Learning Dissipative Dynamics in Chaotic Systems

Anonymous Author(s)
Affiliation
Address
`email`

Abstract

Chaotic systems are notoriously challenging to predict because of their sensitivity to perturbations and errors due to time stepping. Despite this unpredictable behavior, for many dissipative systems, the long term trajectories converge to an invariant measure supported on a low dimensional set, known as the global attractor. For Markovian systems, the statistical properties of long-term trajectories are uniquely determined by a Markov operator that maps the evolution of the system over an infinitesimal time step. In this work, we propose a machine learning framework to learn the underlying Markov operator of dissipative chaotic systems which captures their statistical behavior without the need to predict the exact trajectories. Using this framework, for the first time, we are able to predict various statistics of the invariant measure for the turbulent Kolmogorov Flow dynamics with Reynolds numbers up to 500.

1 Introduction

Machine learning methods for chaotic systems. Chaotic systems are characterized by strong instabilities. Small changes in the initialization or errors during time-stepping accumulate and lead to vastly diverging trajectories. Such instability makes chaotic systems challenging, both for mathematical analysis and numerical simulation. Because of the intrinsic instability, it is infeasible for any method to capture the exact trajectory of a chaotic system for long periods. Therefore, prior works either fit recurrent neural networks (RNN) on extremely short trajectories or only learn a step-wise projection from a randomly generated evolution using reservoir computing (RC) [1–4]. These previous attempts are able to push the limits of faithful prediction to moderate periods on low dimensional ordinary differential equations (ODEs), e.g. the Lorenz-63 system, or on one-dimensional partial differential equations (PDEs), e.g. the Kuramoto-Sivashinsky (KS) equation. However, they are less effective at modeling more complicated turbulent systems such as the Navier-Stokes equation (NS), especially over long time periods. Indeed, predicting long trajectories of such chaotic systems is an ill-posed problem and we cannot expect such attempts to be successful. Instead, we take a new perspective: we aim to capture statistical properties of long trajectories, even if we cannot precisely predict them.

Invariants in chaos. Despite their instability, many chaotic systems exhibit certain reproducible statistical properties, such as the auto-correlation and, for PDEs, the energy spectrum. Such properties remain the same for different realizations of the initial condition [5]. This is provably the case for the Lorenz-63 model [6, 7] and empirically holds for many dissipative PDEs, such as the KS equation and the two-dimensional Navier-Stokes equation (Kolmogorov flows) [8]. Dissipativity is a physical property of many natural systems. Intuitively higher-energy flows in such systems dissipate more strongly. Mathematically, there exists a global **attractor** which is a compact set towards which the system tends to evolve. The dissipativity property implies that for a given system there is a set which any given trajectory enters in finite time, depending on the initial condition, and thereafter

Submitted to 36th Conference on Neural Information Processing Systems (NeurIPS 2022). Do not distribute.

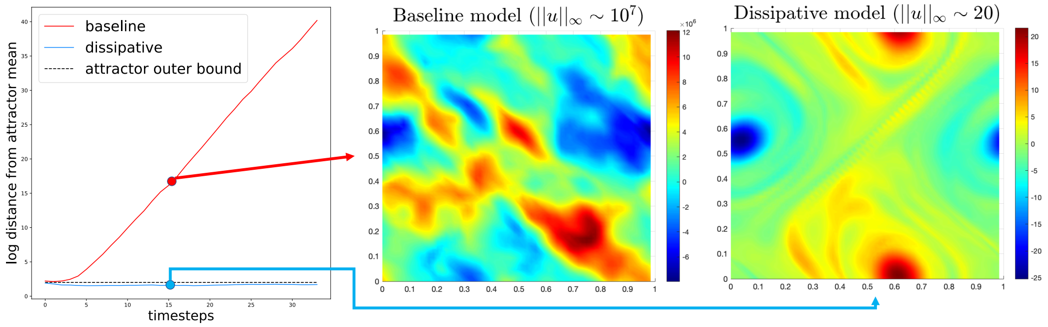


Figure 1: Dynamic evolution of the Markov neural operator for Kolmogorov Flow systems, from initial conditions near the attractor, with and without the enforced dissipativity constraints. The baseline model in the middle has no dissipativity constraint while the dissipative model on the right has the constraint enforced during its time evolution. Baseline model blows up, whereas the dissipative model returns to the attractor. The dissipative models is trained using the Fourier neural operator architecture in the manner shown in Figure 2.

remains inside. The global attractor is defined by mapping all initial conditions from bounded sets forward in time. Furthermore there is strong empirical evidence that many dissipative systems are ergodic i.e. there exists an invariant measure which charges the global attractor. While learning infinite time-horizon trajectories is intractable, it is possible to approximate the attractor and invariant measure using a **Markov operator**, a memoryless deterministic operator that captures the evolution of the dynamics along an infinitesimal time step. The dissipativity property helps to make this problem tractable [9, 10]. For Markovian systems, e.g. autonomous differential equations, samples from the invariant measure can be obtained through repeatedly composing the Markov operator. This property is implied since the family of Markov operators defined for any fixed time forms a semigroup [9]. By learning a Markov operator, we are able to quickly and accurately generate an approximate attractor and estimate its invariant measure for a variety of chaotic systems that are of interest to the physics and applied mathematics communities [11–17].

Neural operators. To learn the Markov operators for PDEs, we need to model the time-evolution of functions in infinite-dimensional function spaces. This is especially challenging when we need to generate long trajectories since even a small error accumulates over multiple compositions of the learned operator, potentially causing an exponential build-up or a collapse due to the high dimension of the space. Because we study the evolution of functions in time, we propose to use the recently developed operator learning method known as the neural operator [18, 19]. The neural operator remedies the mesh-dependent nature of finite-dimensional operator methods such as RNNs, CNNs, and RC. Neural operators are guaranteed to universally approximate any operator in a mesh independent manner, and hence, can capture the Markov operator of chaotic systems. This approximation guarantee and the absorption of trajectories by the global attractor makes it possible to accurately follow it over long time horizons, allowing us access to the invariant measure of chaotic systems.

Our contributions. In this work, we formulate a machine learning framework for chaotic systems exploiting their dissipativity and Markovian properties. We propose the Markov neural operator (MNO) and train it given only one-step evolution data from a chaotic system. By composing the learned operator over a long horizon, we accurately approximate the global attractor of the system [20]. Our architecture is outlined in Figure 2. In order to assess its performance, we study the statistics of the associated invariant measure such as the Fourier spectrum, the spectrum of the proper orthogonal decomposition (POD), the point-wise distribution, the auto-correlation, and other domain-specific statistics such as the turbulence kinetic energy and the dissipation rate. Furthermore we study the behavior of our leaned operator over long horizons and ensure that it does not blow up or collapse but rather accurately follows the global attractor. In this work:

- We theoretically prove that, under suitable conditions, the MNO can approximate the underlying Markov operator of chaotic PDEs, while conventional neural networks lack such strong guarantees.

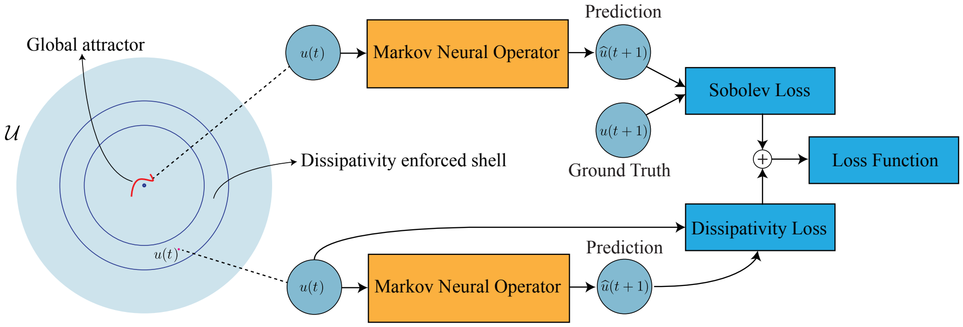


Figure 2: Markov neural operator (MNO): learn global dynamics from local data

Learn the MNO from the local time-evolution data with the Sobolev and dissipativity losses . $u(t)$ is t'th time step of the chaotic system. Sobolev losses of various order are used to compute the loss of prediction next time step $\widehat{u}(t+1)$ as of $u(t+1)$. Dissipativity loss is computed by drawing a random sample $u(t)$ from the dissipativity shell to make sure that in expectation next time step prediction $\widehat{u}(t+1)$ dissipates and has a smaller norm as of v.

- We impose dissipativity by augmenting the data on an outer shell to enforce that the dynamic evolution stays close to the attractor. We show this is a crucial for learning in a chaotic regime as demonstrated by Figure 1. The resulting system remains stable against large perturbations.
- We study the choice of time steps for training the MNO, demonstrating that the error follows a valley-shaped phenomenon. This gives rise to a recipe for choosing the optimal time step for accurate learning.
- We show that standard mean square error (MSE) type losses for training are not adequate, and the models often fail to capture the higher frequency information induced from the derivatives.
- We investigate various Sobolev losses in operator learning. We show that using the Sobolev norms for training captures higher-order derivatives and moments, as well as high frequency details [21, 22]. This is similar in spirit to pre-multiplying the spectrum by the wavenumber, an approach commonly used in fluid analysis [23].
- We investigate multiple exiting deep learning architectures, including U-Net [24], long short-term memory convolution neural networks (LSTM-CNN) [25], and gated recurrent unit (GRU) [26], in place of the neural operator to learn the predictive operator. We show MNO provides an order of magnitude lower error on all loss functions studied in this paper. Furthermore, we show that the MNO desirably outperforms the above mentioned neural network models previous models, on all statistics mentioned.

Our experiments show that Sobolev norms are crucial in training the MNO, allowing for significant performance improvement over standard loss functions (e.g. MSE) and for preserving invariant statistics such as the spectrum. Thus we propose a principled approach for learning chaotic systems that incorporates operator learning and dissipativity.

2 Problem setting

We consider potentially infinite dimensional dynamical systems where the phase space $\mathcal{U}$ is a Banach space and, in particular, a function space on a Lipschitz domain $D \subset \mathbb{R}^d$ (for finite dimensional systems, $\mathcal{U}$ will be a Euclidian space). We are interested in the initial-value problem

$$\frac{du}{dt}(t) = F(u(t)), \qquad u(0) = u_0, \qquad t \in (0, \infty) \tag{1}$$

for initial conditions $u_0 \in \mathcal{U}$ where F is usually a non-linear operator. We will assume, given some appropriate boundary conditions on ∂D when applicable, the solution $u(t) \in \mathcal{U}$ exists and is unique for all times $t \in (0, \infty)$. When making the spatial dependence explicit, if it is present, we will write $u(x,t)$ to indicate the evaluation $u(t)|_x$ for any $x \in D$. We define the family of operators

$S_t : \mathcal{U} \to \mathcal{U}$ as mapping $u_0 \mapsto u(t)$ for any $t \geq 0$, and note that, since (1) is autonomous, S_t satisfies the Markov property i.e. $S_t(S_s(u_0)) = u(s+t)$ for any $s, t \geq 0$. We adopt the viewpoint of casting time-dependent PDEs into function space ODEs (1), as this leads to the semigroup approach to evolutionary PDEs which underlies our learning methodology.

