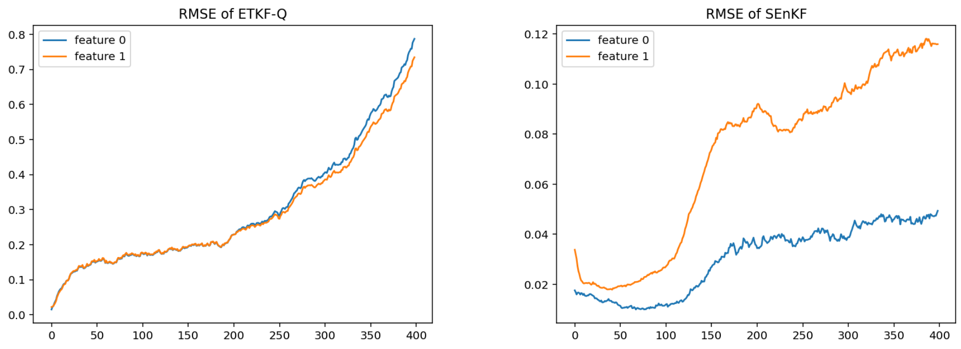


Figure 7: Assimilation error (left: AEflow-ReZero with ETKF-Q, mod_sigma=0.1, $n_{\text{obs}} = 1024$; right: INR-NeuralODE (ours) with SEnKF, mod_sigma=0.05, $n_{\text{obs}} = 1024$)

Table 4: Comparison of performance metrics for different filters for AEflow-ReZero model.

mod_sigma	n_{obs}	EnKF	SEnKF	DEnKF	EnSRKF	ETKF	ETKF-Q
1.0	1024	0.2542	0.2734	0.2764	0.2848	0.3351	0.1936
0.5	1024	0.2468	0.2294	0.2745	0.2626	0.2462	0.1914
0.2	1024	0.2559	0.2247	0.2255	0.2424	0.2163	0.1766
0.1	1024	0.2255	0.2283	0.2084	0.1955	0.2123	0.2082
0.05	1024	0.2490	0.2391	0.2159	0.2067	0.2043	0.2968
0.02	1024	0.2759	0.2784	0.2698	0.2565	0.2493	0.3906
0.01	1024	0.3314	0.3746	0.3073	0.3478	0.3114	0.4379
1.0	2048	0.2180	0.2159	0.2346	0.2236	0.2281	0.1731
0.5	2048	0.2085	0.1958	0.2036	0.2003	0.1960	0.1837
0.2	2048	0.2028	0.2117	0.2023	0.2111	0.2035	0.1978
0.1	2048	0.2107	0.1997	0.2127	0.2448	0.2129	0.1985
0.05	2048	0.2338	0.2346	0.2092	0.2396	0.2658	0.2870
0.02	2048	0.2543	0.2752	0.2369	0.2500	0.2503	0.4051
0.01	2048	0.3246	0.3513	0.2992	0.3331	0.3156	0.4359
1.0	4096	0.1953	0.2105	0.2051	0.1955	0.1963	0.1746
0.5	4096	0.2035	0.1910	0.1911	0.2019	0.2033	0.1948
0.2	4096	0.1881	0.1947	0.1941	0.1889	0.1883	0.1905
0.1	4096	0.2077	0.2025	0.1868	0.1977	0.1966	0.2018
0.05	4096	0.2138	0.2104	0.2132	0.2004	0.2044	0.2766
0.02	4096	0.2459	0.2760	0.2277	0.2359	0.2445	0.4026
0.01	4096	0.3192	0.3583	0.2978	0.3190	0.3141	0.4393

prediction accuracy compared to previous works, which indicates that the LAINR framework is a promising alternative in the realm of data assimilation especially when no accurate forward integrator is available. Its non-linear embedding capability and continuous mapping methodology allow a more accurate depiction and interpretation of complex systems. Its flexibility in handling partial and irregular input data makes it robust for real-world applications, while its scalability makes it highly suitable for managing the burgeoning volumes of data in this era of big data.

Nonetheless, we have to stress that the full potential of LAINR has not yet been expored, and it is also important to acknowledge that areas where further research and development are needed. One such aspect is the testing of LAINR in real datasets such as the ERA5 dataset [23]. The performance and robustness of our LAINR framework under real-world conditions would provide further validation of its effectiveness. Additionally, more complicated observation operators need to be incorporated into the experiments since the real observations tends to be highly non-linear such as the solution of radiative-transfer models. Moreover, future work should explore the design of more advanced architectures for INR as well. As the model error is tuned empirically at present, new structures considering the embedding error and the prediction error are likely to provide a more comprehensive and accurate model.

In conclusion, LAINR brings a heightened level of intricacy, scalability, and flexibility to the handling of complex systems. As we continue to traverse the era of big data and complex systems, trailblazing frameworks like LAINR will be indispensable in augmenting our predictive capabilities and understanding of complex systems. With its full potential yet to be explored, LAINR promises to catalyze a paradigm shift in our interaction with and comprehension of complex systems.

References

[1] Troy Arcomano, Istvan Szunyogh, Jaideep Pathak, Alexander Wikner, Brian R. Hunt, and Edward Ott. A machine learning-based global atmospheric forecast model. *Geophysical Research Letters*, 47(9):e2020GL087776, 2020.

[2] Mark Asch, Marc Bocquet, and Malle Nodet. *Data Assimilation.* Society for Industrial and Applied Mathematics, Philadelphia, PA, 2016.

