Proper Orthogonal Decomposition in Accelerated Neural Ordinary Differential Equations

We demonstrate the effectiveness of training accelerated neural ordinary differential equation (NODE) architectures on the set of time variable coefficient functions generated by the proper orthogonal decomposition of a differential equation. Proper orthogonal decomposition allows for the system to be significantly reduced while still maintaining a high degree of accuracy. Significant advances in neural ordinary differential equations have led to an increased learning rate when using a heavy ball differential equation or a generalized heavy ball differential equation. These findings support the accelerated behavior of HBNODE and GHBNODE particularly over the reduced trajectory. Furthermore we consider the effectiveness of the neural network to make long term predictions about the convergence of the differential equation to a steady state solution.

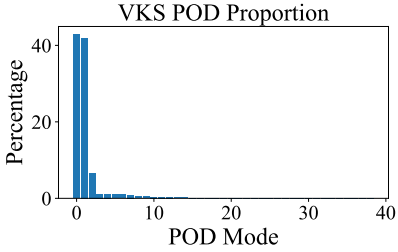
1 Introduction

Training neural differentials on proper orthogonal decompositions has been shown to have positive results (insert refs). The accuracy depends heavily on the dynamics of the differential equation and the ability to sufficiently learn the This paper compares the results of training neural ordinary differentials on the Von Karman street flow with training on heavy ball neural ordinary differentials. Additionally, we consider a generated dataset on the Euler equations as well as a data set of the cylindrical dynamics in three dimensions (what is this data set called?). These datasets are based on higher order differential equations and have more complicated dynamics. Additionally we consider transient, and quasi-periodic states of training. These separate phases demonstrate the improvement of higher order neural ordinary differentials as well as the limitations of differential solvers.

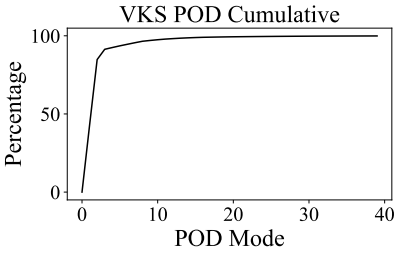
Discussion of the "impact" of the results or analysis of the seen results. Discussion of future area of research? (neural fractional solvers, etc)

2 Proper Orthogonal Decomposition

Proper orthogonal decomposition formulates a physical system as the product of orthogonal spatial functions and time dependent coefficients. The data is taken as a series of snapshots and centered to have zero mean. The decomposition is an eigenvalue decomposition performed on the covariance matrix generated by the inner product of the snapshot matrices. The eigenvalues are sorted by magnitude corresponding to spatial modes of larger variance. A select few of these predominant eigenvalues are chosen for training based on their proportional importance.



(a) Dominant eigenvalues by percentage



(b) Alternative Figure

Consider a one dimensional system with state variables ρ, E and v. The data snapshots form the data matrix U. The rows of this data matrix correspond to the fluctuating components of the state variables at a given time.

$$U = \begin{pmatrix} u_\rho(x_1, y_1, t_1) & \dots & u_E(x_n, y_m, t_1) & \dots & u_v(x_n, y_m, t_1) \\ \vdots & & \vdots & & \vdots \\ u_\rho(x_1, y_1, t_k) & \dots & u_E(x_n, y_m, t_k) & \dots & u_v(x_n, y_m, t_k) \end{pmatrix}$$

The mean is then subtracted from the state matrix so that it has zero mean. Then the covariance matrix is formed by considering the inner product $C = U^T U$. The eigenvalues of the covariance matrix form the time dependent coefficients also called POD modes $\alpha_i(t)$. The eigenvectors form the spatial functions $\psi_i(x, y)$. Thus the full system dynamics can be recovered by computing

$$u(x,t) = \sum_{i=0}^{N} \alpha_i(t)\psi_i(x,y)$$

This decomposition significantly reduces the number of state variables for training. It is unclear how to generate the spatial functions for unseen data.

3 Neural Ordinary Differential Equations

Neural ordinary differential equations consider the continuous limit of the sequence of states in a neural network. Consider the following limit for the hidden state variables h_t and neural network function $f(h_t, t, \theta)$ with learn-able parameters θ.

$$h_{t+1} = h_t + f(h_t, t, \theta)$$

In the continuous limit, which is similar to increasing the number of layers in the network, we have the following differential equation.

$$\frac{dh(t)}{dt} f(h(t), t, \theta)$$

Given some initial input sequence $h(t_i)$ for $i = 0, \dots n-1$, the NODE learns the output at $h(t_n)$ using a black-box numerical ODE solver. For the black-box solver the required number of forward pass function evaluations (NFEs) is analogous to the depth of the network in a residual network (ResNet).