Dissipativity. Systems for which there exists some bounded, positively-invariant set E such that for any bounded $B \subset \mathcal{U}$, there is some time $t^* = t^*(E, B)$ beyond which the dynamics of any trajectory starting in B enters and remains in E are known as **dissipative systems** ([9]). The set E is known as the **absorbing set** of the system. For such systems, the global attractor A, defined subsequently, is characterized as the ω-limit set of E. In particular, for any initial condition $u_0 \in \mathcal{U}$, the trajectory $u(t)$ approaches A as $t \to \infty$. In this work, we consider dissipative dynamical systems where there exist some $\alpha \geq 0$ and $\beta > 0$ such that

$$\frac{1}{2}\frac{d}{dt}||u||^2 \leq \alpha - \beta||u||^2 \tag{2}$$

for all $u \in \mathcal{U}$. It can be shown that systems which satisfy this inequality are dissipative ([9]) with the absorbing set E, an open ball of radius $\sqrt{\alpha/\beta + \varepsilon}$ for any $\varepsilon > 0$. There are several well-known examples of dynamical systems that satisfy the above inequality. In this paper we consider the finite-dimensional Lorenz-63 system and the infinite-dimensional cases of the Kuramoto-Sivashinsky and 2D incompressible Navier-Stokes equations, in the form of Kolmogorov flows ([8]).

Global Attractors. The long time behavior of the solution to (1) is characterized by the set $U = U(u_0) \subset \mathcal{U}$ which is **invariant** under the dynamic i.e. $S_t(U) = U$ for all $t \geq 0$, and the orbit $u(t)$ converges

$$\inf_{v \in U} \|u(t) - v\|_{\mathcal{U}} \to 0 \qquad \text{as} \qquad t \to \infty.$$

When it exists, U is often identified as the ω-limit set of u_0. The chaotic nature of certain dynamical systems arises due to the complex structure of this set because $u(t)$ follows U and U can be, for example, a fractal set. A compact, invariant set A is called a **global attractor** if, for any bounded set $B \subset \mathcal{U}$ and any $\epsilon > 0$ there exists a time $t^* = t^*(\epsilon, A, B)$ such that $S_t(B)$ is contained within an ϵ-neighborhood of A for all $t \geq t^*$. Many PDEs arising in physics such as reaction-diffusion equations describing chemical dynamics or the Navier-Stokes equation describing the flow of fluids are dissipative and possess a global attractor which is often times finite-dimensional [8]. Therefore, numerically characterizing the attractor is an important problem in scientific computing with many potential applications.

Data distribution. For many applications, an exact form for the possible initial conditions to (1) is not available; it is therefore convenient to use a stochastic model to describe the initial states. To that end, let μ_0 be a probability measure on $\mathcal{U}$ and assume that all possible initial conditions to (1) come as samples from μ_0 i.e. $u_0 \sim \mu_0$. Then any possible state of the dynamic (1) after some time $t > 0$ can be thought of as being distributed according to the pushforward measure $\mu_t := S_t^\sharp \mu_0$ i.e. $u(t) \sim \mu_t$. Therefore as the dynamic evolves, so does the type of likely functions that result. This further complicates the problem of long time predictions since training data may only be obtained up to finite time horizons hence the model will need the ability to predict not only on data that is out-of-sample but also out-of-distribution.

Ergodic systems. To alleviate some of the previously presented challenges, we consider **ergodic** systems. Roughly speaking, a system is ergodic if there exists an **invariant measure** μ such that after some time $t^* > 0$, we have $\mu_t \approx \mu$ for any $t \geq t^*$ (in fact, μ can be defined without any reference to μ_0 or its pushforwards, see [27] for details). That is, after some large enough time, the distribution of possible states that the system can be in is fixed for any time further into the future. Indeed, μ charges the global attractor A. Notice that ergodicity is a much more general property than having **stationary states** which means that the system has a fixed period in time, or having **steady states** which means the system is unchanged in time.

Ergodicity mitigates learning a model that is able to predict out-of-distribution since both the input and the output of $\hat{S}_h$, an approximation to S_h, will approximately be distributed according to μ. Furthermore, we may use $\hat{S}_h$ to learn about μ since sampling it simply corresponds to running the dynamic forward. Indeed, we need only generate data on a finite time horizon in order to learn $\hat{S}_h$, and, once learned, we may use it to sample μ indefinitely by simply repeatedly composing $\hat{S}_h$ with itself. Having samples of μ then allows us to compute statistics which characterize the long term behavior of the system and therefore the global attractor A. Note further that this strategy avoids the issue of accumulating errors in long term trajectory predictions since we are only interested in the property that $\hat{S}_h(u(t)) \sim \mu$.

Notably, the existence of a global attractor does not imply the existence of an invariant measure. Indeed, the only deterministic and chaotic systems that are proven to possess an invariant measure are certain ODEs such as the Lorenz-63 system [7]. On the other hand, proving the existence of an invariant measure for deterministic and chaotic PDEs such as the KS or KF equations are still open problems, despite ergodic behavior being observed empirically.

3 Learning the Markov neural operator in chaotic dynamics

We propose the Markov neural operator, a method for learning the underlying Markov operators of chaotic dynamical systems. In particular, we approximate the operator mapping the solution from the current step to the next step $\hat{S}_h : u(t) \mapsto u(t+h)$. We approximate the Markov operator S_h, an element of the underlying continuous time semigroup $\{S_t : t \in [0, \infty)\}$, using a neural operator as detailed in Figure 2. See Appendix B.1 for background on the Markov operator and semigroup.

Long-term predictions. Having access to the map $\hat{S}_h$, its semigroup properties allow for approximating long time trajectories of (1) by repeatedly composing $\hat{S}_h$ with its own output. Therefore, for any $n \in \mathbb{N}$, we compute $u(nh)$ as follows,

$$u(nh) \approx \hat{S}_h^n(u_0) := \underbrace{(\hat{S}_h \circ \cdots \circ \hat{S}_h)}_{n \text{ times}}(u_0). \tag{3}$$

The above semigroup formulation can be applied with various choices of the backbone model for $\hat{S}_h$. In general, we prefer models that can be evaluated quickly and have approximation guarantees so the per-step error can be controlled. Therefore, we choose the standard feed-forward neural network [28] for ODE systems, and the Fourier neural operator [29] for infinite dimensional PDE systems.

For the the neural operator parametric class, we prove the following theorem regarding the Markov neural operator. The result states that our construction can approximate trajectories of infinite-dimensional dynamical systems arbitrary well. The proof is given in the Appendix.

Theorem 1. *Let $K \subset \mathcal{U}$ be a compact set and assume that, for some $h > 0$, the Markov operator $S_h : \mathcal{U} \to \mathcal{U}$ associated to the dynamic* (1) *is locally Lipschitz. Then, for any $n \in \mathbb{N}$ and $\epsilon > 0$ there exists a neural operator $\hat{S}_h : \mathcal{U} \to \mathcal{U}$ such that*

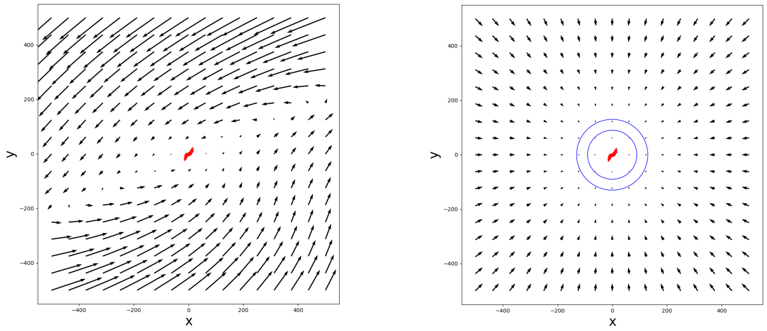
$$\sup_{u_0 \in K} \sup_{k \in \{1,\dots,n\}} \|u(kh) - \hat{S}_h^k(u_0)\|_{\mathcal{U}} < \epsilon.$$

Theorem 1 indicates that of choice of backbone model is rich enough to approximate many chaotic dynamical systems for a arbitrarily long period. For finite-dimensional systems, the same theorem holds with feed-forward neural networks instead of neural operators. We note that standard neural networks such as RNNs and CNNs *do not possess* such approximation theorems in the infinite-dimensional setting.

Invariant statistics. A useful application of the Markov operators is to estimate statistics of the invariant measure of a chaotic system. Assume the target system is ergodic and there exists an invariant measure μ such that $u(t) \sim \mu$ for any t as discussed in Section 2. An invariant statistic is defined as

$$T_G := \int_{\mathcal{U}} G(u) \, \mathrm{d}\mu(u) = \lim_{T \to \infty} \frac{1}{T} \int_0^T G(u(t)) \, \mathrm{d}t \tag{4}$$

for any functional $G : \mathcal{U} \to \mathbb{R}^d$. Examples include the L^2 norm, any spectral coefficients, and the spatial correlation, as well as problem-specific statistics such as the turbulence kinetic energy and dissipation rates in fluid flow problems. Given the property (3) and using the ergodicity from (4), the approximate model $\hat{S}_h$ can be used to estimate any invariant statistic simply by computing $T_G \approx \frac{h}{T} \sum_{k=1}^{n} G(S_h^k(u_0))$ for some $n = T/h$ and $T > 0$ large enough. Examples of fast approximation $\hat{S}_h$ which accurately predict invariant statics are given in Section 4.



(a) Without dissipativity enforced (b) Dissipativity enforced

Figure 3: Enforcing dissipativity on the Lorenz 63 system – extrapolated flow maps. The red points are training data on the attractor. The dissipativity is imposed by augmenting the data on the blue shell. The dissipativity enforcing training results in a learned dissipative Markov operator.

Enforcing dissipativity. In practice, training data for dynamical systems is typically drawn from trajectories that lie close to the global attractor of the system, so a priori there is no guarantee of a learned model's behavior far from the attractor. Thus, if we seek to learn the global attractor and invariant statistics of a dynamical system, it is crucial that we place contraints on the model to enforce that it learns a dissipative dynamical system.

