Towards Interpretable Deep Learning and Analysis of Dynamical Systems via the Discrete Empirical Interpolation Method

Anonymous submission

Abstract

We present a differentiable framework that leverages the Discrete Empirical Interpolation Method (DEIM) for interpretable deep learning and dynamical system analysis. Although DEIM efficiently approximates nonlinear terms in projection-based reduced-order models (POD-ROM), its fixed interpolation points limit the adaptability to complex and time-varying dynamics. To address this limitation, we first develop a differentiable adaptive DEIM formulation for the one-dimensional viscous Burgers' equation, which allows neural networks to dynamically select interpolation points in a computationally efficient and physically consistent manner. We then apply DEIM as an interpretable analysis tool for examining the learned dynamics of a pre-trained Neural Ordinary Differential Equation (NODE) on a two-dimensional vortex-merging problem. The DEIM trajectories reveal physically meaningful features in the NODE's learned dynamics and expose its limitations when extrapolating to unseen flow configurations. These findings demonstrate that DEIM can serve not only as a model reduction tool but also as a diagnostic framework for understanding and improving the generalization behavior of neural differential equation models.

Introduction

Machine learning has recently emerged as a powerful tool for modeling high-dimensional nonlinear dynamical systems. In particular, neural differential equation models such as Neural Ordinary Differential Equations (NODEs) provide a continuous-time formulation for learning the evolution of physical systems directly from data (Chen et al. 2018). However, despite their success in reproducing complex dynamics, these models often operate as black boxes, offering limited interpretability and uncertain generalization when applied to unseen physical regimes. Developing interpretable frameworks that can diagnose, analyze, and improve such models is therefore essential for scientific machine learning.

Projection-based reduced-order modeling (ROM) methods, such as Proper Orthogonal Decomposition (POD) and the Discrete Empirical Interpolation Method (DEIM), have long been used to reduce the computational complexity of partial differential equations (PDEs) while retaining physical interpretability. DEIM in particular identifies sparse interpolation points for approximating nonlinear terms efficiently, linking reduced-order dynamics to spatially localized physical structures (Chaturantabut and Sorensen 2010). However, the original variant of DEIM uses a fixed set of interpolation points determined offline, which limits its ability to capture transient and nonlinear phenomena that evolve dynamically.

To address this limitation, we first propose a differentiable adaptive DEIM framework that learns to dynamically select interpolation points in an end-to-end differentiable manner. Using the one-dimensional viscous Burgers' equation as a benchmark, we train a neural network to generate DEIM sampling matrices using the Gumbel–Softmax estimator, allowing discrete, yet differentiable point selection. This formulation preserves the computational efficiency of DEIM while allowing adaptivity to changing flow features.

Building on this concept, we further apply DEIM as a tool for interpretable analysis of learned NODE dynamics in a two-dimensional vortex-merging problem. The assumption, here, is that the dynamically identified interpolation points correspond to 'important' locations in the flow-field. By applying DEIM to windows of snapshots obtained during the evolution of the trained NODE, we extract trajectories of representative sampling points and analyze how they evolve compared to those obtained from ground-truth dynamics. These trajectories reveal physically meaningful flow structures and provide direct insight into the NODE's ability (or lack thereof) to generalize to unseen flow conditions.

Together, these two components demonstrate the potential of DEIM not only as a model reduction method but also as a general-purpose framework for interpretable and physically consistent analysis of learned dynamical systems.

Background and Related Work

Projection-based Reduced-order Models

Proper Orthogonal Decomposition (POD) is a data-driven method to construct low-dimensional subspaces that optimally capture the dynamics of high-dimensional systems. This process includes two steps: snapshot collection and basis construction. Snapshot collection extracts solutions $\mathbf{u}(t_i) \in \mathbb{R}^n$ from full-order simulations or experiments at time instances $t_i (i = 1, ..., N)$. Then, a basis is identified via the singular value decomposition (SVD) on the snapshot matrix $\mathbf{U} = [\mathbf{u}(t_1), ..., \mathbf{u}(t_N)] \in \mathbb{R}^{n \times N}$ to obtain orthogonal POD modes $\mathbf{\Psi} = [\Psi_1, ..., \Psi_m] \in \mathbb{R}^{n \times m}$, where m is the number of truncated POD modes and $n >> m$.

$\Psi_i (i = 1, ..., m)$ is ranked by each mode's energy content. The number of truncated POD modes is typically determined by setting an energy threshold, such that the retained modes collectively captures a predetermined percentage (e.g., 99%) of the total variance (energy) in the dataset. This is achieved by computing the cumulative sum of the eigenvalues (or singular values) and selecting the smallest number of modes for which the cumulative energy exceeds the chosen threshold. In some cases, additional dynamical or physical considerations may influence the final selection of modes.

Using the orthonormal property of these POD modes, Galerkin projection projects governing equations (e.g. nonlinear PDEs) onto a reduced subspace spanned by these modes. A nonlinear PDE in Eq. 1,

$$\frac{\partial \mathbf{u}}{\partial t} = \mathcal{L}_f[\mathbf{u}] + \mathcal{N}_f[\mathbf{u}], \quad (1)$$

where $\mathcal{L}_f$ and $\mathcal{N}_f$ are linear and nonlinear operators in the full-order space, respectively, can be approximated using the POD modes as follows:

$$\frac{\partial \mathbf{\Psi a}}{\partial t} = \mathcal{L}_f[\mathbf{\Psi a}] + \mathcal{N}_f[\mathbf{\Psi a}], \quad (2)$$

where $\mathbf{a} = [a_1, ..., a_m] \in \mathbb{R}^m$ are temporal POD coefficients, and $\mathbf{u}$ is approximated as $\mathbf{u} \approx \mathbf{\Psi a} = \sum_{i=1}^{m} \psi_i a_i$

Then, by using the orthonormality of the POD modes, the final reduced system becomes as follows:

$$\frac{\partial \mathbf{a}}{\partial t} = \mathcal{L}_r[\mathbf{a}] + \mathbf{\Psi}^T \mathcal{N}_f[\mathbf{\Psi a}], \quad (3)$$

where $\mathcal{L}_r$ is a linear operator in the reduced-order space. This approach preserves system dynamics while reducing computational cost, because the dimension of $\mathbf{a}$ in the reduced system, m, is orders of magnitude smaller than that of $\mathbf{u}$, n in the original full-order system. However, since the POD modes cannot commute with the nonlinear operator unlike the linear operator, $\mathcal{N}_f$ remains expensive to evaluate in high dimensional systems.