Table 5: Comparison of performance metrics for different filters for INR-NeuralODE (ours).

mod_sigma	n_{obs}	EnKF	SEnKF	DEnKF	EnSRKF	ETKF	ETKF-Q
0.1	1024	0.0619	0.0606	0.0645	0.0585	nan	0.1164
0.05	1024	0.0708	$\underline{0.0575}$	0.0598	0.0628	0.0642	0.1290
0.02	1024	0.0590	0.0598	0.0681	0.0600	0.0615	0.1259
0.01	1024	0.0578	0.0587	0.0638	0.0581	0.0606	0.1129
0.005	1024	0.0645	0.0808	0.0686	0.0629	0.0633	0.1098
0.002	1024	0.1342	0.4105	0.0873	0.1490	0.1147	0.1436
0.001	1024	4.3256	1.5767	5.8916	2.9570	29.9085	0.4060
0.1	2048	0.0454	0.0465	0.0498	0.0454	nan	0.1252
0.05	2048	0.0442	0.0440	0.0466	0.0439	0.0445	0.1108
0.02	2048	0.0443	$\underline{0.0437}$	0.0463	0.0445	0.0443	0.1156
0.01	2048	0.0459	0.0449	0.0466	0.0445	0.0448	0.1275
0.005	2048	0.0518	0.0544	0.0496	0.0505	0.0501	0.1141
0.002	2048	0.0656	0.0802	0.0623	0.0751	0.0765	0.1337
0.001	2048	0.3156	1.0958	0.1431	0.2924	0.1659	0.1936
0.1	4096	0.0412	0.0414	0.0433	0.0418	nan	0.1413
0.05	4096	0.0411	0.0405	0.0420	$\underline{0.0400}$	nan	0.1231
0.02	4096	0.0417	0.0420	0.0438	0.0412	0.0413	0.1145
0.01	4096	0.0416	0.0421	0.0424	0.0417	0.0416	0.1144
0.005	4096	0.0448	0.0530	0.0482	0.0467	0.0468	0.1074
0.002	4096	0.0694	0.1119	0.0637	0.0679	0.0731	0.1269
0.001	4096	0.1392	0.4200	0.1451	0.2540	0.2150	0.2179

[3] Thomas Bachlechner, Bodhisattwa Prasad Majumder, Henry Mao, Gary Cottrell, and Julian McAuley. Rezero is all you need: Fast convergence at large depth. In *Uncertainty in Artificial Intelligence*, pages 1352–1361. PMLR, 2021.

[4] Mojtaba Bemana, Karol Myszkowski, Hans-Peter Seidel, and Tobias Ritschel. X-fields: Implicit neural view-, light- and time-image interpolation. *ACM Transactions on Graphics (Proc. SIGGRAPH Asia 2020)*, 39(6), 2020.

[5] Kaifeng Bi, Lingxi Xie, Hengheng Zhang, Xin Chen, Xiaotao Gu, and Qi Tian. Pangu-weather: A 3d high-resolution model for fast and accurate global weather forecast. *arXiv preprint arXiv:2211.02556*, 2022.

[6] Steven L. Brunton, Joshua L. Proctor, and J. Nathan Kutz. Discovering governing equations from data by sparse identification of nonlinear dynamical systems. *Proceedings of the National Academy of Sciences*, 113(15):3932–3937, 2016.

[7] Keaton J. Burns, Geoffrey M. Vasil, Jeffrey S. Oishi, Daniel Lecoanet, and Benjamin P. Brown. Dedalus: A flexible framework for numerical simulations with spectral methods. *Phys. Rev. Res.*, 2:023068, Apr 2020.

[8] Kang Chen, Tao Han, Junchao Gong, Lei Bai, Fenghua Ling, Jing-Jia Luo, Xi Chen, Leiming Ma, Tianning Zhang, Rui Su, et al. Fengwu: Pushing the skillful global medium-range weather forecast beyond 10 days lead. *arXiv preprint arXiv:2304.02948*, 2023.

[9] Peter Yichen Chen, Jinxu Xiang, Dong Heon Cho, Yue Chang, G A Pershing, Henrique Teles Maia, Maurizio M Chiaramonte, Kevin Thomas Carlberg, and Eitan Grinspun. CROM: Continuous reduced-order modeling of PDEs using implicit neural representations. In *The Eleventh International Conference on Learning Representations*, 2023.

[10] Ricky T. Q. Chen, Brandon Amos, and Maximilian Nickel. Learning neural event functions for ordinary differential equations. *International Conference on Learning Representations*, 2021.

[11] Ricky TQ Chen, Yulia Rubanova, Jesse Bettencourt, and David K Duvenaud. Neural ordinary differential equations. *Advances in neural information processing systems*, 31, 2018.

[12] Zhiqin Chen and Hao Zhang. Learning implicit fields for generative shape modeling. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5939–5948, 2019.

[13] Sibo Cheng, Jianhua Chen, Charitos Anastasiou, Panagiota Angeli, Omar K. Matar, Yi-Ke Guo, Christopher C. Pain, and Rossella Arcucci. Generalised latent assimilation in heterogeneous reduced spaces with machine learning surrogate models. *J. Sci. Comput.*, 94(1), jan 2023.

[14] Junyoung Chung, Çaglar Gülçehre, KyungHyun Cho, and Yoshua Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling. *CoRR*, abs/1412.3555, 2014.