The following gradient is used to calculate the loss based on the prediction $h(t_n)$.

$$\frac{d\mathcal{L}}{d\theta} = \int_{t_0}^{t_n} a(t) \frac{\partial f(h(t), t, \theta)}{\partial \theta} dt$$

The gradient uses the adjoint state $a(t) = \partial \mathcal{L} / \partial h(t)$ which satisfies

$$\frac{da(t)}{dt} = -a(t) \frac{\partial f(h(t), t, \theta)}{\partial h}$$

The HBNODE paper contains all of my knowledge of the advantages of NODE and I was essentailly parrotting here.

3.1 Accelerated NODEs

Consider additionally the heavy ball method which adds a momentum parameter to the state sequence.

$$h_{t+1} = h_t + f(h_t, t, \theta) + \beta(h_t + h_{t-1})$$

For step size $s > 0$ and momentum parameter β, introduce parameter γ so that the following holds.

$$m_{t+1} = \frac{h_{t+1} - h_t}{\sqrt{s}}, \beta = 1 - \gamma\sqrt{s}$$

In the continuous limit we have the following

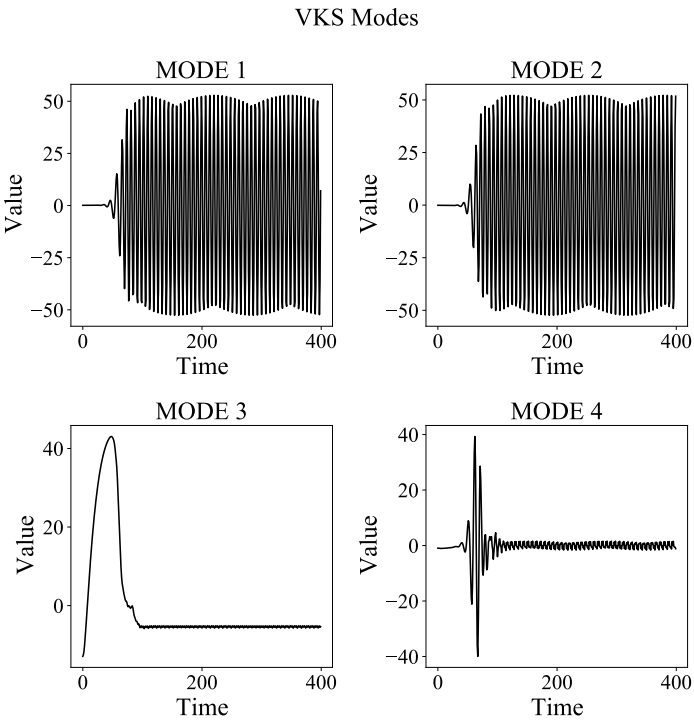
$$\frac{dh(t)}{dt} = m(t), \quad \frac{dm(t)}{dt} = -\gamma m(t) + f(h(t), t, \theta)$$

For which we have the dampened oscillator equation

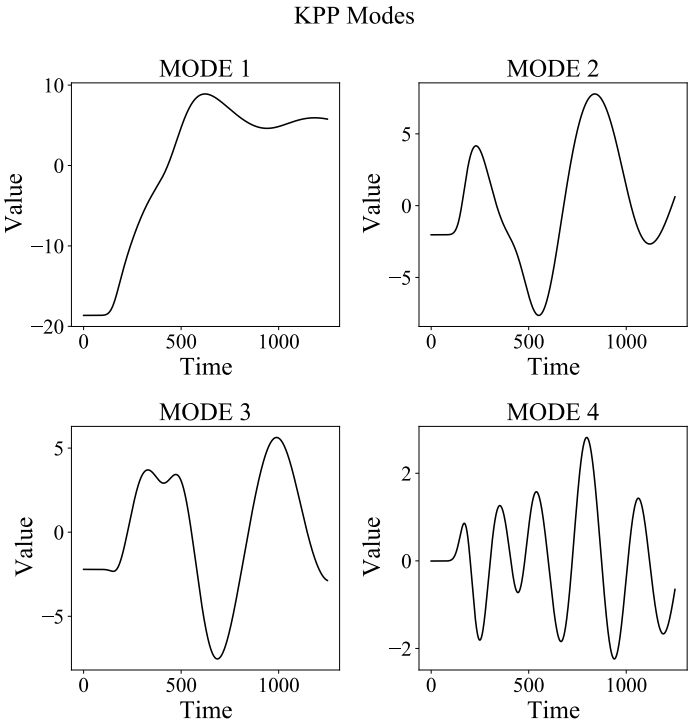
$$\frac{d^2h(t)}{dt^2} + \gamma\frac{dh(t)}{dt} = f(h(t), t, \gamma)$$