Discrete empirical interpolation

In order to address the complexity of calculating the nonlinear term in Eq. 3, Discrete Empirical Interpolation Method (DEIM) (Chaturantabut and Sorensen 2010) approximates $\mathcal{N}_f[\mathbf{u}]$ using a sparse subset of interpolation points. This method starts by constructing a separate POD basis $\Phi \in \mathbb{R}^{n \times l}$ from nonlinear snapshots, $\mathbf{U}_{nl} = [\mathbf{u}_{nl}(t_1), ..., \mathbf{u}_{nl}(t_N)] \in \mathbb{R}^{n \times N}$. Here, l is the number of truncated POD modes for nonlinear snapshots, and $n \gg l$. Then, the DEIM sampling matrix, $\mathbf{P} = [e^{p_1}, ..., e^{p_l}]$, where $\mathbf{P} \in \mathbb{R}^{n \times l}$ and $e^{p_k} (k = 1, ..., l)$ is a one-hot vector with a 1 at the p_k-th entry and zeros elsewhere, is constructed using a greedy algorithm to select l interpolation points that minimize the approximation error. Then, $\mathcal{N}_f[\mathbf{u}]$ can be approximated as follows:

$$\mathcal{N}_f[\mathbf{u}] \approx \Phi(\mathbf{P}^T \Phi)^{-1} \mathbf{P}^T \mathcal{N}_f[\mathbf{u}] \quad (4)$$

Eq. 4 reduces the cost of evaluating $\mathcal{N}_f[\mathbf{u}]$ from $O(n)$ to $O(l)$, because the nonlinear terms $\mathcal{N}_f[\mathbf{u}]$ are only calculated at $x_{p_k} (k = 1, ..., l)$. The sampling matrix $\mathbf{P}$ is pre-computed and fixed during online stage of ROM. Eq. 4 is integrated to Eq. 3 to get the final form of the PDE in the reduced order space as follows:

$$\frac{\partial \mathbf{a}}{\partial t} = \mathcal{L}_r[\mathbf{a}] + \mathbf{\Psi}^T \Phi(\mathbf{P}^T \Phi)^{-1} \mathbf{P}^T \mathcal{N}_f[\mathbf{\Psi a}] \quad (5)$$

DEIM can reduce the computational cost of POD-Galerkin frameworks by maintaining a low computational complexity for the calculation of the nonlinear term. For details on the DEIM algorithm, the reader is referred to the previous work (Chaturantabut and Sorensen 2010).

Differentiable Physics

Differentiable physics frameworks aim to seamlessly integrate physical simulation and machine learning by enabling gradient-based optimization through the governing equations of dynamical systems (Sanderse et al. 2024). These frameworks reformulate numerical solvers in a differentiable manner, allowing model parameters to be optimized via backpropagation while preserving the physical consistency imposed by the underlying PDEs. Differentiability can be achieved using automatic differentiation, adjoint-based methods, or source-to-source transformations of existing solvers. Recent studies have leveraged such frameworks to train machine learning models for LES turbulence closures, reduced-order model corrections, and boundary conditions in fluid–structure interaction problems (Sirignano and MacArt 2023; Kim et al. 2023; Ahmed and Stinis 2023; Fan and Wang 2024). Ultimately, differentiable physics bridges the gap between data-driven and physics-based modeling, providing a unified foundation for interpretable and physically consistent learning.

Neural Ordinary Differential Equations

A neural ordinary differential equation (NODE)(Chen et al. 2018) can be formulated as:

$$\frac{d\mathbf{u}(\mathbf{t})}{dt} = f(\mathbf{u}(\mathbf{t}), t; \theta) \quad (6)$$

, where $\mathbf{u}(\mathbf{t}) \in \mathbb{R}^N$ is a state variable in each timestep, and $f(\mathbf{u}(\mathbf{t}), t, \theta)$ is the hidden unit dynamics, which is learned by neural networks with the parameter θ and evaluated to determine the solution with a differential equation solver. For a nonlinear PDE as in Eq. 1, learning $f(\mathbf{u}(\mathbf{t}), t, \theta)$ using $\mathbf{u}(\mathbf{t})$ implies that one learns the entirety of right-hand side of the PDE including the linear operator $\mathcal{L}_f$ and the nonlinear operator $\mathcal{N}_f$, which is evaluated at each time step. This can be expressed as:

$$\frac{d\mathbf{u}(\mathbf{t})}{dt} = \mathcal{L}_f[\mathbf{u}(\mathbf{t})] + \mathcal{N}_f[\mathbf{u}(\mathbf{t})] \approx f(\mathbf{u}(\mathbf{t}), t; \theta) \quad (7)$$

Adaptive DEIM with differentiable physics

In this section, we present a differentiable adaptive DEIM framework for interpretable learning. This approach enables deep neural networks to generate sampling matrices compatible with the original DEIM formulation, preserving the computational efficiency of nonlinear term evaluation in

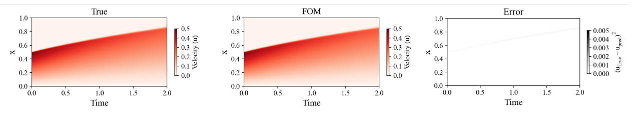

Figure 1: **(a)** Velocity contour from the analytic solution; **(b)** Velocity contour from the FOM; **(c)** Squared error

ROMs while allowing for adaptive and data-driven selection of interpolation points.

