# "Deep learning for universal linear embeddings of nonlinear dynamics."

Mollicone Alessandro
Verrando Giuliano


SAPIENZA
UNIVERSITÀ DI ROMA

# Koopman operators

The object of discussion are **discrete – time** dynamical systems

$$\mathbf{x}_{k+1} = \mathbf{F}(\mathbf{x}_k)$$

**State** of the system

Dynamics of the system that maps the state **forward in time**

Koopman provided an alternative description of dynamical systems in terms of the evolution of functions in the **Hilbert space of possible measurements of the the state,** g(x).

$$\mathcal{K}\mathbf{g} := \mathbf{g} \circ \mathbf{F} \quad \text{i.e.} \quad \mathcal{K}\mathbf{g}(\mathbf{x}_k) = \mathbf{g}(\mathbf{x}_{k+1})$$

$\mathcal{K}$ **is the so called Koopman operator**

- Koopman operator is an infinitedimensional **linear** operator

- The eigenfunctions of the Koopman operator provide **intrinsic coordinates** that globally linearize the dynamics.

- Obtaining **finite-dimensional approximations** of the infinite-dimensional Koopman operator has proven **challenging** in practical applications possible approach is to seek the eigenfunctions of the Koopman operator directly

- A possible approach is to seek the **eigenfunctions of the Koopman operator directly**

$$\varphi(\mathbf{x}_{k+1}) = \mathcal{K}\varphi(\mathbf{x}_k) = \lambda\varphi(\mathbf{x}_k).$$

These eigenfunctions are guaranteed to **span an invariant subspace**, and the Koopman operator will yield a **matrix** when restricted to this subspace: the evolution of the system is described in terms of a **linear operator**

In practice, Koopman eigenfunctions are **very difficult to obtain**

The aim is to exploit **deep learning** to find the representation of the smallest number of Koopman eigenfunctions needed to describe the dynamics

# Data handling

- Several example systems:
  A simple model with a discrete spectrum
  Nonlinear pendulum
  Fluid on attractor
  Fluid in a cylindrical box
- Datasets by solving the systems of differential equations in **MATLAB** using the ode45 solver.

For each dynamical system, they choose **5000** initial conditions for the **test set, 5000** for the **validation set**, and **5000 - 20000** for the **training set**.

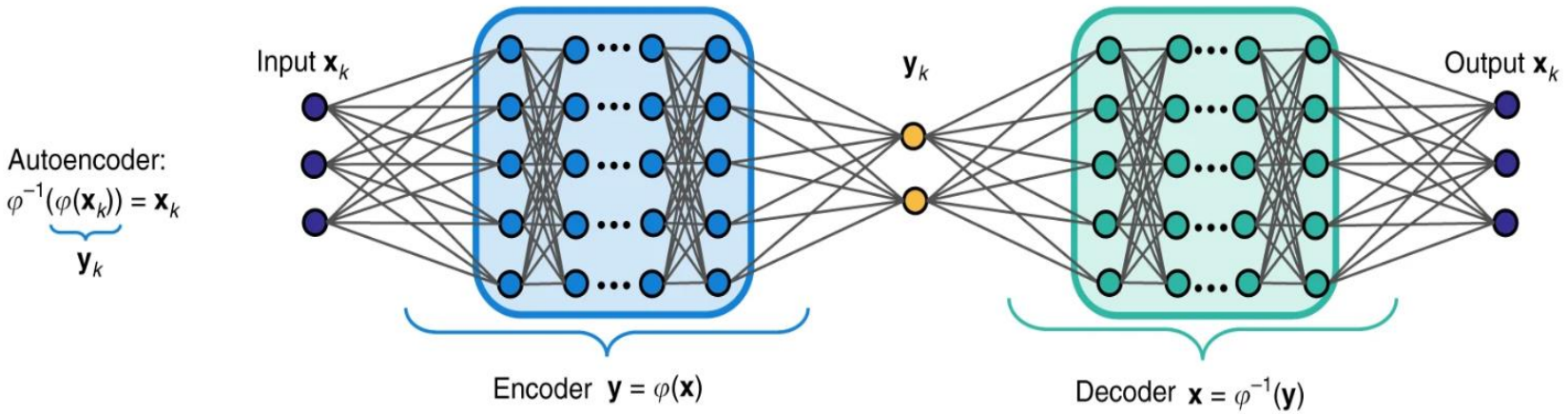
For each initial condition, the differential equations were solved for some **time span**.