In particular, given some Markov operator mapping between time-steps $\hat{S}_h : u(t) \mapsto u(t+h)$ and cost functional $C_D : \mathcal{U} \times \mathcal{U} \to \mathbb{R}$, we supplement the loss function with the additional term

$$\mathbb{E}_{u \sim \nu}\left[C_D\left(\hat{S}_h(u), u\right)\right] = \int_{\mathcal{U}} \|\hat{S}_h(u) - \lambda u\|_{\mathcal{U}}^2 \, \mathrm{d}\nu(u), \tag{5}$$

up to some multiplicative constant with respect to the other terms in the loss function, where $0 < \lambda < 1$ is some constant factor for scaling down (i.e., enforcing dissipativity) inputs u drawn from a probability measure ν. We choose ν to be a uniform probability distribution supported on some shell with a fixed inner and outer radii from the origin in $\mathcal{U}$. Our choice of cost functional C_D as given in eq. 5 scales down u by some constant factor λ, but in principle alternative dissipative cost functionals can be used.

We find that enforcing this dissipativity constraint on a shell at a sufficiently large radius encourages the learned Markov operator to produce dissipative predictions arbitrarily far away from the shell. In Section 4 we demonstrate that the constraint in eq. 5 prevents blow-up of a Markov operator trained on the turbulent Kolmogorov flow system.

4 Experiments

We evaluate the efficacy of our approach on the finite-dimensional, chaotic Lorenz-63 system as well as chaotic regimes of the 1D Kuramoto-Sivashinsky and 2D Navier-Stokes equations. In all cases we show that enforcing dissipativity is crucial for capturing the global attractor and evaluating statistics of the invariant measure. To the best of our knowledge, we showcase the first machine learning method able to predict the statistical behavior of a highly turbulent regime of the Navier-Stokes.

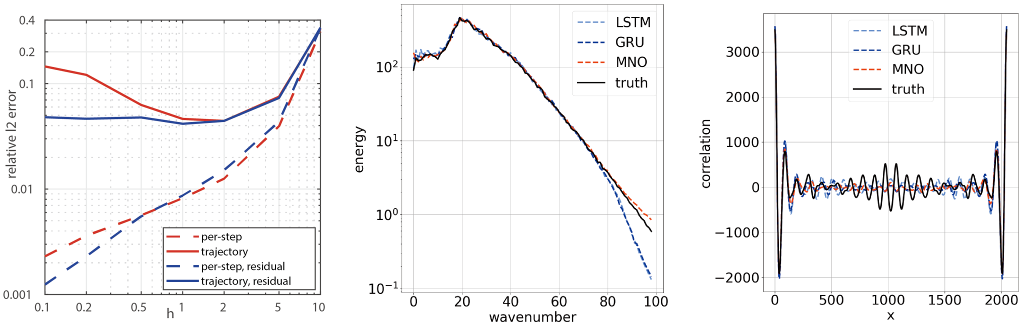
4.1 Lorenz-63 system

To motivate and justify our framework for learning chaotic systems in the infinite-dimensional setting (e.g., Navier-Stokes equations), we first apply our framework on the simple yet still highly chaotic Lorenz-63 ODE system.

The Lorenz-63 system constitutes a simplified climate model and is described by the following ODEs,

$$\dot{u}_x = \alpha(u_y - u_x), \qquad \dot{u}_y = -\alpha u_x - u_y - u_x u_z, \qquad \dot{u}_z = u_x u_y - b u_z - b(r+\alpha). \tag{6}$$

In this paper, we use the canonical parameters $(\alpha, b, r) = (10, 8/3, 28)$ [30]. Since the Markov operator of the Lorenz-63 system is finite-dimensional, we learn the Markov operator by training a



(a) (b) (c)

Figure 4: (a) Choice of time step for directly learning the Markov operator and learning its identity residual. We observe that both models induce smaller per step error for smaller h. When learn models are composed to generate longer trajectories, we observe that learning residual is advantageous. (b) Fourier spectrum of the predicted attractor. All models are able to capture the Fourier modes with magnitude larger than $O(1)$, while MNO is more accurate on the tail. (c) Spatial correlation of the attractor, averaged in the time dimension. MNO is more accurate on the near-range correlation, but all models miss the long-range correlation.

feedforward neural network on a single trajectory with $h = 0.05s$ on the Lorenz attractor. Figure 3 shows that enforcing dissipativity produces predictions that isotropically point towards the attractor, implying that the attractive properties of the Lorenz attractor are learned in the process. Observe that the our network is also dissipative outside the shell in which dissipativity was enforced during training. We conjecture that this is a property of ReLU networks that prevents model blow-up even in more difficult learning problems (see Figure 1).

We empirically find that enforcing dissipativity does not reduce the relative L^2 error compared to the baseline neural network. Further, the invariant statistics of the dissipative model align well with the ground-truth data distribution. Our results suggest that dissipativity can be enforced without significantly affecting the model's step-wise error and the learned statistsical properties of the attractor. See Appendix A.1 for more details.

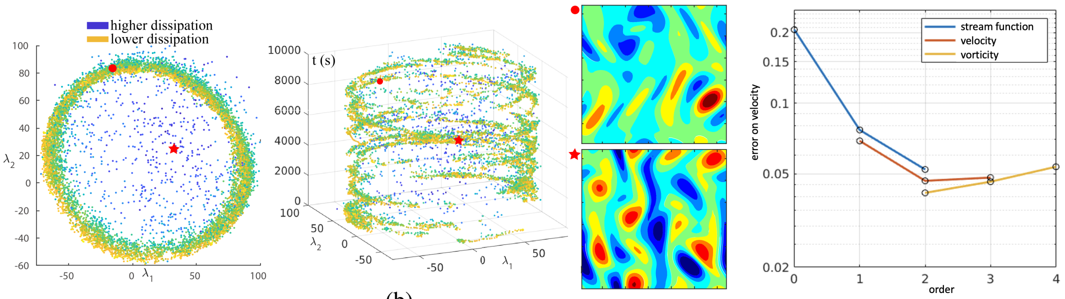
4.2 Kuramoto-Sivashinsky equation

We consider the following one-dimensional KS equation,

$$\frac{\partial u}{\partial t} = -u\frac{\partial u}{\partial x} - \frac{\partial^2 u}{\partial x^2} - \frac{\partial^4 u}{\partial x^4}, \text{on } [0, L] \times (0, \infty), \text{ with initial condition: } u(\cdot, 0) = u_0, \text{on } [0, L] \quad (7)$$

where the spatial domain $[0, L]$ is equipped with periodic boundary conditions. We study the impact the time step h has on learning. Our study shows that when the time steps are too large, the correlation is chaotic and hard to capture. But counter-intuitively, when the time steps are too small, the evolution is also hard to capture. In this case, the input and output of the learned operator will be very close, and the identity map will be a local minimum. We thus propose to use the MNO to also learn the time-derivative or residual directly. Figure 4a shows the results for varying h and when MNO is used to learn either the identity residual or the Markov operator itself. We observe that the residual model has a better per-step error and accumulated error at smaller h. When the time step is large, there is no difference in modeling the residual. This idea can generalize to other integrators as an extension of Neural ODEs to PDEs [31].

As shown in Figure 4b and 4c, we compare the performance of the MNO model against LSTM and GRU that we use to model the evolution operator of the KS equation with $h = 1s$. We observe that MNO model accurately recovers the Fourier spectrum of KS equation. We also present various other invariant statistics of the KS equation in Appendix A.2. For other statistics, all models perform similarly, except on the velocity distribution where MNO outperforms the LSTM and GRU. We emphasize that some these statistics are very challenging to capture and most machine learning approaches in the literature thus far fail to do so. (see Appendix A.2 for the details).



(a) (b) (c) (d)

Figure 5: The learned attractor of the Kolmogorov flow and choice of Sobolev loss for KF. (a-c) The 10000 time steps trajectory generated by MNO projected onto the first two components of PCA. Each point corresponds to an snapshot on the attractor. Two points are selected for further visualization of vorticity field. (d) The velocity error in models trained on stream function, velocity and vorticity using Sobolev loss of different orders. We observe that Sobolev norm of second order provides the best performing model across different learning regimes.

4.3 Kolmogorov Flow

We consider two-dimensional Kolmogorov flow (a form of the Navier-Stokes equations) for a viscous, incompressible fluid,

$$\frac{\partial u}{\partial t} = -u \cdot \nabla u - \nabla p + \frac{1}{Re}\Delta u + \sin(ny)\hat{x}, \qquad \nabla \cdot u = 0, \qquad \text{on } [0, 2\pi]^2 \times (0, \infty) \tag{8}$$

with initial condition $u(\cdot, 0) = u_0$ where u denotes the velocity, p the pressure, and $Re > 0$ is the Reynolds number. We enforce dissipativity during training with the criterion described in eq. 5, with $\lambda = 0.5$ and ν being a uniform probability distribution supported on a shell around the origin. We test the effect of enforcing dissipativity in the highly turbulent (and blow-up prone) $Re = 500$ setting, where we observe that a non-dissipative model blows up when composed with itself multiple times if the initial condition is perturbed slightly from the attractor (Figure 1), even though the model achieves relatively low L^2 error. In contrast, we empirically observe that the dissipative MNO does not blow up and its composed predictions returns to the attractor even when the initial condition is perturbed.

Model	training	loss	$\mathbf{L^2}$ error	$\mathbf{H^1}$ error	$\mathbf{H^2}$ error	TKE error	ϵ error
MNO	L^2 loss	0.0166	0.0187	0.0474	0.1729	**0.0136**	0.0303
	H^1 loss	0.0184	0.0151	0.0264	0.0656	0.0256	**0.0017**
	H^2 loss	0.0202	**0.0143**	**0.0206**	**0.0361**	0.0226	0.0193
U-Net	L^2 loss	0.0269	0.0549	0.1023	0.3055	0.0958	0.0934
	H^1 loss	0.0377	0.0570	0.0901	0.2164	0.1688	0.1127
	H^2 loss	0.0565	0.0676	0.0936	0.1749	0.0482	0.0841
ConvLSTM	L^2 loss	0.2436	0.2537	0.3854	1.0130	0.0140	24.1119
	H^1 loss	0.2855	0.2577	0.3308	0.5336	0.6977	6.9167
	H^2 loss	0.3367	0.2727	0.3311	0.4352	0.8594	4.0976

Table 1: Benchmark on vorticity for the Kolmogorov flow with Re= 40

Accuracy with respect to various norms. We study performance of MNO along with other predictive architectures, U-Net [24] and LSTM-CNN [25], on modeling the vorticity w in the relatively non-turbulent $Re = 40$ case 1. The MNO achieves one order of magnitude better accuracy compared to this architectures. As shown in Table 1, we train each model using the balanced $L^2 (= H^0)$, H^1, and H^2 losses, defined as the sum of the relative L^2 loss grouped by each order of derivative. We measure the error with respect to the standard (unbalanced) norms.

The MNO with H^2 loss consistently achieves the smallest error on vorticity for all of the L^2, H^1, and H^2 norms. However, the L^2 loss achieves the smallest error on the turbulence kinetic energy (TKE), while theH^1 loss achieves the smallest error on the dissipation ϵ.