Problem Setup

Eq. 8 represents one-dimensional viscous Burger's equation, and it has been a benchmark case for testing new numerical schemes or machine learning models. This test case includes quadratic nonlinearity and a moving discontinuous shock. Here, the Reynolds number Re controls the ratio of the convection strength to dissipation, therefore, a smaller Re means a more dissipative flow condition.

$$\frac{\partial u}{\partial t} + \frac{1}{2}\frac{\partial(u^2)}{\partial x} - \frac{1}{Re}\frac{\partial^2 u}{\partial x^2} = 0 \tag{8}$$

Applying the Cole–Hopf transformation (Cole 1951; Hopf 1950) to the specific initial and boundary conditions given in Eq. 10 yields the analytical solution shown in Eq. 9.

$$u(x,t) = \frac{\frac{x}{t+1}}{1 + \sqrt{\frac{t+1}{\exp(Re/8)}}\exp(Re\frac{x^2}{4t+4})} \tag{9}$$

$$\begin{aligned} u(x,0) &= \frac{x}{1 + \sqrt{\frac{1}{\exp(Re/8)}}\exp\left(Re\frac{x^2}{4}\right)}, \\ u(0,t) &= 0, u(L,t) = 0 \end{aligned} \tag{10}$$

In this study, we use a discretized version of Eq. 8 under boundary and initial conditions as in Eq. 10 as the full-order model (FOM). Here, the number of points in x-direction is set to 128, and 2nd-order upwind and central differencing schemes are used for approximating the advection and dissipation terms, respectively. For time integration, explicit Strong Stability Preserving Runge-Kutta 3rd order (SSP-RK3) is used to reduce oscillations and instability of solution while attaining high-order accuracy (Gottlieb, Shu, and Tadmor 2001). The time integration is conducted until $t = 2.0$, and the number of time steps is 300. Fig. 1 shows velocity contour from the analytic solution in Eq. 9 and from the FOM and MSE error. As seen in the analytic solution in Fig. 1 (a), the discontinuity formed near $x = 0.5$ at $t = 0.0$ propagates to $x = 1.0$, and the magnitude of the discontinuity gradually decreases as time goes because of the dissipation term in Eq. 8. FOM solution in Fig. 1 (b) captures these advection and diffusion of the discontinuity without oscillation and unstable behavior of solutions. This is also confirmed in Fig. 1 (c), where oscillations around the discontinuity are not observed.

Adaptive DEIM formulation

While the original DEIM uses a fixed set of interpolation points to approximate nonlinear terms during the online stage of ROM, it may fail to capture transient dynamics or complex physical phenomena that evolve over time. To overcome these limitations, we propose a machine learning (ML)-based adaptive sampling strategy. In the proposed approach, instead of using a pre-computed, static sampling matrix $\mathbf{P}$, a deep neural network (DNN) is trained to predict the optimal set of interpolation points at each time step based on the current system state:

$$\mathbf{P}^t = f_\theta(\mathbf{u}_t, a_t) \tag{11}$$

, where f_θ is the DNN with trainable parameters θ, $\mathbf{u}_t$ is the current state, $\mathbf{a}_t$ are the POD coefficients, and $\mathbf{P}^t$ denotes the adaptive interpolation points for the next time step. However, as mentioned in the previous section, the DEIM sampling matrix $\mathbf{P}^t$ consists of l one-hot vectors, while the outputs of the DNN f_θ in Eq. 11 have continuous values. This cannot reduce the computational complexity of calculating the nonlinear terms, because the nonlinear terms have to be calculated at every points. One possible way of making the output of a DNN as a discrete one-hot vectors is to apply argmax function to the output tensor of the DNN. However, argmax is not differentiable, so gradients calculated from the final loss function cannot be backpropagated through the solver. A well-established approach to approximating discrete outputs in a differentiable manner is to apply temperature scaling to the softmax function(Jang, Gu, and Poole 2016). This technique is notably formalized in the Gumbel-Softmax estimator, which enables differentiable sampling of categorical variables by introducing a temperature parameter into the softmax operation. As the temperature decreases, the softmax output becomes increasingly similar to a one-hot vector, thereby closely approximating discrete selection while preserving differentiability for gradient-based optimization (Jang, Gu, and Poole 2016). The procedure of the differentiable adaptive DEIM sampling method is outlined as follows.

First, we get a tensor with the same shape as the DEIM sampling matrix $\mathbf{P}^t$ as the output of the DNN:

$$\mathbf{z}^t = f_\theta(\mathbf{u}_t, a_t) \tag{12}$$

, where $\mathbf{z}^t = [z_1^t, ..., z_l^t] \in \mathbb{R}^{n \times l}$ is the raw output of the DNN. The proposed DNN architecture is designed as follows. The network consists of two fully connected hidden layers, each containing 2,048 units. The first block applies a linear transformation followed by Layer Normalization, ReLU activation, and Dropout with a rate of 0.2 to prevent overfitting. The second block incorporates a residual skip connection that adds the input of the block to its output, facilitating gradient flow and stable training. Finally, the output layer projects the extracted features to the target spatial resolution, which is then reshaped to match the dimensions of the sampling matrix. Note that the final layer outputs raw logits without an activation function to be compatible with the subsequent sampling logic.