# Network architecture

The aim is to identify only a ***few key*** intrinsic coordinates $\mathbf{y} = \varphi(\mathbf{x})$ , i.e. few Koopman eigenfunctions

Intrinsic coordinates which are useful in order to recostruct the dynamics the **state x can be recovered with the inverse**: $\mathbf{x} = \varphi^{-1}(\mathbf{y})$

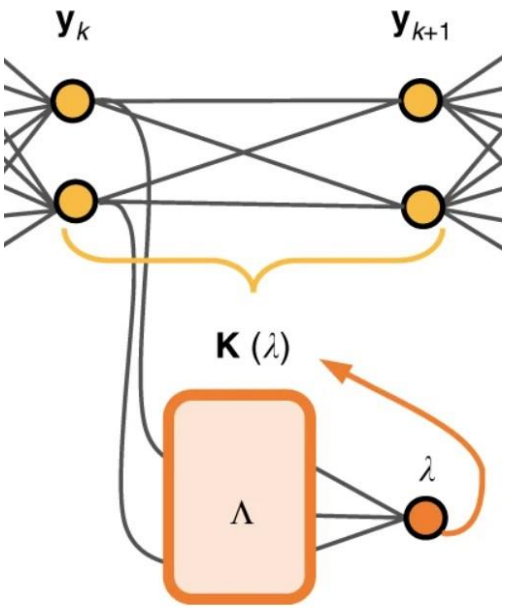
This is archieved by an **autoencoder**




The dimension of the Koopman subspace and the number of (linear) hidden layers in the encoder and in the decoder are a **hyperparameter** and depends on the specific example. Each hidden layer is followed by a **ReLU activation**. The output layers of the encoder and the decoder are **linear.**

# Auxiliary network

- This framework is generalized to include a broad class of nonlinear systems that exhibit a **continuous eigenvalue spectrum λ**.
- The continuous spectrum spoils the simple Koopman descriptions, as **there is not a straightforward finite approximation** in terms of a small number of eigenfunctions.




- To deal with this problem, they allowed the eigenvalues of the matrix **K** to vary, **parametrized by the function λ=Λ(y)**, which is **learned by an auxiliary network.**
- The eigenvalues **λ±=μ±iω** are then used to **parametrize block-diagonal K(μ,ω)**.
- They built **separate auxiliary network**s, one for the complex conjugate pair of eigenvalues and one for the the real eigenvalue.
- Linear hidden layers whose number and dimension vary according to the specific system and the output layer is linear.

- For each pair of complex eigenvalues, **K** has a Jordan block of the form

$$B(\mu,\omega) = \exp(\mu\Delta t)\begin{bmatrix} \cos(\omega\Delta t) & -\sin(\omega\Delta t) \\ \sin(\omega\Delta t) & \cos(\omega\Delta t) \end{bmatrix}.$$

- In the problems with Koopman subspace dimension = 2, **circular symmetry** is enforced in the eigenfunction coordinates: obtained parametrizing the eigenvalues by the radius.

# Loss function

$$\mathcal{L} = \alpha_1(\mathcal{L}_{recon} + \mathcal{L}_{pred}) + \mathcal{L}_{lin} + \alpha_2\mathcal{L}_{\infty} + \alpha_3||\mathbf{W}||_2^2$$

$$\mathcal{L}_{recon} = ||\mathbf{x}_1 - \varphi^{-1}(\varphi(\mathbf{x}_1))||_{\text{MSE}}$$

**Reconstruction** loss

$$\mathcal{L}_{pred} = \frac{1}{S_p}\sum_{m=1}^{S_p}||\mathbf{x}_{m+1} - \varphi^{-1}(K^m\varphi(\mathbf{x}_1))||_{\text{MSE}}$$

Prediction loss

$$\mathcal{L}_{lin} = \frac{1}{T-1}\sum_{m=1}^{T-1}||\varphi(\mathbf{x}_{m+1}) - K^m\varphi(\mathbf{x}_1)||_{\text{MSE}}$$

**Linearity** of dynamics Loss

$$\mathcal{L}_{\infty} = ||\mathbf{x}_1 - \varphi^{-1}(\varphi(\mathbf{x}_1))||_{\infty} + ||\mathbf{x}_2 - \varphi^{-1}(K\varphi(\mathbf{x}_1))||_{\infty}$$

**Norm-infinity** loss

- The ℓ2 regularization on the weights **W** designed to reduce overfitting was non included. The weight reduction during the training was obtained by adding a **weight decay** to Adam.
- The weights **α1**, **α2**, and **α3** are hyperparameters which are different for each example. The integer **Sp** is a hyperparameter for how many steps to check in the prediction loss.