The KF equation with $Re = 40$ is shown in Figure 10. (a) shows the ground truth vorticity field where each column represents a snapshot at $t = 100s$ with different initial conditions. (b), (c), and (d) show the predicted trajectory of MNO vorticity, using the L^2, H^1, and H^2 losses respectively. The five columns represent $t = 1000s, 2000s, 3000s, 4000s, 5000s$ respectively. The figure indicates that the predicted trajectories (b) (c) (d) share the same behaviors as in the ground truth (a), indicating that the MNO model is stable.

Estimating the attractor and invariant statistics. We compose MNO 10000 times to obtain the global attractor, and we compute the PCA (POD) basis of these 10000 snapshots and project them onto the first two components. As shown in Figure (a), we obtain a cycle-shaped attractor. The true attractor has dimension on the order of $\mathcal{O}(100)$ [8]. If the attractor is a high-dimensional sphere, then most of its mass concentrates around the equator. Therefore, when projected to low-dimension, the attractor will have a ring shape. We see that most of the points are located on the ring, while a few other points are located in the center. The points in the center have high dissipation, implying they are intermittent states. In Figure (b) we add the time axis. While the trajectory jumps around the cycle, we observe there is a rough period of 2000s. We perform the same PCA analysis on the training data, showing the same behavior. Furthermore, in Figure 11, we present invariant statistics for the KF equation ($Re = 40$), computed from a large number of samples from our approximation of the invariant measure. We find that the MNO consistently outperforms all other models in accurately capturing these statistics.

Derivative orders. Roughly speaking, vorticity is the derivative of velocity; velocity is the derivative of the stream function. Therefore we can denote the order of derivative of vorticity, velocity, and stream function as 2, 1, and 0 respectively. Combining vorticity, velocity, and stream function, with L^2, H^1, and H^2 loss, we have in total the order of derivatives ranging from 0 to 4. We observe, in general, it is best practice to keep the order of derivatives in the model at a number slightly higher than that of the target quantity. For example, as shown in Figure 5d, when querying the velocity (first-order quantity), it is best to use second-order (modeling velocity plus H^1 loss or modeling vorticity plus L^2 loss). This is further illustrated in Table 3 for the $Re = 500$ case. In general, using a higher order of derivatives as the loss will increase the power of the model and capture the invariant statistics more accurately. However, a higher-order of derivative means higher irregularity. It in turn requires a higher resolution for the model to resolve and for computing the discrete Fourier transform. This trade-off again suggests it is best to pick a Sobolev norm not too low or too high.

5 Discussion and future works

In this work, we propose a machine learning framework that trains from local data and predicts the global attractor and invariant statistics of chaotic systems. By enforcing dissipativity, we learn a Markov operator that empirically does not collapse or blow up over a long or infinite time horizon. Experiments also show the MNO predicts the attractor which shares the same distribution and statistics as the true function space trajectories. The simulations achieved by our MNO model have the potential to further our understanding of many physical phenomena and the mathematical models that underline them. Furthermore, the MNO shows great potential for modeling partially observed systems or studying systems that exhibit bifurcations both of which are of great interest for engineering applications. We also expect to generalize our theoretical work to show convergence to the global attractor and invariant measure by adapting ideas from the standard theory of numerical integrators [32].

This work provides a method for fast computation with applications to many scientific computing problems. Our methods have two main long-term impacts beyond the immediate interests of the scientific computing community. Since they are orders of magnitude faster than the traditional solvers currently employed in supercomputers, edge devices, and servers, the deployment of our methods significantly reduces the carbon footprint caused by scientific studies. Furthermore, the proposed methods are extremely flexible. The off-the-shelf usage of our methods allows scientists from a variety of disciplines, e.g. chemistry, biology, ecology, epidemiology, physics, and applied mathematics, to deploy them on their complex systems without the need to build elaborate numerical methods.

References

[1] Pantelis R Vlachas, Wonmin Byeon, Zhong Y Wan, Themistoklis P Sapsis, and Petros Koumoutsakos. Data-driven forecasting of high-dimensional chaotic systems with long short-term memory networks. *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 474(2213):20170844, 2018.

[2] Jaideep Pathak, Brian Hunt, Michelle Girvan, Zhixin Lu, and Edward Ott. Model-free prediction of large spatiotemporally chaotic systems from data: A reservoir computing approach. *Physical review letters*, 120(2):024102, 2018.

[3] Pantelis R Vlachas, Jaideep Pathak, Brian R Hunt, Themistoklis P Sapsis, Michelle Girvan, Edward Ott, and Petros Koumoutsakos. Backpropagation algorithms and reservoir computing in recurrent neural networks for the forecasting of complex spatiotemporal dynamics. *Neural Networks*, 2020.

[4] Huawei Fan, Junjie Jiang, Chun Zhang, Xingang Wang, and Ying-Cheng Lai. Long-term prediction of chaotic systems with machine learning. *Physical Review Research*, 2(1):012080, 2020.

[5] James Gleick. *Chaos: Making a new science*. Open Road Media, 2011.

[6] Warwick Tucker. The lorenz attractor exists. *Comptes Rendus de l'Académie des Sciences - Series I - Mathematics*, 328(12):1197–1202, 1999.

[7] Mark Holland and Ian Melbourne. Central limit theorems and invariance principles for Lorenz attractors. *Journal of the London Mathematical Society*, 76(2):345–364, 10 2007.

[8] Roger Temam. *Infinite-dimensional dynamical systems in mechanics and physics*, volume 68. Springer Science & Business Media, 2012.

[9] A. Stuart and A.R. Humphries. *Dynamical Systems and Numerical Analysis*. Cambridge Monographs on Applie. Cambridge University Press, 1998.

[10] AR Humphries and AM Stuart. Runge–kutta methods for dissipative and gradient dynamical systems. *SIAM journal on numerical analysis*, 31(5):1452–1485, 1994.

[11] Kai Fukami, Koji Fukagata, and Kunihiko Taira. Machine-learning-based spatio-temporal super resolution reconstruction of turbulent flows. *Journal of Fluid Mechanics*, 909, 2021.

[12] José I Cardesa, Alberto Vela-Martín, and Javier Jiménez. The turbulent cascade in five dimensions. *Science*, 357(6353):782–784, 2017.

[13] Gary J Chandler and Rich R Kerswell. Invariant recurrent solutions embedded in a turbulent two-dimensional kolmogorov flow. *Journal of Fluid Mechanics*, 722:554–595, 2013.

[14] Péter Koltai and Stephan Weiss. Diffusion maps embedding and transition matrix analysis of the large-scale flow structure in turbulent rayleigh–bénard convection. *Nonlinearity*, 33(4):1723, 2020.

[15] Jason J Bramburger, J Nathan Kutz, and Steven L Brunton. Data-driven stabilization of periodic orbits. *IEEE Access*, 9:43504–43521, 2021.

[16] Jacob Page, Michael P Brenner, and Rich R Kerswell. Revealing the state space of turbulence using machine learning. *Physical Review Fluids*, 6(3):034402, 2021.

[17] Victor Churchill and Dongbin Xiu. Deep learning of chaotic systems from partially-observed data. *arXiv preprint arXiv:2205.08384*, 2022.

[18] Zongyi Li, Nikola Kovachki, Kamyar Azizzadenesheli, Burigede Liu, Kaushik Bhattacharya, Andrew Stuart, and Anima Anandkumar. Neural operator: Graph kernel network for partial differential equations. *arXiv preprint arXiv:2003.03485*, 2020.

[19] Nikola B. Kovachki, Zongyi Li, Liu Burigede, Kamyar Azizzadenesheli, Kaushik Bhattacharya, Andrew M. Stuart, and Anima Anandkumar. Neural operator: Learning maps between function spaces. *In Preparation*.

[20] Georgy Anatolevich Sviridyuk. On the general theory of operator semigroups. *Russian Mathematical Surveys*, 49(4):45–74, 1994.

[21] Wojciech Marian Czarnecki, Simon Osindero, Max Jaderberg, Grzegorz Świrszcz, and Razvan Pascanu. Sobolev training for neural networks. *arXiv preprint arXiv:1706.04859*, 2017.

[22] Alex Beatson, Jordan Ash, Geoffrey Roeder, Tianju Xue, and Ryan P Adams. Learning composable energy surrogates for pde order reduction. *Advances in Neural Information Processing Systems*, 33, 2020.

[23] Alexander J Smits, Beverley J McKeon, and Ivan Marusic. High–reynolds number wall turbulence. *Annual Review of Fluid Mechanics*, 43:353–375, 2011.

[24] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*, pages 234–241. Springer, 2015.

[25] Xingjian Shi, Zhourong Chen, Hao Wang, Dit-Yan Yeung, Wai-Kin Wong, and Wang-chun Woo. Convolutional lstm network: A machine learning approach for precipitation nowcasting. *arXiv preprint arXiv:1506.04214*, 2015.

[26] Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555*, 2014.

[27] Grigorios Pavliotis and Andrew Stuart. *Multiscale Methods: Averaging and Homogenization*, volume 53. 01 2008.

[28] Jooyoung Park and Irwin W Sandberg. Universal approximation using radial-basis-function networks. *Neural computation*, 3(2):246–257, 1991.

[29] Zongyi Li, Nikola Kovachki, Kamyar Azizzadenesheli, Burigede Liu, Kaushik Bhattacharya, Andrew Stuart, and Anima Anandkumar. Fourier neural operator for parametric partial differential equations. *arXiv preprint arXiv:2010.08895*, 2020.

[30] Edward N Lorenz. Deterministic nonperiodic flow. *Journal of atmospheric sciences*, 20(2):130–141, 1963.

[31] Tian Qi Chen, Yulia Rubanova, Jesse Bettencourt, and David K Duvenaud. Neural ordinary differential equations. In *Advances in neural information processing systems*, pages 6571–6583, 2018.

[32] A. R. Humphries and A. M. Stuart. Runge–kutta methods for dissipative and gradient dynamical systems. *SIAM Journal on Numerical Analysis*, 31(5):1452–1485, 1994.

[33] Aly-Khan Kassam and Lloyd N. Trefethen. Fourth-order time-stepping for stiff pdes. *SIAM Journal on Scientific Computing*, 26(4):1214–1233, 2005.

[34] Gabriel J Lord, Catherine E Powell, and Tony Shardlow. *An introduction to computational stochastic PDEs*, volume 50. Cambridge University Press, 2014.

[35] Felix A Gers, Jürgen Schmidhuber, and Fred Cummins. Learning to forget: Continual prediction with lstm. 1999.

[36] Nikola Kovachki, Samuel Lanthaler, and Siddhartha Mishra. On universal approximation and error bounds for fourier neural operators. *Journal of Machine Learning Research*, 22:1–76, 2021.

[37] Andrew Stuart. Perturbation theory for infinite dimensional dynamical systems. 1995.