4 Data Sets

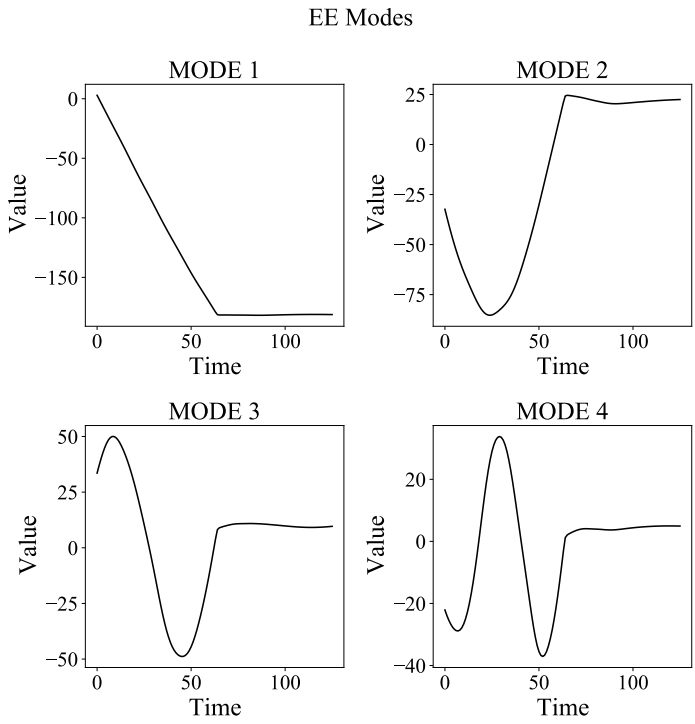
We consider three datasets that pose various difficulties in training. The first dataset is the von Karman vortex street (VKS) flow simulates a fluid flow past a cylinder, generating vortex-shedding dynamics.



After a transient phase, the vortex dynamics become periodic, establishing clear oscillating dynamics for the POD modes. The second dataset is a simulation of diffusivity of a cylinder in three dimensions using the Kurganov Petrova Popov (KPP) equation. Like the VKS dataset this dataset has a clear transient and periodic phase. However, during the periodic phase, the POD modes are not perfectly oscillatory.



The final dataset is a simulation of the Euler equations in one dimension. This dataset uses density, energy and velocity data which achieve a periodic phase at different time intervals.



These datasets offer a distinct range of dynamics for learning. The dynamics in each dataset grow more complex for the periodic and transient phases. In general it is desirable for the dataset trajectory to be representative of the whole dynamics. This problem is exaserbated by reducing the system to a select number of modes. Nevertheless, the dynamics of the reduced order model can be learned.

5 Results

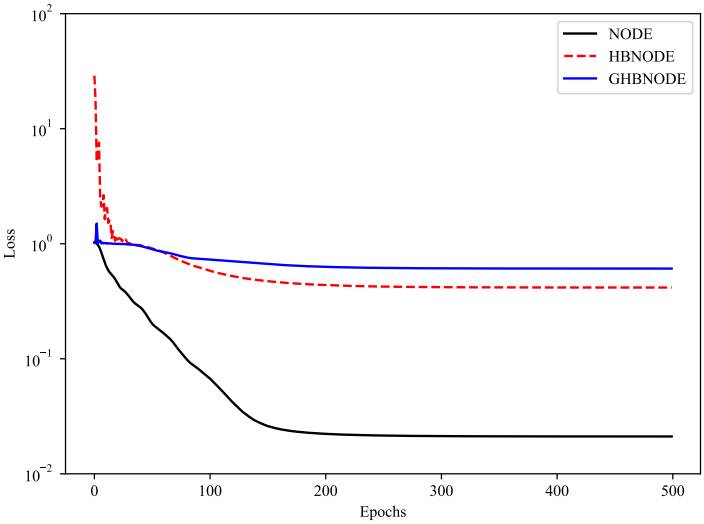


Fig. 2: Figure 1