# Training

- **Weights** are randomly initialized using a uniform distribution in the range **[−s,s]** for **s=1/√a**, where **a** is the dimension of the **input of the layer**. Each **bias vector b** is initialized **to 0**. The **learning rate** for the Adam optimizer is **0.001**. We also use **early stopping**; for each model, at the end of training, we resume the step with the lowest validation error.

# Results

In order to test our model we considered the following plots:

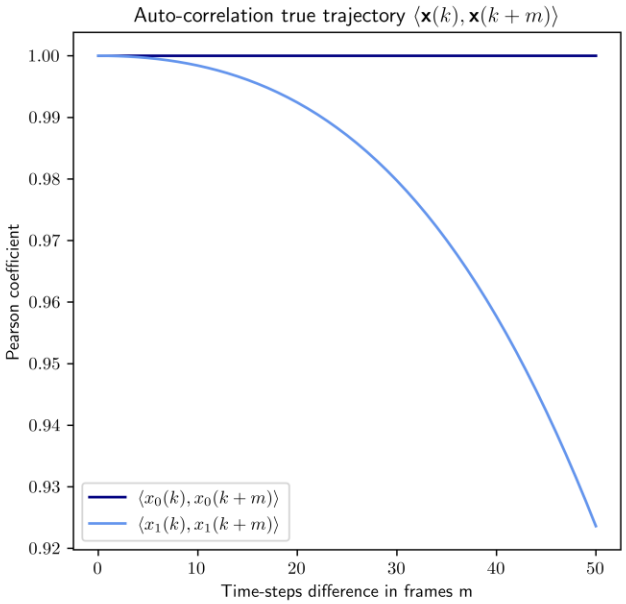
1) The correlation **⟨x(k),x(k+m)⟩** at different discrete time steps m (evolution provided by the network by applying m times the Koopman matrix) and fixed k.
2) The correlation **⟨x(k+m),y(k+m)⟩** at different discrete time steps m (evolution provided by the network by applying m times the Koopman matrix) and fixed k.
3) The correlation **⟨y(k),y(k+m)⟩** at different discrete time steps m (evolution provided by the network by applying m times the Koopman matrix) and fixed k.
4) **$x_i(k)$ vs $x_j(k)$** at different time steps k (evolution provided by the network by the application of the Koopman matrix).

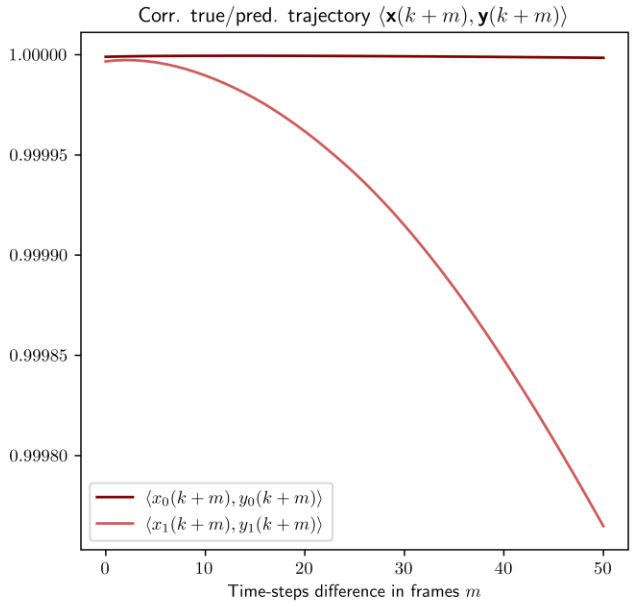
The results are then compared with those found with those provided by the authors of the paper and found with numerical integration.