Checklist

The checklist follows the references. Please read the checklist guidelines carefully for information on how to answer these questions. For each question, change the default **[TODO]** to [Yes] , [No] , or [N/A] . You are strongly encouraged to include a **justification to your answer**, either by referencing the appropriate section of your paper or providing a brief inline description. For example:

- Did you include the license to the code and datasets? [Yes]
- Did you include the license to the code and datasets? [No] The code and the data are proprietary.
- Did you include the license to the code and datasets? [N/A]

Please do not modify the questions and only use the provided macros for your answers. Note that the Checklist section does not count towards the page limit. In your paper, please delete this instructions block and only keep the Checklist section heading above along with the questions/answers below.

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [Yes] See Section C
   (c) Did you discuss any potential negative societal impacts of your work? [No]
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [Yes] Statement of Theorem 1
   (b) Did you include complete proofs of all theoretical results? [Yes] See Appendix
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes] we plan to do so.
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes] Specified in Section 4
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [No] The error is consistent in the operator learning for PDEs.
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes] Specified in Section 4
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes]
   (b) Did you mention the license of the assets? [Yes] Supplement
   (c) Did you include any new assets either in the supplemental material or as a URL? [Yes] Supplement
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A]
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A]
5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]

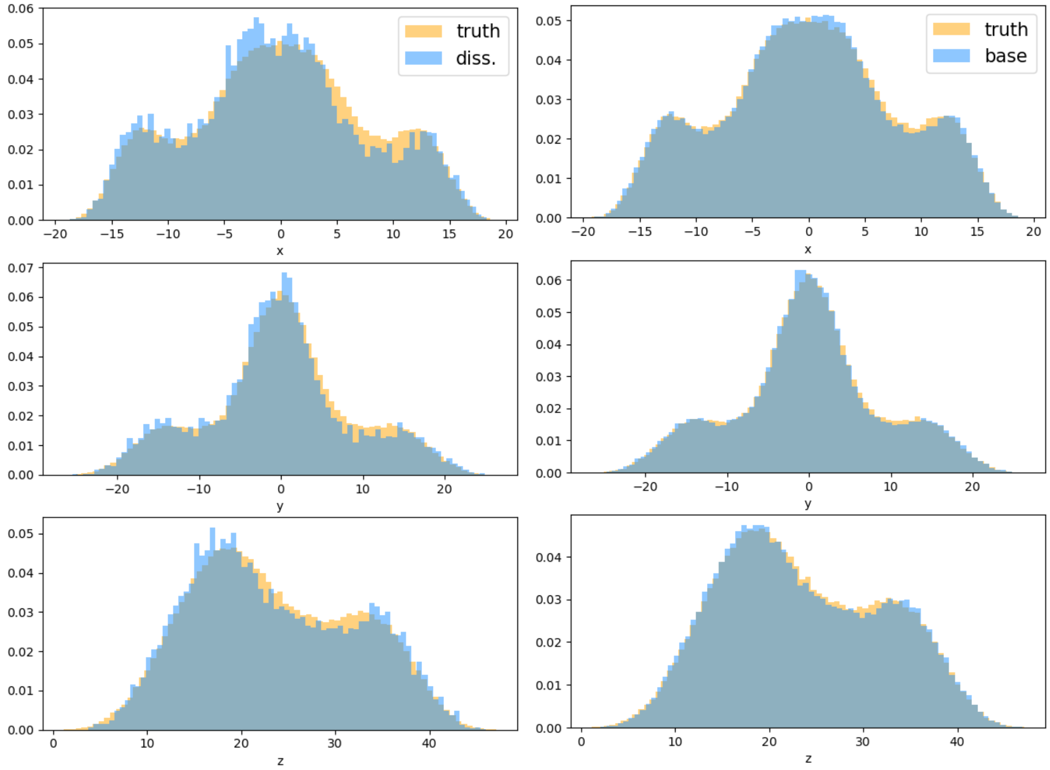


Figure 6: Lorenz-63 histograms of x, y, z-coordinates for the baseline network (no dissipativity enforced) and the MNO (dissipativity enforced).

A Full Experiments

In this section, we present the full numerical experiments on the Lorenz-63 system, the Kuramoto-Sivashinsky equation, and the Kolmogorov flow.

A.1 Lorenz-63

The Lorenz-63 system is three dimensional ODE system:

$$\begin{aligned} \dot{u}_x &= \alpha(u_y - u_x), \\ \dot{u}_y &= -\alpha u_x - u_y - u_x u_z, \\ \dot{u}_z &= u_x u_y - b u_z - b(r + \alpha). \end{aligned}$$

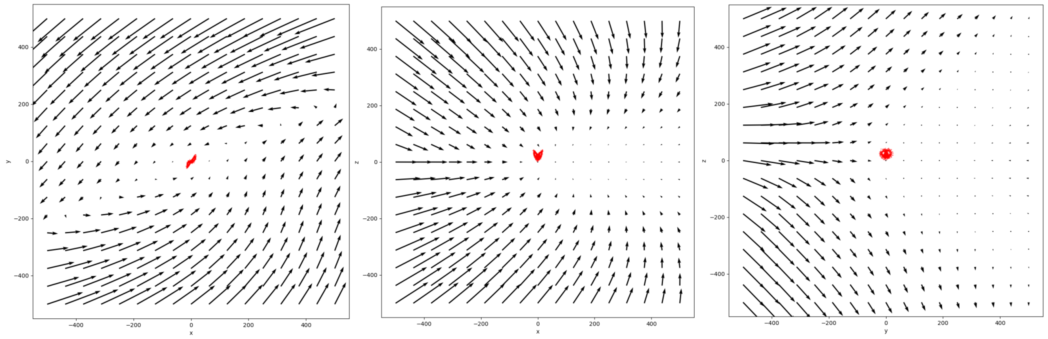
In this paper, we use the canonical parameters $(\alpha, b, r) = (10, 8/3, 28)$ [30].

Learning the Markov operator. We use a simple feedforward neural network with 6 hidden layers and 150 neurons per layer to learn the Markov operator for the Loren-63 system. We discretize the training trajectory into time-steps of 0.05 seconds.

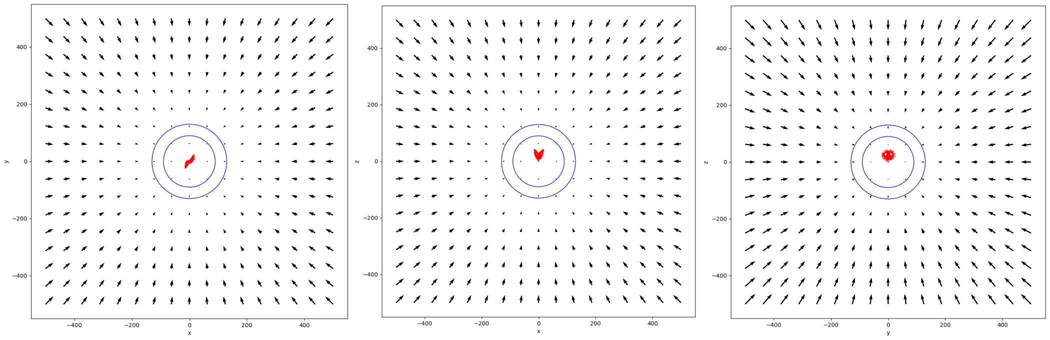
Enforcing dissipativity. We enforce dissipativity during training with the criterion described in eq. 5, with $\lambda = 0.5$ and ν being a uniform probability distribution supported on a shell around the origin with inner radius 90 and outer radius 130.

Evaluation metrics. We evaluate the effectiveness of our technique for enforcing dissipativity in three ways:

- **Relative L2 error:** We evaluate the learned Markov operators on relative (i.e., normalized) L2 error over 1 second by composing the model 20 times to gauge their ability to predict



(a) xy-plane at $z = 0$ (b) xz-plane at $y = 0$ (c) yz-plane at $x = 0$



(d) xy-plane at $z = 0$ (e) xz-plane at $y = 0$ (f) yz-plane at $x = 0$

Figure 7: Predicted flow field of dissipative network on Lorenz-63. The red points are training points on the attractor. Dissipativity is enforced uniformly within the blue shell.

Model	Per-step error	Per-second error	Dissipativity error
No diss. enforced	0.000570	0.0300	-
Diss. enforced	0.000564	0.0264	0.000667

Table 2: Relative L^2 error rates on the Lorenz-63 system. Per-step error is the error on the time-scale used in training. Per-second error is the error of the model composed with itself 21 times. Dissipativity error is the error of following the enforced dissipativity constraint on each step.

longer trajectories. As seen in Table 2, enforcing dissipativity does not cause a decrease in relative L2 error when compared to the baseline network, even though both models have the same number of parameters.

- **Vector field far from the attractor:** We also qualitatively evaluate the learned vector fields far from the attractor both with and without enforced dissipativity. We observe that enforcing dissipativity produces predictions that isotropically point towards the attractor, implying that the attractive properties of the Lorenz attractor are learned in the process. See Figure 7. Observe that the dissipative network is also dissipative outside the shell in which dissipativity was enforced during training.

- **Invariant statistics of the attractor:** To justify learning the Markovian map between time-steps, we also compare the coordinate histograms of the ground-truth and the learned attractors (with and without dissipativity enforced) after composing for 200000 time steps. Both models match the ground-truth coordinate histograms. Since the coordinate-wise histograms of the models matches the ground-truth, this indicates that the models are learning the distribution of the attractor and thus the invariant measure of the system.

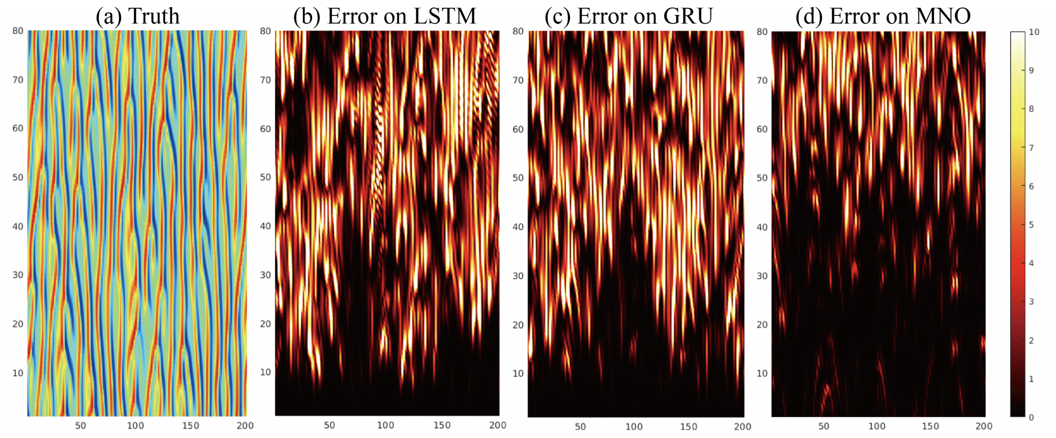

(a) Truth (b) Error on LSTM (c) Error on GRU (d) Error on MNO

Figure 8: Trajectory and error on the KS equation
The x-axis is the spatial domain; the y-axis is the temporal domain. The figure shows that LSTM and GRU start to diverge at $t = 20s$ while MNO is able to keep up with the exact trajectory until $t = 50s$.