[15] Mariana C.A. Clare, Omar Jamil, and Cyril J. Morcrette. Combining distribution-based neural networks to predict weather forecast probabilities. *Quarterly Journal of the Royal Meteorological Society*, 147(741):4337–4357, 2021.

[16] Emilien Dupont, Hyunjik Kim, S. M. Ali Eslami, Danilo Jimenez Rezende, and Dan Rosenbaum. From data to functa: Your data point is a function and you can treat it like one. In Kamalika Chaudhuri, Stefanie Jegelka, Le Song, Csaba Szepesvari, Gang Niu, and Sivan Sabato, editors, *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pages 5694–5725. PMLR, 17–23 Jul 2022.

[17] Emilien Dupont, Hrushikesh Loya, Milad Alizadeh, Adam Goliński, Yee Whye Teh, and Arnaud Doucet. Coin++: Data agnostic neural compression. *arXiv preprint arXiv:2201.12904*, 2022.

[18] Rizal Fathony, Anit Kumar Sahu, Devin Willmott, and J Zico Kolter. Multiplicative filter networks. In *International Conference on Learning Representations*, 2021.

[19] Anthony Fillion, Marc Bocquet, Serge Gratton, Selime Gürol, and Pavel Sakov. An iterative ensemble kalman smoother in presence of additive model error. *SIAM/ASA Journal on Uncertainty Quantification*, 8(1):198–228, 2020.

[20] Joseph Galewsky, Richard K. Scott, and Lorenzo M. Polvani. An initial-value problem for testing numerical models of the global shallow-water equations. *Tellus A: Dynamic Meteorology and Oceanography*, Jan 2004.

[21] Andrew Glaws, Ryan King, and Michael Sprague. Deep learning for in situ data compression of large turbulent flow simulations. *Phys. Rev. Fluids*, 5:114602, Nov 2020.

[22] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *arXiv preprint arXiv:1512.03385*, 2015.

[23] Hans Hersbach, Bill Bell, Paul Berrisford, Shoji Hirahara, Andrs Hornyi, Joaqun Muoz-Sabater, Julien Nicolas, Carole Peubey, Raluca Radu, Dinand Schepers, Adrian Simmons, Cornel Soci, Saleh Abdalla, Xavier Abellan, Gianpaolo Balsamo, Peter Bechtold, Gionata Biavati, Jean Bidlot, Massimo Bonavita, Giovanna De Chiara, Per Dahlgren, Dick Dee, Michail Diamantakis, Rossana Dragani, Johannes Flemming, Richard Forbes, Manuel Fuentes, Alan Geer, Leo Haimberger, Sean Healy, Robin J. Hogan, Elas Hlm, Marta Janiskov, Sarah Keeley, Patrick Laloyaux, Philippe Lopez, Cristina Lupu, Gabor Radnoti, Patricia de Rosnay, Iryna Rozum, Freja Vamborg, Sebastien Villaume, and Jean-Nol Thpaut. The era5 global reanalysis. *Quarterly Journal of the Royal Meteorological Society*, 146(730):1999–2049, 2020.

[24] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Comput.*, 9(8):1735–1780, nov 1997.

[25] Chiyu "Max" Jiang, Soheil Esmaeilzadeh, Kamyar Azizzadenesheli, Karthik Kashinath, Mustafa Mustafa, Hamdi A. Tchelepi, Philip Marcus, Prabhat, and Anima Anandkumar. Meshfreeflownet: A physics-constrained deep continuous space-time super-resolution framework. In *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, SC '20. IEEE Press, 2020.

[26] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[27] Remi Lam, Alvaro Sanchez-Gonzalez, Matthew Willson, Peter Wirnsberger, Meire Fortunato, Alexander Pritzel, Suman Ravuri, Timo Ewalds, Ferran Alet, Zach Eaton-Rosen, et al. Graphcast: Learning skillful medium-range global weather forecasting. *arXiv preprint arXiv:2212.12794*, 2022.

[28] Zongyi Li, Nikola Kovachki, Kamyar Azizzadenesheli, Burigede Liu, Kaushik Bhattacharya, Andrew Stuart, and Anima Anandkumar. Fourier neural operator for parametric partial differential equations, 2020.

[29] Zichao Long, Yiping Lu, and Bin Dong. Pde-net 2.0: Learning pdes from data with a numeric-symbolic hybrid deep network. *Journal of Computational Physics*, 399:108925, 2019.

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

[31] Mantas Lukoeviius and Herbert Jaeger. Reservoir computing approaches to recurrent neural network training. *Computer Science Review*, 3(3):127–149, 2009.

[32] Nicholas J. Lutsko, Isaac M. Held, and Pablo Zurita-Gotor. Applying the fluctuation-dissipation theorem to a two-layer model of quasigeostrophic turbulence. *Journal of the Atmospheric Sciences*, 72(8):3161 – 3177, 2015.

[33] Jeong Joon Park, Peter Florence, Julian Straub, Richard Newcombe, and Steven Lovegrove. Deepsdf: Learning continuous signed distance functions for shape representation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 165–174, 2019.

[34] Jaideep Pathak, Shashank Subramanian, Peter Harrington, Sanjeev Raja, Ashesh Chattopadhyay, Morteza Mardani, Thorsten Kurth, David Hall, Zongyi Li, Kamyar Azizzadenesheli, Pedram Hassanzadeh, Karthik Kashinath, and Animashree Anandkumar. Fourcastnet: A global data-driven high-resolution weather model using adaptive fourier neural operators. *arXiv preprint arXiv:2202.11214*, 2022.