In order to approximate the discrete, one-hot structure required by DEIM while maintaining differentiability during training, we apply a Gumbel-softmax estimator (Jang, Gu, and Poole 2016) to each column of the raw output:

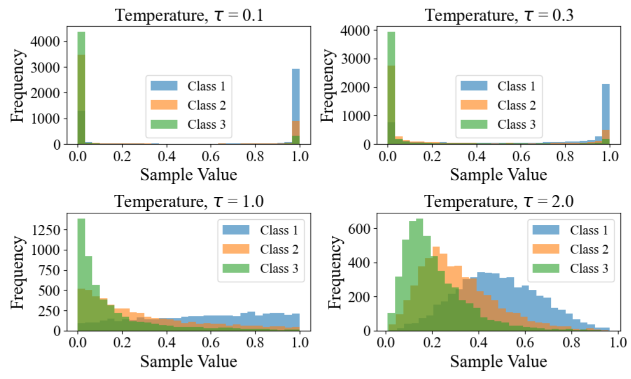

Figure 2: Effect of the temperature parameter τ on Gumbel–Softmax samples. Lower values of τ produce sharper, nearly one-hot samples, while higher values lead to smoother and more uniform distributions.

$$\mathbf{\Pi}^t = [\pi_1^t, .., \pi_l^t] = [\text{softmax}\left(\frac{z_1^t}{\tau}\right), ..., \text{softmax}\left(\frac{z_l^t}{\tau}\right)] \tag{13}$$

, where $\mathbf{\Pi}^t = \pi_i^t \ (i = 1, ..., l) \in \mathbb{R}^{n \times l}$ denotes the Gumbel-softmax distribution for choosing each interpolation point, and the softmax temperature τ is a positive scalar controlling the sharpness of the output distribution. Fig. 2 shows example Gumbel-Softmax distributions of sampled values under different temperature parameters. By setting a low temperature (e.g., $\tau = 0.3$), the softmax output becomes sharply peaked, closely approximating a one-hot vector while remaining differentiable. This allows the model to be trained end-to-end using gradient-based optimization.

The total loss function $\mathcal{L}_{total}$ used for training is defined as a linear combination of the reconstruction error $\mathcal{L}_{recon}$ for the time-evolved snapshots and the spectral error $\mathcal{L}_{freq}$ in the frequency domain, as shown in Eq. 14.

$$\mathcal{L}_{total} = \mathcal{L}_{recon} + \lambda \cdot \mathcal{L}_{freq} \tag{14}$$

Here, $\mathcal{L}_{recon}$ denotes the Root Mean Square Error (RMSE) between the predicted and target values, while $\mathcal{L}_{freq}$ represents L_1-norm of the FFT-transformed coefficients, designed to preserve high-frequency information essential for simulating rapid changes such as moving shocks. The hyperparameter λ controls the trade-off between two terms.

While we use a Gumbel–Softmax distribution to enable differentiability during training, we apply the argmax operation to the output probability distribution during inference to obtain a discrete selection of interpolation points. This approach is widely adopted in machine learning, particularly in classification tasks, where the model outputs a probability distribution via softmax over possible classes, and the final predicted class is chosen as the one corresponding to the highest probability (i.e., the index with the maximum value). In our formulation, applying argmax during inference effectively restores the one-hot structure required for DEIM as follows:

$$\mathbf{p}^t = [p_1^t, ..., p_l^t] = [\arg\max_j \pi_0^{(j),t}, ..., \arg\max_j \pi_l^{(j),t}] \tag{15}$$

where $\mathbf{p}^t = [p_1^t, ..., p_l^t] \in \mathbb{R}^{n \times l}$ denotes the indices of the selected interpolation points. Then, the ML-based adaptive DEIM sampling matrix, $\mathbf{P}_{ML}^t$, can be constructed as follows:

$$\mathbf{P}_{ML}^t = [e^{p_1^t}, ..., e^{p_l^t}] \tag{16}$$

This approach enables DNNs to generate DEIM sampling matrices that are compatible with the original DEIM framework, thus preserving the computational efficiency for the evaluation of nonlinear terms during the online stage of ROM, while also allowing for adaptive and data-driven selection of interpolation points.

Results

When applying DEIM to the POD-ROM as in Eq. 5, the numbers of POD modes for the full solution field, m, and for the nonlinear terms, l, must be determined. We set $m = 12$ and $l = 24$, and train the adaptive DEIM model to compare its performance with the standard DEIM under the same ROM parameters. In addition, the temperature parameter for the Gumbel-Softmax distribution is set to $\tau = 0.3$ and fixed during training, and the hyperparameter λ for the loss function in Eq. 14 is set to $\lambda = 0.001$. Fig. 3 shows the velocity contours obtained from the original DEIM and the trained adaptive DEIM model, along with the corresponding errors relative to the full-order model (FOM) results. The original DEIM model exhibits large errors along the discontinuity throughout the entire simulation, whereas the adaptive DEIM model significantly reduces the errors in that region. Fig. 4 presents the mean squared error (MSE) at each time step. The adaptive DEIM model consistently maintains lower errors than the original DEIM model throughout the simulation. Fig. 5 illustrates the trajectories of the sampling points selected by the original DEIM and the adaptive DEIM model, respectively. While the original DEIM uses fixed sampling points, the adaptive DEIM model dynamically selects a point near $x = 0.3$ and adaptively adjusts the sampling points toward the end of the simulation to further reduce errors.