A.2 Kuramoto-Sivashinsky equation

We consider the following one-dimensional Kuramoto-Sivashinsky equation,

$$\begin{aligned} \frac{\partial u}{\partial t} &= -u\frac{\partial u}{\partial x} - \frac{\partial^2 u}{\partial x^2} - \frac{\partial^4 u}{\partial x^4}, && \text{on } [0, L] \times (0, \infty) \\ u(\cdot, 0) &= u_0, && \text{on } [0, L] \end{aligned}$$

where $L = 32\pi$ or 64π and the spatial domain $[0, L]$ is equipped with periodic boundary conditions. We assume the initial condition $u_0 \in \dot{L}^2_{\text{per}}([0, L]; \mathbb{R})$, where $\dot{L}^2_{\text{per}}([0, L]; \mathbb{R})$ is the space of all mean zero L^2-functions that are periodic on $[0, L]$. Existence of the semigroup $S_t : \dot{L}^2_{\text{per}}([0, L]; \mathbb{R}) \to \dot{L}^2_{\text{per}}([0, L]; \mathbb{R})$ is established in [8, Theorem 3.1]. Data is obtained by solving the equation using the exponential time-differencing fourth-order Runge-Kutta method from [33]. Random initial conditions are generated according to a mean zero Gaussian measure with covariance $L^{-2/\alpha}\tau^{\frac{1}{2}(2\alpha-1)}(-\Delta + (\tau^2/L^2)I)^{-\alpha}$ where $\alpha = 2$, $\tau = 7$, and periodic boundary conditions on $[0, L]$; for details see [34].

Benchmarks for Kuramoto-Sivashinsky. We compare MNO with common choices of recurrent neural networks including the long short-term memory network (LSTM)[35] and gated recurrent unit (GRU)[26]. All models use the time-discretization $h = 1s$. The training dataset consists of 1000 different realizations of trajectories on the time interval $t \in [50, 200]$ (the first 50s is truncated so the dynamics reach the ergodic state), which adds up to $1000 \times 150 = 150,000$ snapshots in total. Another 200 realizations are generated for testing. Every single snapshot has the resolution 2048. We use Adam optimizer to minimize the relative L^2 loss with learning rate $= 0.001$, and step learning rate scheduler that decays by half every 10 epochs for 50 epochs in total. **LSTM and GRU:** having tested many different configurations, we choose the best hyper-parameters: the number of layers $= 1$, width $= 1000$. During the evaluation, we additionally provide 1000 snapshots as a warm-up of the memory. **MNO:** we choose 1-d Fourier neural operator as our base model with four Fourier layers with 20 frequencies per channel and width $= 64$. Experiments run on Nvidia V100 GPUs.

Accuracy with respect to the true trajectory. In general, MNO has a smaller per-step error compared to RNN. As shown in Figure 8, the MNO model captures a longer period of the exact trajectory compared to LSTM and GRU. LSTM and GRU start to diverge at $t = 20s$ while FNO is able to keep up with the exact trajectory until $t = 50s$.

Invariant statistics for the KS equations As shown in Figure 9, we present enormous invariant statistics for the KS equation. We use 150000 snapshots to train the MNO, LSTM, and GRU to model the evolution operator of the KS equation with $h = 1s$. We compose each model for $T = 1000$ time steps to obtain a long trajectory (attractor), and estimate various invariant statistics from them.

- **(a) Fourier spectrum**: the Fourier spectrum of the predicted attractor. All models are able to capture the Fourier modes with magnitude larger than $O(1)$, while MNO is more accurate on the tail.
- **(b) Spatial correlation**: the spatial correlation of the attractor, averaged in the time dimension. MNO is more accurate on the near-range correlation, but all models miss the long-range correlation.
- **(c) Auto-correlation of the Fourier mode**: the auto-correlation of the 10^{th} Fourier mode. Since the Fourier modes are nearly constant, the auto-correlation is constant too.
- **(d) Auto-correlation of the PCA mode**: the auto-correlation of the first PCA mode (with respect to the PCA basis of the ground truth data). The PCA mode oscillates around $[-40, 40]$, showing an ergodic state.
- **(e) Distribution of kinetic energy**: the distribution of kinetic energy with respect to the time dimension. MNO captures the distribution most accurately.
- **(f) Pixelwise distribution of velocity**: since the KS equation is homogeneous, we can compute the distribution of velocity with respect to pixels. MNO captures the pixelwise distribution most accurately too.

Choice of time discretization. We further study the choice of time steps h. As shown in Figure 4a, when the time steps are too large, the correlation is chaotic and hard to capture. But counter-intuitively, when the time steps are too small, the evolution is also hard to capture. In this case, the input and output of the network will be very close, and the identity map will be a local minimum. An easy fix is to use MNO to learn the time-derivative or residual. This is shown in the figure, where the residual model (blue line) has a better per-step error and accumulated error at smaller h. When the time step is large, there is no difference in modeling the residual. This idea can generalize to other integrators as an extension of Neural ODEs to PDEs [31].

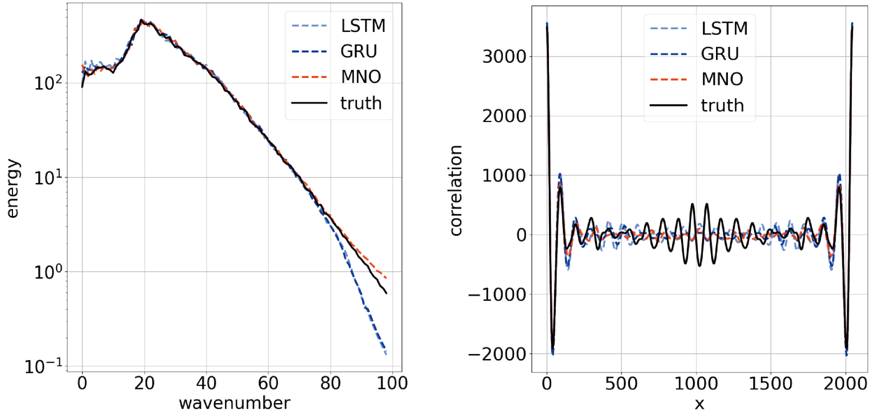
A.3 Kolmogorov Flow

We consider the two-dimensional Navier-Stokes equation for a viscous, incompressible fluid,

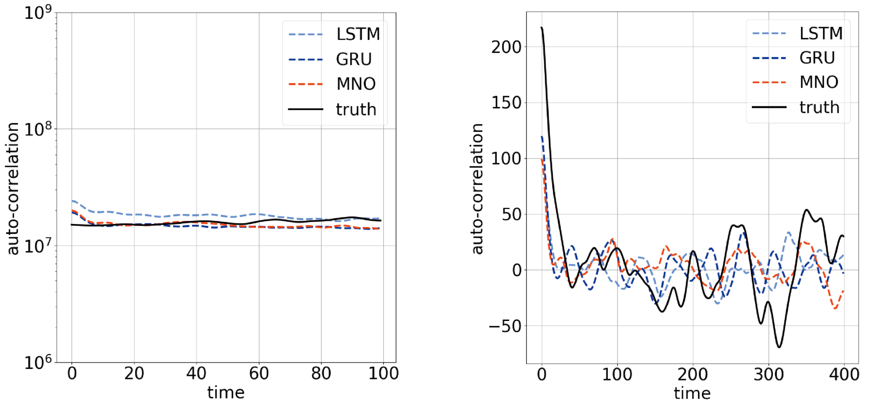
$$\begin{aligned}
\frac{\partial u}{\partial t} &= -u \cdot \nabla u - \nabla p + \frac{1}{Re}\Delta u + \sin(ny)\hat{x}, \qquad && \text{on } [0, 2\pi]^2 \times (0, \infty) \\
\nabla \cdot u &= 0 && \text{on } [0, 2\pi]^2 \times [0, \infty) \\
u(\cdot, 0) &= u_0 && \text{on } [0, 2\pi]^2
\end{aligned}$$

where u denotes the velocity, p the pressure, and $Re > 0$ is the Reynolds number. The domain $[0, 2\pi]^2$ is equipped with periodic boundary conditions. The specific choice of forcing $\sin(ny)\hat{x}$ constitutes a Kolmogorov flow; we choose $n = 4$ in all experiments. We define $\mathcal{U}$ to be the closed subspace of $L^2([0, 2\pi]^2; \mathbb{R}^2)$, $\mathcal{U} = \big\{u \in \dot{L}^2_{\text{per}}([0, 2\pi]^2; \mathbb{R}^2) \,:\, \nabla \cdot u = 0\big\}$ and assume $u_0 \in \mathcal{U}$. We define the vorticity $w = (\nabla \times u)\hat{z}$ and the stream function f as the solution to the Poisson equation $-\Delta f = w$. Existence of the semigroup $S_t : \mathcal{U} \to \mathcal{U}$ is established in [8, Theorem 2.1]. We denote turbulence kinetic energy (TKE) $\langle (u - \bar{u})^2 \rangle$, and dissipation $\epsilon = \langle w^2 \rangle / Re$. Data is obtained by solving the equation in vorticity form using the pseudo-spectral split step method from [13]. Random initial conditions are generated according to a mean zero Gaussian measure with covariance $7^{3/2}(-\Delta + 49I)^{-2.5}$ with periodic boundary conditions on $[0, 2\pi]^2$.

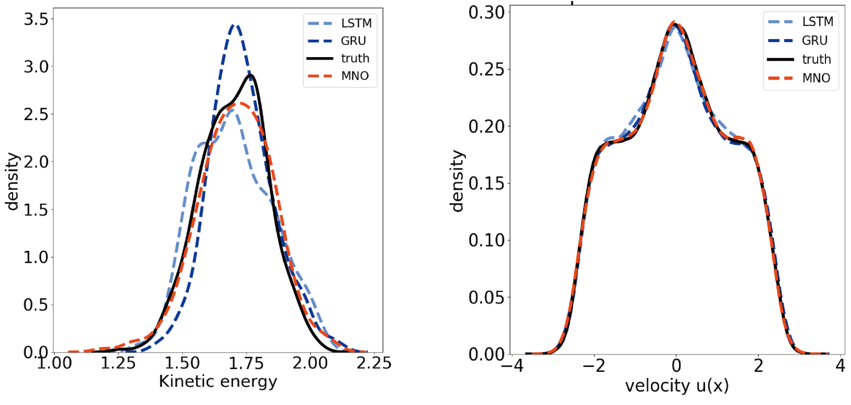
Benchmarks for the 2d Kolmogorov flow. We compare MNO with common standard two-dimensional dynamic models including U-Net[24] and LSTM-CNN[25] on modeling the vorticity w in the relatively non-turbulent $Re = 40$ case. We choose the discretization $h = 1s$. The training dataset consists of 180 realizations of trajectories on time interval $t \in [100, 500]$ (the first 100 seconds are discarded) which adds up to $180 \times 400 = 72,000$ snapshots in total. Another 20 realizations are generated for testing. Each single snapshot has resolution 64×64. We use the Adam optimizer to minimize the relative L^2 loss with learning rate $= 0.0005$, and step learning rate scheduler that decays by half every 10 epochs for 50 epochs in total. **U-Net:** we use five layers of convolution and deconvolution with width from 64 to 1024. **LSTM-CNN:** we use one layer of LSTM with width $= 64$. **MNO:** we parameterize the 2-d Fourier neural operator consists of four Fourier layers with 20 frequencies per channel and width $= 64$.