[35] Suraj Pawar and Omer San. Equation-free surrogate modeling of geophysical flows at the intersection of machine learning and data assimilation. *Journal of Advances in Modeling Earth Systems*, 14(11):e2022MS003170, 2022.

[36] S. G. Penny, T. A. Smith, T.-C. Chen, J. A. Platt, H.-Y. Lin, M. Goodliff, and H. D. I. Abarbanel. Integrating recurrent neural networks with data assimilation for scalable data-driven state estimation. *Journal of Advances in Modeling Earth Systems*, 14(3):e2021MS002843, 2022.

[37] Mathis Peyron, Anthony Fillion, Selime Grol, Victor Marchais, Serge Gratton, Pierre Boudier, and Gael Goret. Latent space data assimilation by using deep learning. *Quarterly Journal of the Royal Meteorological Society*, 147(740):3759–3777, 2021.

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

[39] Stephan Rasp, Peter D. Dueben, Sebastian Scher, Jonathan A. Weyn, Soukayna Mouatadid, and Nils Thuerey. Weatherbench: A benchmark data set for data-driven weather forecasting. *Journal of Advances in Modeling Earth Systems*, 12(11):e2020MS002203, 2020. e2020MS002203 10.1029/2020MS002203.

[40] Stephan Rasp and Nils Thuerey. Data-driven medium-range weather prediction with a resnet pretrained on climate simulations: A new model for weatherbench. *Journal of Advances in Modeling Earth Systems*, 13(2):e2020MS002405, 2021. e2020MS002405 2020MS002405.

[41] Sebastian Scher and Gabriele Messori. Ensemble methods for neural network-based weather forecasts. *Journal of Advances in Modeling Earth Systems*, 13(2), 2021.

[42] M. G. Schultz, C. Betancourt, B. Gong, F. Kleinert, M. Langguth, L. H. Leufen, A. Mozaffari, and S. Stadtler. Can deep learning beat numerical weather prediction? *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 379(2194):20200097, 2021.

[43] Justin Sirignano and Konstantinos Spiliopoulos. Dgm: A deep learning algorithm for solving partial differential equations. *Journal of Computational Physics*, 375:1339–1364, 2018.

[44] Vincent Sitzmann, Julien N.P. Martel, Alexander W. Bergman, David B. Lindell, and Gordon Wetzstein. Implicit neural representations with periodic activation functions. In *arXiv*, 2020.

[45] Silviu-Marian Udrescu and Max Tegmark. Ai feynman: A physics-inspired method for symbolic regression. *Science Advances*, 6(16):eaay2631, 2020.

[46] E. Weinan and Bing Yu. The deep ritz method: A deep learning-based numerical algorithm for solving variational problems. *Communications in Mathematics and Statistics*, 6(1):1–12, March 2018.

[47] Hilary Weller, John Thuburn, and Colin J. Cotter. Computational modes and grid imprinting on five quasi-uniform spherical c grids. *Monthly Weather Review*, 140(8):2734 – 2755, 2012.

[48] Jonathan A. Weyn, Dale R. Durran, and Rich Caruana. Improving data-driven global weather prediction using deep convolutional neural networks on a cubed sphere. *Journal of Advances in Modeling Earth Systems*, 12(9):e2020MS002109, 2020. e2020MS002109 10.1029/2020MS002109.

[49] Yuan Yin, Matthieu Kirchmeyer, Jean-Yves Franceschi, Alain Rakotomamonjy, and Patrick Gallinari. Continuous pde dynamics forecasting with implicit neural representations. *arXiv preprint arXiv:2209.14855*, 2022.

A Mathematical preliminaries

Proposition A.1. *Given $\{x_k\}_{k=1}^K \subseteq \mathbb{R}^n$, the matrix consists of the first m ($m < n$) left singular vectors of the matrix $X = (x_1, \cdots, x_K) \in \mathbb{R}^{n\times K}$ solves the following minimization problem:*

$$\min_A \sum_{k=1}^K \left\|x_k - A^\top A x_k\right\|^2 = \min_A \left\|(I - A^\top A)X\right\|_F^2 \text{ subject to } AA^\top = I_m$$

for $A \in \mathbb{R}^{m\times n}$, where $\|\cdot\|_F$ is the Frobenius norm.

Proof. For any $A \in \mathbb{R}^{m\times n}$ satisfying $AA^\top = I_m$ we have

$$\begin{aligned}\sum_{k=1}^K \left\|x_k - A^\top A x_k\right\|^2 &= \sum_{k=1}^K \|x_k\|^2 - 2x_k^\top A^\top A x_k + \left\|A^\top A x_k\right\|^2 \\ &= \sum_{k=1}^K \|x_k\|^2 - 2x_k^\top A^\top A x_k + x_k^\top A^\top A A^\top A x_k \\ &= \sum_{k=1}^K \|x_k\|^2 - x_k^\top A^\top A x_k = \sum_{k=1}^K \|x_k\|^2 - \|Ax_k\|^2.\end{aligned}$$

Hence the minimization problem is equivalent to the maximization problem

$$\max_A \sum_{k=1}^K \|Ax_k\|^2 = \max_A \|AX\|_F^2 \text{ subject to } AA^\top = I_m.$$