We also train the adaptive DEIM model in Eq. 12 using a different number of modes for the nonlinear terms ($l = 18$), resulting in 18 corresponding sampling points. The results are compared with those from the original DEIM with $l = 18$ as well as with the previous results obtained using $l = 24$. Figure 6 presents the velocity prediction contours from both the original and the adaptive DEIM models for $l = 18$ and $l = 24$, along with their corresponding squared error contours. Both adaptive DEIM models with $l = 18$ and $l = 24$ exhibit lower errors than the original DEIM model with the same number of sampling points. Notably, the adaptive DEIM model with $l = 18$ achieves a lower error than the original DEIM with $l = 24$, despite using fewer sampling points. Throughout the entire simulation, the POD-ROM coupled with the adaptive DEIM

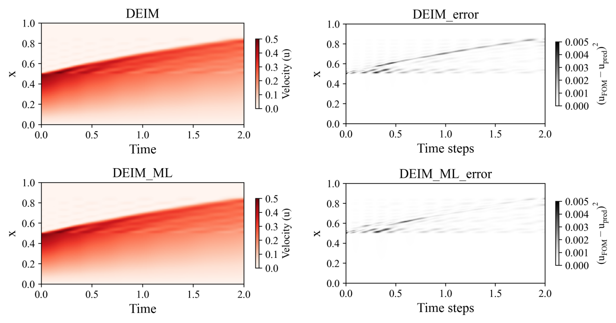

Figure 3: Velocity prediction contours obtained from the original DEIM and the trained adaptive DEIM model, along with the corresponding squared error contours for each model

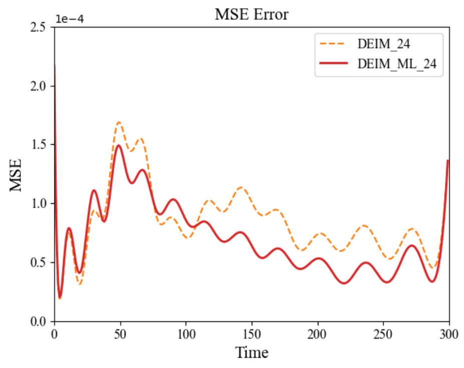

Figure 4: Mean squared error (MSE) at each time step for the original DEIM and the trained adaptive DEIM model

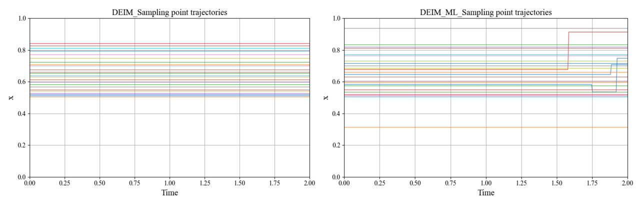

Figure 5: Trajectories of sampling points for the original DEIM and the trained adaptive DEIM model

model ($l = 18$) performs approximately 5,400 nonlinear term evaluations, whereas the POD-ROM with the original DEIM model ($l = 24$) requires about 7,200 such evaluations. Figure 7 presents the mean squared error (MSE) at each time step. It can be observed that the adaptive DEIM model with 18 sampling points yields smaller errors than the original DEIM model with 24 sampling points, despite requiring fewer nonlinear term evaluations. To evaluate the computational overhead of the proposed method, we measured the total wall-clock time for the time integration process. The standard DEIM with $M = 18$ and $M = 24$ required approximately 5.24s and 5.36s, respectively. In contrast, the ML-DEIM models took 6.90s ($M = 18$) and 8.20s ($M = 24$). The proposed framework incurs a computational overhead of approximately 30% to 50% compared to the standard DEIM. This increase is primarily attributed to the online inference cost of the sampling network at each time step. It is also noteworthy that the ML-DEIM with $M = 18$ exhibits a longer runtime (6.90s) than the standard DEIM with $M = 24$ (5.36s), despite performing fewer nonlinear term evaluations. This counter-intuitive result arises because the current benchmark has a relatively low spatial degree of freedom ($N = 256$), making the physics-based nonlinear calculation computationally inexpensive. Consequently, the time saved by reducing sampling points is insufficient to offset the fixed cost of the ML inference. However, we anticipate that this balance will shift in high-dimensional systems (e.g., $N \gg 10^5$) or problems involving complex physics, such as reacting flows or 3D turbulence. In such regimes, the cost of evaluating the nonlinear term dominates the total runtime, rendering the ML overhead negligible in comparison. Therefore, further experiments on larger-scale systems are necessary to fully demonstrate the computational efficiency and scalability of the proposed framework.

DEIM for Analysis of Trained Neural ODEs and Nudging-based Data Assimilation

In this section, we perform an interpretabilty analysis of the pre-trained NODEs. Specifically, windowed-DEIM is applied to the learned dynamics of the PDEs, $f(\mathbf{u}(\mathbf{t}), t; \theta)$ in Eq. 7, to identify representative sampling points and examine their dynamic behavior. Furthermore, Nudging-based data assimilation is conducted based on the sampled DEIM points.

Problem Setup

For this experiment, we employ the two-dimensional vortex-merging problem (Ahmed et al. 2023). The incompressible two-dimensional Navier–Stokes equations, expressed in terms of the vorticity and stream function, can be written as follows:

$$\frac{\partial \omega}{\partial t} = -J(\omega, \psi) + \frac{1}{Re}\nabla^2 \omega \tag{17}$$

$$\nabla^2 \psi = -\omega \tag{18}$$

$$J(\omega, \psi) = \frac{\partial \omega}{\partial x}\frac{\partial \psi}{\partial y} - \frac{\partial \omega}{\partial y}\frac{\partial \psi}{\partial x}, \tag{19}$$