(a) Fourier spectrum

(b) spatial correlation


(c) auto-correlation of Fourier mode

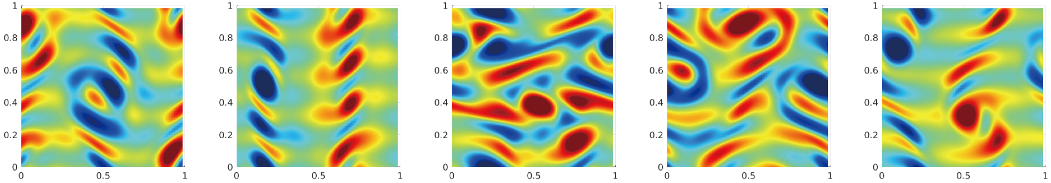
(d) auto-correlation of PCA mode


(e) distribution of kinetic energy

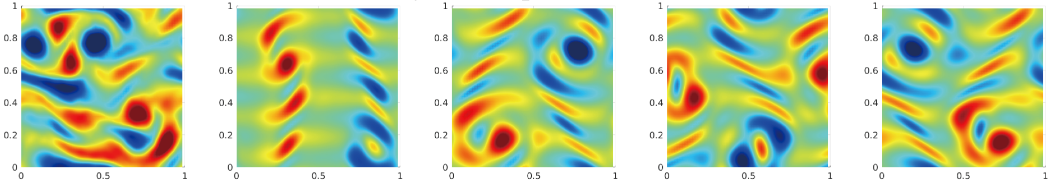
(f) pixelwise distribution

Figure 9: Invariant statistics for the KS equation

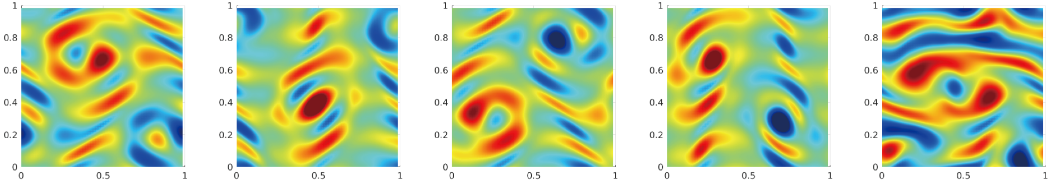
(a) vorticity, Re40, ground truth



(b)vorticity, Re40, prediction with L2 loss



(c)vorticity, Re40, prediction with H1 loss



(d)vorticity, Re40, prediction with H2 loss

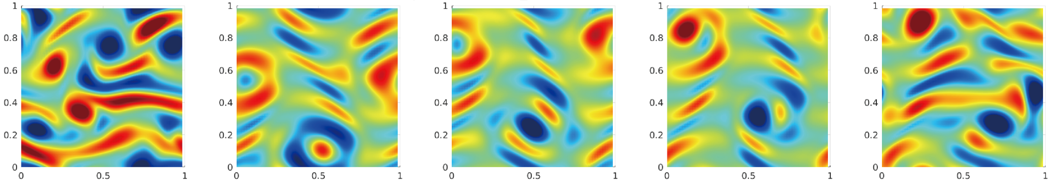


Figure 10: visualization of the KF equation, Re40

Enforcing dissipativity. We enforce dissipativity during training with the criterion described in eq. 5, with $\lambda = 0.5$ and ν being a uniform probability distribution supported on a shell around the origin.

Accuracy with respect to various norms. MNO shows near one order of magnitude better accuracy compared to U-Net and LSTM-CNN. As shown in Table 1, we train each model using the balanced $L^2 (= H^0)$, H^1, and H^2 losses, defined as the sum of the relative L^2 loss grouped by each order of derivative. And we measure the error with respect to the standard (unbalanced) norms. The MNO with H^2 loss consistently achieves the smallest error on vorticity on all of the L^2, H^1, and H^2 norms. However, L^2 loss achieves the smallest error on the turbulence kinetic energy (TKE); H^1 loss achieves the smallest error on the dissipation ϵ.

The NS equation with Re40 is shown in Figure 10. (a) show the ground truth data of vorticity field, each column represents a snapshot at $t = 100s$ with a different initial condition. (b), (c), (d) show the predicted trajectory of MNO on vorticity, using L^2, H^1, and H^2 losses respectively. We are able to generate a long trajectory with the MNO model. The five columns represent $t = 1000s, 2000s, 3000s, 4000s, 5000s$ respectively. As shown in the figure, the predicted trajectories (b) (c) (d) share the same behaviors as in the ground truth (a). It indicates the MNO model is stable.

Visualizing the attractor generated by MNO. We compose MNO 10000 times to obtain the global attractor, and we compute the PCA (POD) basis of these 10000 snapshots and project them onto the first two components. As shown in Figure (a), we obtain a cycle-shaped attractor. The true attractor has a degree of freedom around $O(100)$ [8]. If the attractor is a high-dimensional sphere, then most of the mass concentrates around its equator. Therefore, when projected to low-dimension, the attractor will have the shape of a ring. Most of the points are located on the ring, while a few

Model	training	loss	(order)	error on f	error on u	error on w
Stream function f	L^2 loss	0.0379	(0^{th} order)	0.0383	0.2057	2.0154
Stream function f	H^1 loss	0.0512	(1^{st} order)	0.0268	0.0769	0.3656
Stream function f	H^2 loss	0.0973	(2^{nd} order)	0.0198	0.0522	0.2227
Velocity u	L^2 loss	0.0688	(1^{st} order)	0.0217	0.0691	0.3217
Velocity u	H^1 loss	0.1246	(2^{nd} order)	**0.0170**	0.0467	0.1972
Velocity u	H^2 loss	0.2662	(3^{rd} order)	0.0178	0.0482	0.1852
Vorticity w	L^2 loss	0.1710	(2^{nd} order)	0.0219	**0.0415**	0.1736
Vorticity w	H^1 loss	0.3383	(3^{rd} order)	0.0268	0.0463	**0.1694**
Vorticity w	H^2 loss	0.4590	(4^{th} order)	0.0312	0.0536	0.1854

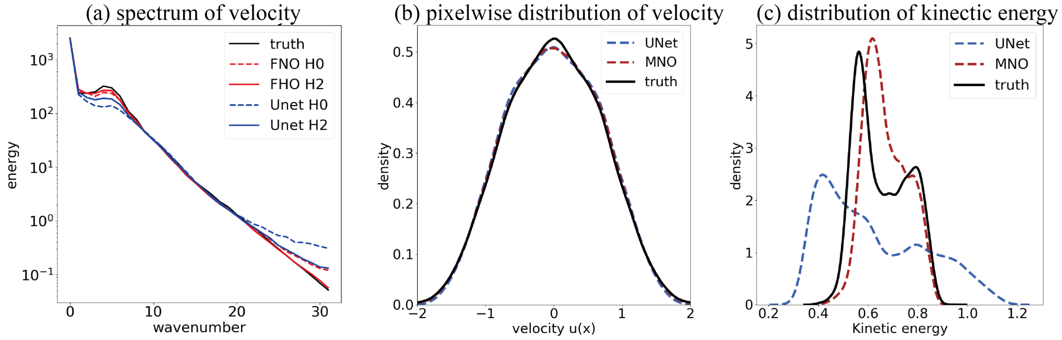
Table 3: Vorticity, velocity, and stream function for the Kolmogorov flow with $Re = 500$

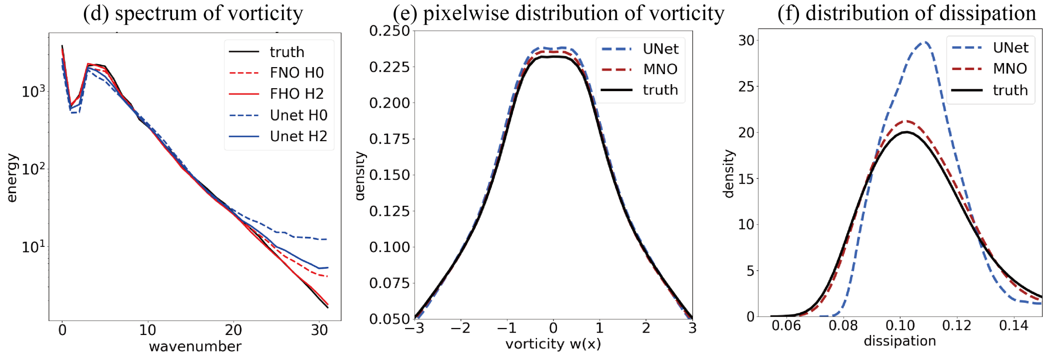
other points are located in the center. The points in the center have high dissipation, implying they are intermittent states. In Figure (b) we add the time axis. While the trajectory jumps around the cycle, we observe there is a rough period of 2000s. We perform the same PCA analysis on the training data, which shows the same behavior.

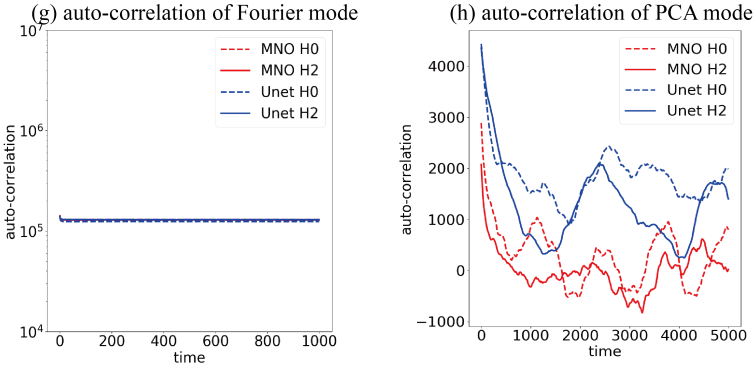
Invariant statistics. Similarly, we present enormous invariant statistics for the NS equation (Re40), as shown in Figure 11,. We use 72000 snapshots to train the MNO, UNet, and ConvLSTM to model the evolution operator of the KS equation with $h = 1s$. We compose each model for $T = 10000$ time steps to obtain a long trajectory (attractor), and estimate various invariant statistics from them.