Take the singular-value decomposition (SVD) for the matrix X as $X = U\Sigma V^\top$, where $U \in \mathbb{R}^{n\times n}$ and $V \in \mathbb{R}^{K\times K}$ are orthogonal matrices and $\Sigma = \operatorname{diag}(\sigma_1, \cdots, \sigma_s)$ with $s = \min(n, K)$ and $\sigma_1 \geq \cdots \geq \sigma_s \geq 0$. It follows that

$$\|AX\|_F^2 = \operatorname{Tr}\left(AX(AX)^\top\right) = \operatorname{Tr}\left(AU\Sigma V^\top V \Sigma^\top U A^\top\right) = \operatorname{Tr}\left(AU\Sigma\Sigma^\top U A^\top\right) = \sum_{l=1}^s \sigma_l^2 \|Au_l\|^2,$$

where u_l is the l-th column of U, which is also the left singular vector corresponding to the singular value σ_l. Note that

$$\sum_{l=1}^s \|Au_l\|^2 \leq \|AU\|_F^2 = \operatorname{Tr}(AUU^\top A^\top) = \operatorname{Tr}(AA^\top) = \|A\|_F^2 = m.$$

Meanwhile, let $B \in \mathbb{R}^{(n-m)\times n}$ such that $(A^\top, B^\top)$ is orthogonal, then for each l,

$$\|Au_l\|^2 = u_l^\top A^\top A u_l = u_l(I_n - B^\top B)u_l = \|u_l\|^2 - \|Bu_l\|^2 \leq \|u_l\|^2 = 1.$$

Therefore,

$$\begin{aligned}
\sum_{l=1}^{s} \sigma_l \|Au_l\|^2 &= \sum_{l=1}^{m} \sigma_l^2 + \sum_{l=1}^{m} \sigma_l^2 \left(\|Au_l\|^2 - 1\right) + \sum_{l=m+1}^{s} \sigma_l^2 \|Au_l\|^2 \\
&\le \sum_{l=1}^{m} \sigma_l^2 + \sigma_m^2 \sum_{l=1}^{m} \left(\|Au_l\|^2 - 1\right) + \sigma_m^2 \sum_{l=m+1}^{s} \|Au_l\|^2 \\
&= \sum_{l=1}^{m} \sigma_l^2 + \sigma_m^2 \left(\sum_{l=1}^{s} \|Au_l\|^2 - m\right) \le \sum_{l=1}^{m} \sigma_l^2.
\end{aligned}$$

Additionally, it is clear that all the equalities hold if $A = (u_1, \cdots, u_m)^\mathsf{T}$. □

B Detailed configurations for the experiments

Most of the configurations are borrowed from [20] and [32] for shallow-water equations and the QG-model, respectively. We restate them here together with dataset configurations for completeness. Note that all the fixed parameters are of SI units (the International System of Units) and thus omitted for simplicity.

B.1 shallow-water equations

The spherical shallow-water equations on the 2D sphere read

$$\begin{aligned}
\frac{\mathrm{d}\boldsymbol{u}}{\mathrm{d}t} &= -f\boldsymbol{k} \times \boldsymbol{u} - g\nabla h + \nu \Delta \boldsymbol{u}, \\
\frac{\mathrm{d}h}{\mathrm{d}t} &= -h\nabla \cdot \boldsymbol{u} + \nu \Delta h,
\end{aligned}$$

where $\boldsymbol{u} = (u, v)^\mathsf{T}$ is the 2D velocity field tangent to the sphere surface, and h is the thickness of the fluid layer, both of which are defined on a channel domain $(\lambda, \phi) \in [0, 2\pi] \times [-\pi/2, \pi/2]$. The Coriolis parameter $f = 2\Omega \sin\phi$ with the angular velocity $\Omega = 7.292 \times 10^{-5}$ and the gravity $g = 9.80616$ are fixed in consistency with the real earth surface. The viscosity coefficient $\nu = \frac{10^5 \times 3600^2}{a^2 \times 32^2} \approx 3.1 \times 10^{-7}$ with the earth radius $a = 6.37122 \times 10^6$ is chosen to be small enough to avoid damping the physical features. The initial conditions for $\boldsymbol{u}$ are given by

$$v|_{t=0} = 0, \quad u|_{t=0} = \begin{cases} \frac{u_m}{e_n} \exp\left(\frac{1}{(\phi-\phi_0)(\phi-\phi_1)}\right), & \phi_0 < |\phi| < \phi_1, \\ 0 & \text{otherwise,} \end{cases}$$

which is slightly modified from [20] to create symmetric fields, where $\phi_0 = \pi/7$ and $\phi_1 = \pi/2 - \phi_0$ gives the boundary of the jet. $e_n = \exp(-4/(\phi_1 - \phi_0)^2)$ is the normalizer so that the maximal zonal wind u_m lies at the jet's mid-point. The thickness h is initialized as a balanced height field, which is obtained by integrating the balance equation

$$gh(\phi) = gh_0 - \int^{\phi} au(\phi') \left(f + a^{-1}u(\phi') \tan\phi'\right) \mathrm{d}\phi'.$$

The constant h_0 is chosen so that the global mean of h is 10^4. The balanced field is then perturbed by adding a localized bump

$$h'(\lambda, \phi) = \hat{h} \exp(-(\lambda/\alpha)^2 - ((\phi_2 - \phi)/\beta)^2) \cos\phi,$$

where $\hat{h} = 120$, $\alpha = 1/3$, $\beta = 1/15$, and $\phi_2 = \pi/4$.