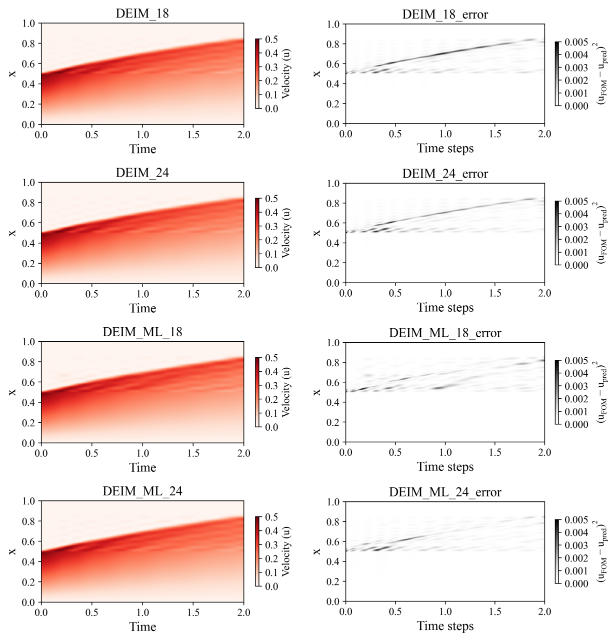

Figure 6: Velocity prediction contours obtained from the original DEIM and the trained adaptive DEIM models for $l = 18$ and $l = 24$, together with the corresponding squared error contours for each case

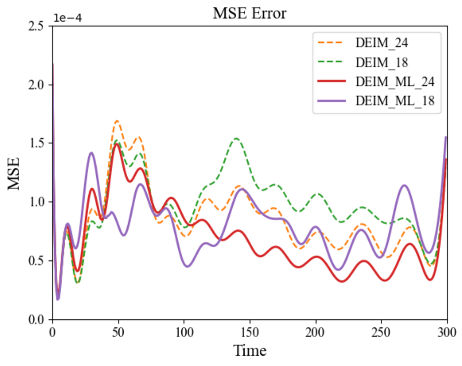

Figure 7: Mean squared error (MSE) at each time step for the original DEIM and the trained adaptive DEIM models with $l = 18$ and $l = 24$

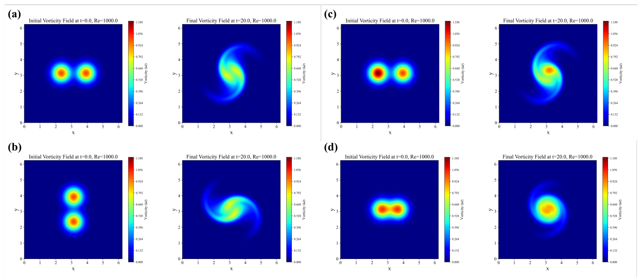

Figure 8: Final solutions of the two-dimensional vortex-merging problem for $Re = 1,000$ with different initial conditions: (a) horizontally symmetric vortices, (b) vertically symmetric vortices, (c) horizontally asymmetric vortices, and (d) horizontally symmetric vortices with a smaller separation distance

where ω and ψ are vorticity and stream function, respectively, and $J(\omega, \psi)$ is the Jacobian operator, which determines the nonlinear advection of vorticity. For the training dataset, the computational domain is set to $[0, 2\pi] \times [0, 2\pi]$, and the number of grid points in x- and y- directions is set to $N_x = N_y = 128$. The timestep size is $dt = 2 \times 10^{-2}$, with a total of 200 timesteps, and the SSP-RK3 scheme is used for time integration. Fig. 8 presents the final solutions of Eq. 17 for different initial conditions at $Re = 1,000$.

The NODE is trained to learn the right-hand side of Eq. 17, which includes both the nonlinear Jacobian operator and the linear viscous dissipation term. For this PDE, we train a single CNN-based NODE model using the first 100 snapshots from the solution initialized with horizontal vortices, as shown in Fig. 8(a), and then apply the trained model to other initial conditions. The CNN architecture consists of four convolutional layers with 3×3 kernels and ReLU activations, each followed by residual connections that propagate low-level spatial information through the network. A final 1×1 convolutional layer projects the 32 intermediate feature channels back to a single scalar field, representing the predicted temporal derivative of the flow variable. This architecture allows the NODE to efficiently capture local nonlinear interactions and spatial correlations in the vortex dynamics. To train this network, we utilized the JAX (Bradbury et al. 2018) framework on a single NVIDIA A100 GPU, minimizing the MSE loss using the Adam optimizer (Kingma 2014) with an initial learning rate of 1×10^{-3}. We employed a step-based learning rate scheduler that decays the rate by a factor of 1.1 every 10,000 iterations over a total of 40,000 training steps. The network weights were initialized using the He normal strategy (He et al. 2015), and the time integration was performed using the Heun method (Ascher and Petzold 1998) provided by the Diffrax library (Kidger 2021). For robust optimization, we used a batch size of 5 with a trajectory length of 60 time steps randomly sampled from the training data.

Applying DEIM to Learned Dynamics

When applying DEIM to the trained NODE models, we employ a time-windowed DEIM approach. In this method, the right-hand side of each PDE is collected over a fixed time interval to form a snapshot matrix. The sampling matrix P is then computed from these snapshots, after which the initial snapshot is shifted forward by a stride time step to construct consecutive snapshot matrices. This procedure enables us to track the trajectories of representative sampling points during simulations and to evaluate the learned dynamics of the NODE models by comparing them with the trajectories obtained from the ground-truth data. For the vortex merging problem, the number of sampling points in the original DEIM is set to 16, the window size is set to 20, and the stride timestep is set to 1.

Figure 9(b) shows the trajectories of sampling points obtained by the DEIM from the ground truth and the NODE prediction for the case with initially symmetric horizontal vortices, as shown in Fig.8(a). The color bar represents time steps, where a color closer to red indicates later time steps. As can be seen in the figure, as the two vortices rotate counterclockwise and merge, the sampling points obtained by the DEIM also rotate in the same direction. Since the two vortices are symmetric, the sampling points rotate around the two vortex cores. Fig. 9(c) shows the trajectories of the first two sampling points for clarity. As can be seen from these two figures, the trajectories of the sampling points from the NODE prediction follow a similar trend to those from the ground truth during the first half of the simulation but begin to deviate from the circular pattern afterward, repeatedly visiting specific points. This trend is consistent with the L2 norm plot in Fig. 9(a), where the L2 error starts to increase after time $t = 10.0$, beyond the range covered by the training data.