- **(a, d) Fourier spectrum of velocity and vorticity**: the Fourier spectrum of the predicted attractor. Again, all models are able to capture the Fourier modes with magnitude larger than $O(1)$, while MNO is more accurate on the tail. Using the Sobolev norm further helps to capture the tail.
- **(b, e) Pixelwise distribution of velocity and vorticity**: All models preserve the pixelwise distribution.
- **(c, f) Distribution of kinetic energy and dissipation rate**: the distribution of kinetic energy with respect to the time dimension. MNO captures the distribution most accurately.
- **(g) Auto-correlation of the Fourier mode**: the auto-correlation of the 10^{th} Fourier mode. Since the Fourier modes are nearly constant, the auto-correlation is constant too. Notice it is very expensive to generate long-time ground truth data, so the figure does not include the ground truth. However, it is easy to obtain the auto-correlation by MNO.
- **(h) Auto-correlation of the PCA mode**: the auto-correlation of the first PCA mode (with respect to the PCA basis of the ground truth data). The PCA mode oscillates around $[-1000, 1000]$, showing an ergodic state. The UNet oscillates around $[0, 2000]$.
- **(i) Spatial correlation**: the spatial correlation of the attractor, averaged in the time dimension. The four columns represent the truth and MNO with different losses. As seen from the figure, there is a wave pattern matching the force term $\sin(4y)$.

Order of derivatives. Roughly speaking, vorticity is the derivative of velocity; velocity is the derivative of the stream function. Therefore we can denote the order of derivative of vorticity, velocity, and stream function as 2, 1, and 0 respectively. Combining vorticity, velocity, and stream function, with L^2, H^1, and H^2 loss, we have in total the order of derivatives ranging from 0 to 4. We observe, in general, it is best practice to keep the order of derivatives in the model at a number slightly higher than that of the target quantity. For example, as shown in Figure 5d, when querying the velocity (first-order quantity), it is best to use second-order (modeling velocity plus H^1 loss or modeling vorticity plus L^2 loss). This is further illustrated in Table 3. In general, using a higher order of derivatives as the loss will increase the power of the model and capture the invariant statistics more accurately. However, a higher-order of derivative means higher irregularity. It in turn requires a higher resolution for the model to resolve and for computing the discrete Fourier transform. This trade-off again suggests it is best to pick a Sobolev norm not too low or too high.


(a) spectrum of velocity (b) pixelwise distribution of velocity (c) distribution of kinectic energy


(d) spectrum of vorticity (e) pixelwise distribution of vorticity (f) distribution of dissipation


(g) auto-correlation of Fourier mode (h) auto-correlation of PCA mode

(i) spatial correlation of velocity

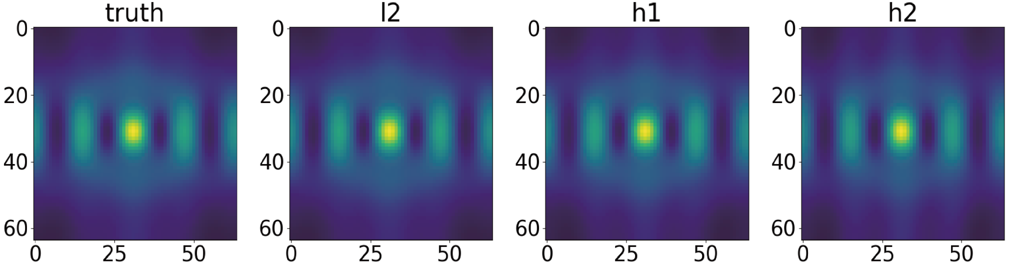


Figure 11: Invariant statistics for the NS equation

B Theoretical background

B.1 Markov operators and the semigroup

Since the system (1) is autonomous, that is, F does not explicitly depend on time, under the well-posedness assumption, we may define, for any $t \in [0, \infty)$, a Markov operator $S_t : \mathcal{U} \to \mathcal{U}$ such that $u(t) = S_t u(0)$. This map satisfies the properties

1. $S_0 = I$,
2. $S_t(u_0) = u(t)$,
3. $S_t(S_s(u_0)) = u(t+s)$,

for any $s, t \in [0, \infty)$ and any $u_0 \in \mathcal{U}$ where I denotes the identity operator on $\mathcal{U}$. In particular, the family $\{S_t : t \in [0, \infty)\}$ defines a semigroup of operators acting on $\mathcal{U}$. Our goal is to approximate a particular element of this semigroup associated to some fixed time step $h > 0$ given observations of the trajectory from (1). We build an approximation $\hat{S}_h : \mathcal{U} \to \mathcal{U}$ such that

$$\hat{S}_h \approx S_h. \tag{9}$$

B.2 Proof of Theorem 1

Proof of Theorem 1. Since S_h is continuous and K is compact, the set

$$R = \bigcup_{l=0}^{n} S_h^l(K)$$

is compact. Therefore, there exist a set of representers $\varphi_1, \varphi_2, \dots \in R$ such that

$$\lim_{m\to\infty} \sup_{v\in R} \inf_{u\in R_m} \|u - v\|_{\mathcal{U}} = 0$$

where $R_m = \text{span}\{\varphi_1, \dots, \varphi_m\}$. For any $m \in \mathbb{N}$, let $P_m : \mathcal{U} \to R_m$ denote a projection of U to R_m. Since R is compact, the set

$$P = R \bigcup \left(\bigcup_{m=1}^{\infty} P_m(R) \right)$$

is compact [19, Lemma 14]. Since S_h is locally Lipschitz and P is compact, there exists a constant $C = C(P) > 0$ such that

$$\|S_h(u_1) - S_h(u_2)\|_{\mathcal{U}} \le C \|u_1 - u_2\|_{\mathcal{U}}, \qquad \forall u_1, u_2 \in P.$$

Without loss of generality, assume $C \neq 1$ and define

$$M = \left(C^n + \frac{1 - C^n}{1 - C} \right)^{-1}.$$

By the universal approximation theorem for neural operators [19, Theorem 4] or [36, Theorem 2.5], there exists a neural operator $\hat{S}_h : \mathcal{U} \to \mathcal{U}$ such that

$$\sup_{u_0 \in P} \|S_h(u_0) - \hat{S}_h(u_0)\|_{\mathcal{U}} < \epsilon M.$$

Perusal of the proof of the universal approximation theorem for neural operators shows that $\hat{S}_h$ can be chosen so that $\hat{S}_h(P) \subseteq R_m$ for some $m \in \mathbb{N}$ large enough, therefore $\hat{S}_h(P) \subseteq P$. Let $u_0 \in K$, then the triangle inequality implies

$$\begin{aligned}
\|u(nh) - \hat{S}_h^n(u_0)\|_{\mathcal{U}} &= \|S_h^n(u_0) - \hat{S}_h^n(u_0)\|_{\mathcal{U}} \\
&= \|S_h(S_h^{n-1}(u_0)) - \hat{S}_h(\hat{S}_h^{n-1}(u_0))\|_{\mathcal{U}} \\
&\le \|S_h(S_h^{n-1}(u_0)) - S_h(\hat{S}_h^{n-1}(u_0))\|_{\mathcal{U}} + \|S_h(\hat{S}_h^{n-1}(u_0)) - \hat{S}_h(\hat{S}_h^{n-1}(u_0))\|_{\mathcal{U}} \\
&\le C\|S_h^{n-1}(u_0) - \hat{S}_h^{n-1}(u_0)\|_{\mathcal{U}} + \epsilon M.
\end{aligned}$$

By the discrete time Grönwall lemma,

$$\begin{aligned}\|u(nh) - \hat{S}_h^n(u_0)\|_{\mathcal{U}} &\leq C^n \|S_h(u_0) - \hat{S}_h(u_0)\|_{\mathcal{U}} + \epsilon M \left(\frac{1 - C^n}{1 - C}\right) \\ &< \epsilon M \left(C^n + \frac{1 - C^n}{1 - C}\right) \\ &= \epsilon\end{aligned}$$

which completes the proof.

□

C Discussion and future work

In this work, we learn MNO from only local data and compose it to obtain the global attractor of chaotic systems, and by explicitly enforcing dissipativity, we empirically show that MNO predictions do not blow up or collapse even in the long-time horizon, while still achieving relatively low error running several orders of magnitude faster than traditional methods.

MNO has two major limitations. First, it assumes the target system is approximately Markovian. If the system is heavily path-dependent, then the MNO framework does not directly apply. Second, although we develop an approximation theorem for finite period, it does not hold for infinite time horizon.

As discussed previously, it is infeasible to track the exact trajectory of chaotic systems on an infinite time horizon. Even very small errors will accumulate in each step, and eventually cause the simulation to diverge from the true trajectory. However, it is possible to track the attractor of the system. An attractor is absorbing. If the simulated trajectory only makes a small error, the attractor will absorb it back, so that the simulated trajectory will never diverge from the true attractor. Therefore, it is possible to have the simulated trajectory capture the true attractor.

To obtain an infinite-time approximation error bound is non-trivial. Previously, [9, 10] (cf. Theorem 3.12) show a result for (finite-dimensional) ODE systems. If the system is Lipschitz then there exists a numerical simulation that forms a dissipative dynamical system that does not blow up or collapse. And the the simulated attractor $\mathcal{A}_h$ approximates the true attractor $\mathcal{A}$ with the time step h

$$dist(\mathcal{A}_h, \mathcal{A}) \to 0, \quad \text{as } h \to 0$$

To generalize such theorem to Markov neural operator (MNO), we need to overcome two difficulties (1) generalize the formulation from (finite-dimensional) ODE systems to (infinite-dimensional) PDE systems, and (2) show MNO can obtain a sufficient error rate with respect to the time step h.

The first aspect requires extending the theory from finite dimension to infinite dimension, which is non-trivial since the operator F in (1) is not compact or bounded. This makes it hard to bound the error with respect to the attractor [37]. The second aspect requires to formulate MNO slightly differently. In the current formulation, the evolution operator is chosen for a fixed time step h. To achieve $O(h)$ error we need to formulate the evolution operator continuously for infinitesimal h. Especially, for a semi-linear PDE system

$$\frac{du}{dt} + Au = F(u)$$

where A is a linear, self-adjoint operator and F is a continuous but nonlinear operator (this formulation includes the KS and NS equations). The evolution can be written as

$$u(t+h) = e^{-Ah}u(t) + \int_0^h e^{-A(h-s)} F(u(t+s))ds$$

Where $\Phi(u(t), A, t) := \int_0^h e^{-A(h-s)} F(u(s))ds$ is bounded despite F is not. If one can approximate $\Phi(u(t), A, h)$ by a neural operator, then MNO can potentially achieve the needed error rate. This shows hope to obtain an approximation error bound for infinite time zero. We leave this as a promising future direction.