The datasets for both training and testing are generated via the Dedalus project [7] based on Python language within a spectral framework. The maximum velocity u_m is chosen from $\{60, 61, \cdots, 80\}$ to generate distinct 21 trajectories for separation of training and testing. Each trajectory has been recorded from 300 hours to 700 hours with 1-hour interval to capture rich physical features, and the spatial resolution is set as 128×64. The vorticity $w = \nabla \times \boldsymbol{u}$ and the thickness h are the two features for network training and assimilation.

C Algorithms for associated Kalman filter methods

C.1 EnKF

The basic Ensenble Kalman Filter (EnKF)

C.1.1 Forecast step

- Forward propagation:
$$x_k^{b,j} = \mathcal{M}_k(x_{k-1}^{a,j}) + \varepsilon_k^{M,j}, \quad \varepsilon_k^{M,j} \sim \mathcal{N}(0, \Sigma_k^M);$$
perturbed propagation

C.1.2 Analysis step

- Obtain the ensemble mean and anomaly matrix from the ensembles:
$$\begin{aligned} x_k^b &= \frac{1}{N}\sum_{j=1}^N x_k^{b,j}, \\ X_k^b &= \frac{1}{\sqrt{N-1}}(x_k^{b,j} - x_k^b)_{j=1}^N \in \mathbb{R}^{n\times N}, \\ P_k^b &= X_k^b X_k^{b\top} \end{aligned}$$
- Obtain the perturbed observation vectors:
$$z_k^{o,j} = y_k^o + \varepsilon_k^{o,j}, \quad \varepsilon_k^{o,j} \sim \mathcal{N}(0, \Sigma_k^o);$$
- Innovation vectors
$$d_k^j = z_k^{o,j} - \mathcal{H}_k(x_k^{b,j});$$
- Evaluate the Kalman gain matrix
$$\begin{aligned} K_k &= P_k^b H_k^\top (H_k P_k^b H_k^\top + \Sigma_k^o)^{-1} \\ &= X_k^b X_k^{b\top} H_k^\top (H_k X_k^b X_k^{b\top} H_k^\top + \Sigma_k^o)^{-1} \\ &= P_{xz}(P_{zz} + \Sigma_k^o)^{-1}, \end{aligned}$$
where
$$P_{xz} = X_k^b (H_k X_k^b)^\top, \quad P_{zz} = (H_k X_k^b)(H_k X_k^b)^\top.$$
- Update the analyzed ensembles:
$$x_k^{a,j} = x_k^{b,j} + K_k d_k^j.$$

C.2 SEnKF

Stochastic EnKF (Algorithm 6.3 in [2])

C.2.1 Forecast step

- Forward propagation:
$$x_k^{b,j} = \mathcal{M}_k(x_{k-1}^{a,j}) + \varepsilon_k^{M,j}, \quad \varepsilon_k^{M,j} \sim \mathcal{N}(0, \Sigma_k^M);$$
perturbed propagation

C.2.2 Analysis step

- Obtain the ensemble mean and anomaly matrix from the ensembles:
$$\begin{aligned} x_k^b &= \frac{1}{N}\sum_{j=1}^N x_k^{b,j}, \\ X_k^b &= \frac{1}{\sqrt{N-1}}(x_k^{b,j} - x_k^b)_{j=1}^N \in \mathbb{R}^{n\times N}, \\ P_k^b &= X_k^b X_k^{b\top} \end{aligned}$$

- Obtain the perturbed observation vectors:

$$z_k^{o,j} = y_k^o + \varepsilon_k^{o,j}, \quad \varepsilon_k^{o,j} \sim \mathcal{N}(0, \Sigma_k^o);$$

- Compute the ensemble means and the normalized anomalies for the perturbed background estimates for the observations:

$$\begin{aligned} y_k^{p,j} &= \mathcal{H}_k(x_k^{b,j}) - \varepsilon_k^{o,j} \\ y_k^p &= \frac{1}{N}\sum_{j=1}^N y_k^{p,j}, \\ Y_k^p &= \frac{1}{\sqrt{N-1}}(y_k^{p,j} - y_k^p)_{j=1}^N \in \mathbb{R}^{n\times N}. \end{aligned}$$

- Kalman gain matrix

$$K_k = X_k^b Y_k^{pT}(Y_k^p Y_k^{pT})^{-1}$$

- Update the analyzed ensembles:

$$x_k^{a,j} = x_k^{b,j} + K_k(z_k^{o,j} - \mathcal{H}_k(x_k^{b,j})).$$

C.3 DEnKF

EnSRKF (See (6.25) in [2])

C.3.1 Forecast step

- Forward propagation:

$$x_k^{b,j} = \mathcal{M}_k(x_{k-1}^{a,j}) + \varepsilon_k^{M,j}, \quad \varepsilon_k^{M,j} \sim \mathcal{N}(0, \Sigma_k^M);$$

  perturbed propagation

C.3.2 Analysis step

- Obtain the ensemble mean and anomaly matrix from the ensembles:

$$\begin{aligned} x_k^b &= \frac{1}{N}\sum_{j=1}^N x_k^{b,j}, \\ X_k^b &= \frac{1}{\sqrt{N-1}}(x_k^{b,j} - x_k^b)_{j=1}^N \in \mathbb{R}^{n\times N}, \\ P_k^b &= X_k^b X_k^{b\intercal} \end{aligned}$$