Figure 10 is structured in the same way as Fig.9, except that it corresponds to the case with initially horizontal asymmetric vortices as in Figs. 8 (b). This case is an extrapolation case, where the model is not exposed to these snapshots during training. In Figs. 10(b) and (c), the trajectories of the sampling points obtained by the DEIM follow a partially circular pattern during the early stage of the simulation but quickly deviate from this regular pattern and begin to oscillate irregularly. This trend is consistent with the L2-norm plot in Fig. 10(a), where the L2 norm starts to increase rapidly soon after the beginning of the simulation.

Figures 11 and 12 share the same structure as Fig. 9, but correspond to the cases with initially horizontal vortices having a smaller separation distance and with initially vertical vortices, as shown in Figs. 8(c) and (d), respectively. The trajectories of the sampling points obtained by the DEIM from the NODE results in these two cases do not form any circular patterns at the beginning; instead, they exhibit highly zigzag and random behaviors. The randomness of these trajectories is even worse than in the previous case with initially asymmetric vortices. This can also be observed from the L_2 norm plots, where the L_2 norms for these two cases increase more rapidly after the prediction starts compared to the asymmetric case. These results indicate that the trained NODE generalizes poorly to cases where vortices are closely placed at

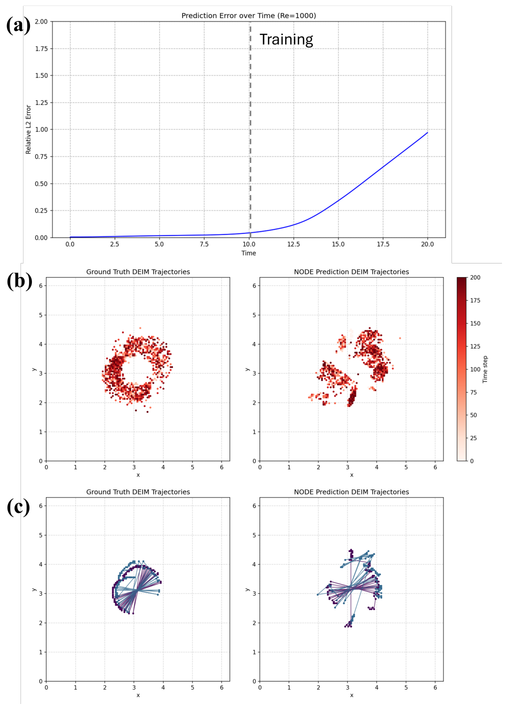

Figure 9: For the case with initial horizontal vortices: (a) L2-norm error at each timestep; (b) trajectories of sampling points from the ground truth and the NODE prediction; (c) trajectories of the first two sampling points.

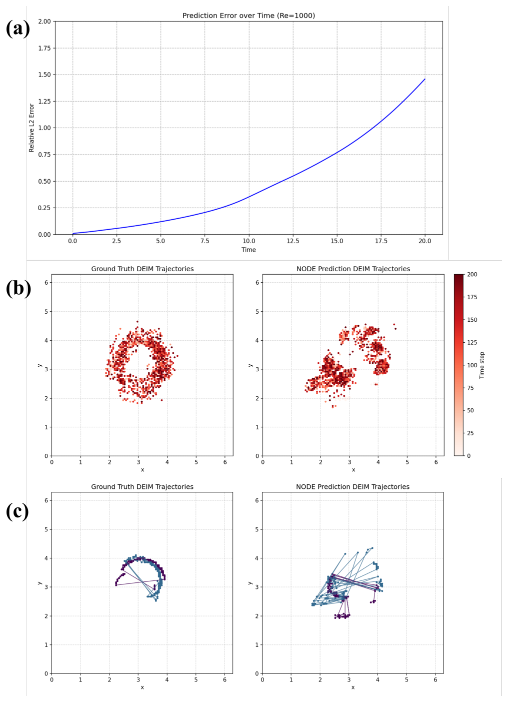

Figure 10: For the case with initial asymmetric horizontal vortices: (a) L2 norm error at each timestep; (b) trajectories of sampling points from the ground truth and the NODE prediction; (c) trajectories of the first two sampling points.

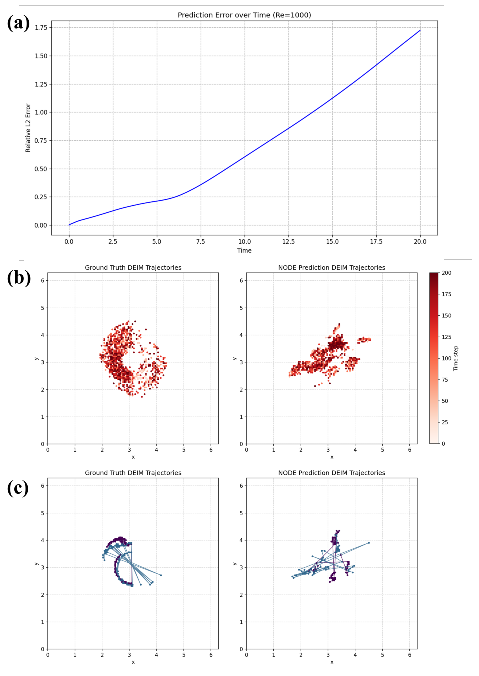

Figure 11: For the case with initial horizontal vortices with a small distance: (a) L2 norm error at each timestep; (b) trajectories of sampling points from the ground truth and the NODE prediction; (c) trajectories of the first two sampling points.