- Innovation vectors

$$d_k = y_k^o - \mathcal{H}_k(x_k^b);$$

- Kalman gain matrix

$$\begin{aligned} K_k &= P_k^b H_k^\intercal (H_k P_k^b H_k^\intercal + \Sigma_k^o)^{-1} \\ &= X_k^b X_k^{b\intercal} H_k^\intercal (H_k X_k^b X_k^{b\intercal} H_k^\intercal + \Sigma_k^o)^{-1} \\ &= P_{xz}(P_{zz} + \Sigma_k^o)^{-1}, \end{aligned}$$

- Assimilate the forecast state estimate with the observation

$$x_k^a = x_k^b + K_k(y_k^o - \mathcal{H}_k(x_k^b))$$

- Compute the analyzed anomalies

$$X_k^a = X_k - \frac{1}{2}K_k H_k X_k$$

- Compute the analyzed ensembles

$$x_k^{a,j} = \sqrt{N-1}X_k^a[:, j] + x_k^a$$

C.4 EnSRKF

EnSRKF (See (6.21) in [2])

C.4.1 Forecast step

- Forward propagation:
$$x_k^{b,j} = \mathcal{M}_k(x_{k-1}^{a,j}) + \varepsilon_k^{M,j}, \quad \varepsilon_k^{M,j} \sim \mathcal{N}(0, \Sigma_k^M);$$
perturbed propagation

C.4.2 Analysis step

- Obtain the ensemble mean and anomaly matrix from the ensembles:
$$\begin{aligned} x_k^b &= \frac{1}{N}\sum_{j=1}^N x_k^{b,j}, \\ X_k^b &= \frac{1}{\sqrt{N-1}}(x_k^{b,j} - x_k^b)_{j=1}^N \in \mathbb{R}^{n\times N}, \\ P_k^b &= X_k^b X_k^{b\top} \end{aligned}$$
- Compute the ensemble means and the normalized anomalies for the observations:
$$\begin{aligned} y_k^{b,j} &= \mathcal{H}_k(x_k^{b,j}) \\ y_k^b &= \frac{1}{N}\sum_{j=1}^N y_k^{b,j}, \\ Y_k^b &= \frac{1}{\sqrt{N-1}}(y_k^{b,j} - y_k^b)_{j=1}^N \in \mathbb{R}^{m\times N}; \end{aligned}$$
- Transforming matrix
$$T_k = (I_m + Y_k^{b\top}(\Sigma_k^o)^{-1}Y_k^b)^{-1} = (I_m + S_k^\top S_k)^{-1}$$
for $S_k = (\Sigma_k^o)^{-1/2}Y_k^b$;
- normalized innovation vector
$$\delta_k = (\Sigma_k^o)^{-1/2}(y_k^o - y_k^b)$$
- Evaluate the original Kalman gain matrix
$$\begin{aligned} K_k &= P_k^b H_k^\top (H_k P_k^b H_k^\top + \Sigma_k^o)^{-1} \\ &= X_k^b X_k^{b\top} H_k^\top (H_k X_k^b X_k^{b\top} H_k^\top + \Sigma_k^o)^{-1} \\ &= P_{xz}(P_{zz} + \Sigma_k^o)^{-1}, \end{aligned}$$
where
$$P_{xz} = X_k^b (H_k X_k^b)^\top, \quad P_{zz} = (H_k X_k^b)(H_k X_k^b)^\top.$$
- Obtain the modified Kalman gain matrix
$$\tilde{K} = K(I + (I + H_k P_k^b H_k^\top (\Sigma_k^o)^{-1})^{-1/2})^{-1}$$
- Update the analyzed ensembles
$$\begin{aligned} w_k^a &= (I_m + Y_k^{b\top}\Sigma_k^o Y_b)^{-1} Y^{b\top}(\Sigma_k^o)^{-1}(y_k^o - y_k^b) = T_k S_k^\top \delta_k \\ x_k^a &= x_k^b + X_k^b w_k^a = x_k^b + X_k^b T_k S_k^\top \delta_k \\ X_k^a &= (I_n - \tilde{K}_k H_k) X_k^b U, \, U1 = 1, U \in O(N) \\ x_k^{a,j} &= x_k^a + \sqrt{N-1} X_k^a[:, j] \end{aligned}$$
OR:
$$(x_k^{a,j})_j = x_k^b 1_N^\top + \sqrt{N-1} X_k^a$$

C.5 ETKF

ETKF (Algorithm 6.4 in [2])

C.5.1 Forecast step

- Forward propagation:
$$x_k^{b,j} = \mathcal{M}_k(x_{k-1}^{a,j}) + \varepsilon_k^{M,j}, \quad \varepsilon_k^{M,j} \sim \mathcal{N}(0, \Sigma_k^M);$$
  perturbed propagation

C.5.2 Analysis step

- Obtain the ensemble mean and anomaly matrix from the ensembles:
$$\begin{aligned} x_k^b &= \frac{1}{N}\sum_{j=1}^N x_k^{b,j}, \\ X_k^b &= \frac{1}{\sqrt{N-1}}(x_k^{b,j} - x_k^b)_{j=1}^N \in \mathbb{R}^{n\times N}, \\ P_k^b &= X_k^b X_k^{b\mathsf{T}} \end{aligned}$$

- Compute the ensemble means and the normalized anomalies for the observations:
$$\begin{aligned} y_k^{b,j} &= \mathcal{H}_k(x_k^{b,j}) \\ y_k^b &= \frac{1}{N}\sum_{j=1}^N y_k^{b,j}, \\ Y_k^b &= \frac{1}{\sqrt{N-1}}(y_k^{b,j} - y_k^b)_{j=1}^N \in \mathbb{R}^{m\times N}; \end{aligned}$$

- Transforming matrix
$$T_k = (I_N + Y_k^{b\mathsf{T}}(\Sigma_k^o)^{-1}Y_k^b)^{-1} = (I_N + S_k^{\mathsf{T}} S_k)^{-1}$$
  for $S_k = (\Sigma_k^o)^{-1/2} Y_k^b$;

- normalized innovation vector
$$\delta_k = (\Sigma_k^o)^{-1/2}(y_k^o - y_k^b)$$

- Update the analyzed ensembles:
$$\begin{aligned} w_k^a &= (I_N + Y_k^{b\mathsf{T}}(\Sigma_k^o)^{-1}Y_k^b)^{-1} Y^{b\mathsf{T}}(\Sigma_k^o)^{-1}(y_k^o - y_k^b) = T_k S_k^{\mathsf{T}}\delta_k \\ x_k^a &= x_k^b + X_k^b w_k^a = x_k^b + X_k^b T_k S_k^{\mathsf{T}}\delta_k \\ X_k^a &= X_k^b(I_N + Y_k^{b\mathsf{T}}(\Sigma_k^o)^{-1}Y_k^b)^{-1/2}U,\ U1 = 1, U \in O(N) \\ x_k^{a,j} &= x_k^a + \sqrt{N-1}X_k^a[:,j] \\ &= x_k^b + X_k^b(w_k + \sqrt{N-1}T^{1/2}U[:,j]) \end{aligned}$$

  OR:
$$(x_k^{a,j})_j = x_k^b 1_N^{\mathsf{T}} + X_k^b(w_k 1_N^{\mathsf{T}} + \sqrt{N-1}T_k^{1/2}U)$$

C.6 ETKF-Q

The ETKF-Q algorithm borrowed from [37] (a special case of IEnKS-Q [19]).

C.6.1 Initialization

Construct $U \in \mathbb{R}^{N\times(N-1)}$ such that

$$\left(\tfrac{1}{\sqrt{N}}1_N \quad U\right)$$

is orthogonal, and let

$$\mathscr{U} = \left(\tfrac{1}{N}1_N \quad \tfrac{1}{\sqrt{N-1}}U\right),$$

then

$$\mathscr{U}^{-1} = \left(1_N \quad \sqrt{N-1}U\right)^{\mathsf{T}},$$

C.6.2 Forecast step

- Forward propagation:
$$x_k^{f,j} = \mathcal{M}_k(x_{k-1}^{a,j});$$
- obtain the mean and deviation for x_k^f
$$(x_k^f, \Delta_x^f) = (x_k^{f,j})_j \mathscr{U}$$
- eigen decomposition (approximately):
$$(\Delta_x^f \Delta_x^{f\mathsf{T}} + \Sigma_k^M)V_k \approx V_k \Lambda_k,$$
where $V_k \in \mathbb{R}^{n\times(N-1)}$ and $\Lambda_k \in \mathbb{R}^{(N-1)\times(N-1)}$ are the eigenvectors and eigenvalues of $\Delta_x^f \Delta_x^{f\mathsf{T}} + \Sigma_k^M$;
- update the deviation:
$$\Delta_x^b = V_k \Lambda_k^{1/2}$$
- update the ensembles:
$$(x_k^{b,j})_j = (x_k^f, \Delta_x^b)\mathscr{U}^{-1}$$

C.6.3 Analysis step

- Obtain the ensemble mean and deviation matrix from the ensembles:
$$(x_k^b, \Delta_x^b) = (x_k^{b,j})_j \mathscr{U}$$
- obtain the mean and deviation for background estimates of observations:
$$\begin{aligned} y_k^{b,j} &= \mathcal{H}_k(x_k^{b,j}) \\ (y_k^b, \Delta_y^b) &= (y_k^{b,j})_j \mathscr{U} \end{aligned}$$
- Transforming matrix:
$$T_k = (I_{N-1} + \Delta_y^{b\mathsf{T}}(\Sigma_k^o)^{-1}\Delta_y^b)^{-1} = (I_{N-1} + S_k^{\mathsf{T}} S_k)^{-1}$$
for $S_k = (\Sigma_k^o)^{-1/2}\Delta_y^b$;
- normalized innovation vector
$$\delta_k = (\Sigma_k^o)^{-1/2}(y_k^o - y_k^b)$$
- Update the analyzed ensembles: (modified from ETKF)
$$\begin{aligned} w_k^a &= (I_{N-1} + \Delta_y^{b\mathsf{T}}\Sigma_k^o\Delta_y^b)^{-1}\Delta_y^{b\mathsf{T}}(\Sigma_k^o)^{-1}(y_k^o - y_k^b) = T_k S_k^{\mathsf{T}}\delta_k \\ x_k^a &= x_k^b + \Delta_x^b w_k^a = x_k^b + \Delta_x^b T_k S_k^{\mathsf{T}}\delta_k \\ \Delta_x^a &= \Delta_x^b(I_{N-1} + \Delta_y^{b\mathsf{T}}(\Sigma_k^o)^{-1}\Delta_y^b)^{-1/2} \\ x_k^{a,j} &= (x_k^a, \Delta_x^a)\mathscr{U}^{-1} \end{aligned}$$

D Implementation

We provide the proposed LAINR framework on Github, with the following detailed explanation and psedo-codes.