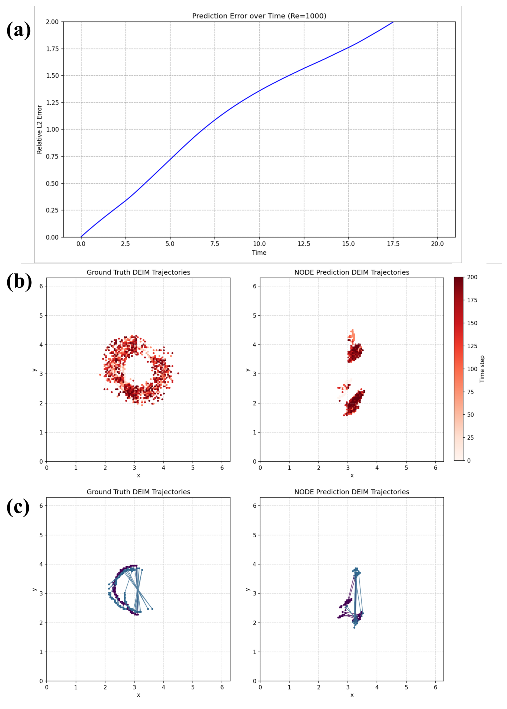

Figure 12: For the case with initial vertical vortices: (a) L2 norm error at each timestep; (b) trajectories of sampling points from the ground truth and the NODE prediction; (c) trajectories of the first two sampling points.

the start of a simulation.

Conclusion

We have developed a differentiable adaptive DEIM framework for interpretable learning and analysis of nonlinear dynamical systems. By leveraging differentiable sampling through the Gumbel–Softmax estimator and coupling it with a POD-ROM, our approach provides a novel pathway towards bridging data-driven learning with reduced-order modeling. Future directions include applying the proposed differentiable sampling mechanism to large-scale industrial applications to demonstrate its scalability and robustness in complex real-world scenarios.

Through systematic experiments on two-dimensional vortex-merging flows, we showed that DEIM-based trajectory analysis serves as a diagnostic tool for evaluating the extrapolation robustness of NODEs. In the cases of initially closer horizontal vortices and initially vertical vortices, the sampling-point trajectories exhibit highly irregular, zigzag patterns and faster growth of L_2 errors, indicating degraded generalization compared to the initial asymmetric vortex case. These findings demonstrate that the trained NODE perceives these flow configurations as more extrapolative, providing an interpretable link between learned dynamics and physical flow regimes.

Beyond interpretability, the DEIM analysis of pre-trained NODEs offers a promising pathway for enhancing predictive performance through data assimilation. The dynamically identified DEIM points correspond to the most informative spatial locations for capturing nonlinear dynamics. By using these points as optimal observation or assimilation locations, future work can integrate data assimilation and optimal sensor placement to elegantly correct NODE predictions, reducing accumulated forecast errors and improving stability in long-term evolution.

References

Ahmed, S. E.; San, O.; Rasheed, A.; Iliescu, T.; and Veneziani, A. 2023. Physics guided machine learning for variational multiscale reduced order modeling. *SIAM Journal on Scientific Computing*, 45(3): B283–B313.

Ahmed, S. E.; and Stinis, P. 2023. A multifidelity deep operator network approach to closure for multiscale systems. *Computer Methods in Applied Mechanics and Engineering*, 414: 116161.

Ascher, U. M.; and Petzold, L. R. 1998. *Computer methods for ordinary differential equations and differential-algebraic equations*. SIAM.

Bradbury, J.; Frostig, R.; Hawkins, P.; Johnson, M. J.; Leary, C.; Maclaurin, D.; Necula, G.; Paszke, A.; VanderPlas, J.; Wanderman-Milne, S.; et al. 2018. JAX: composable transformations of Python+ NumPy programs.

Chaturantabut, S.; and Sorensen, D. C. 2010. Nonlinear model reduction via discrete empirical interpolation. *SIAM Journal on Scientific Computing*, 32(5): 2737–2764.

Chen, R. T.; Rubanova, Y.; Bettencourt, J.; and Duvenaud, D. K. 2018. Neural ordinary differential equations. *Advances in neural information processing systems*, 31.

Cole, J. D. 1951. On a quasi-linear parabolic equation occurring in aerodynamics. *Quarterly of applied mathematics*, 9(3): 225–236.

Fan, X.; and Wang, J.-X. 2024. Differentiable hybrid neural modeling for fluid-structure interaction. *Journal of Computational Physics*, 496: 112584.

Gottlieb, S.; Shu, C.-W.; and Tadmor, E. 2001. Strong stability-preserving high-order time discretization methods. *SIAM review*, 43(1): 89–112.

He, K.; Zhang, X.; Ren, S.; and Sun, J. 2015. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE international conference on computer vision*, 1026–1034.

Hopf, E. 1950. The partial differential equation.

Jang, E.; Gu, S.; and Poole, B. 2016. Categorical reparameterization with gumbel-softmax. *arXiv preprint arXiv:1611.01144*.

Kidger, P. 2021. *On Neural Differential Equations*. Ph.D. thesis, University of Oxford.

Kim, H.; Shankar, V.; Viswanathan, V.; and Maulik, R. 2023. Generalizable data-driven turbulence closure modeling on unstructured grids with differentiable physics. *arXiv e-prints*, arXiv–2307.

Kingma, D. P. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Sanderse, B.; Stinis, P.; Maulik, R.; and Ahmed, S. E. 2024. Scientific machine learning for closure models in multiscale problems: A review. *arXiv preprint arXiv:2403.02913*.

Sirignano, J.; and MacArt, J. F. 2023. Deep learning closure models for large-eddy simulation of flows around bluff bodies. *Journal of Fluid Mechanics*, 966: A26.