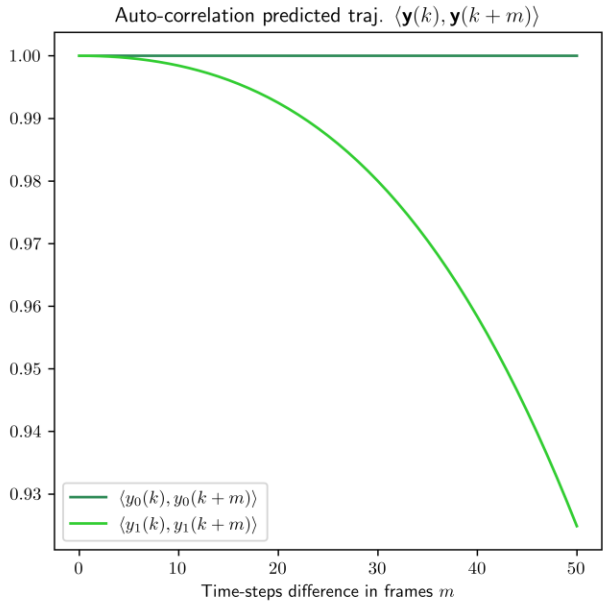
# Results

## Discrete spectrum

### Correlations and auto-correlations

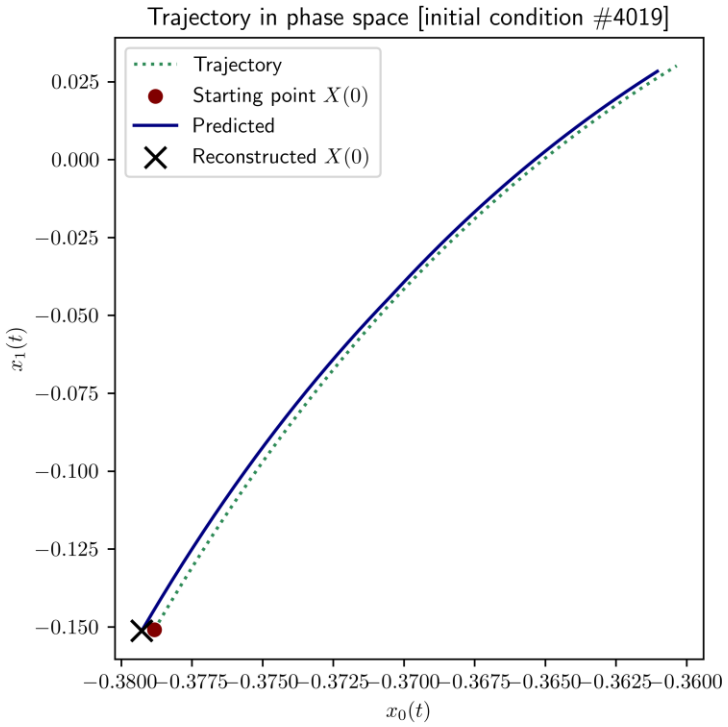


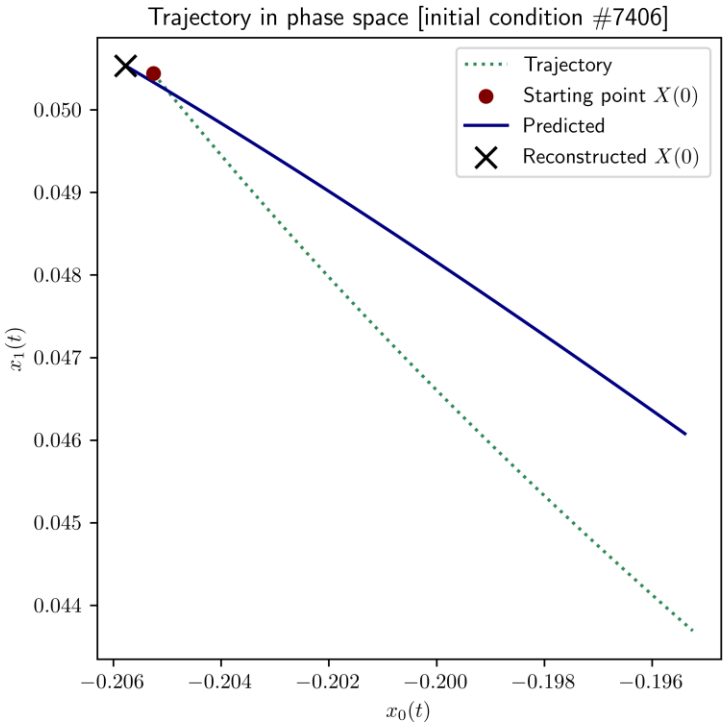



# Trajectories

BEST and WORST

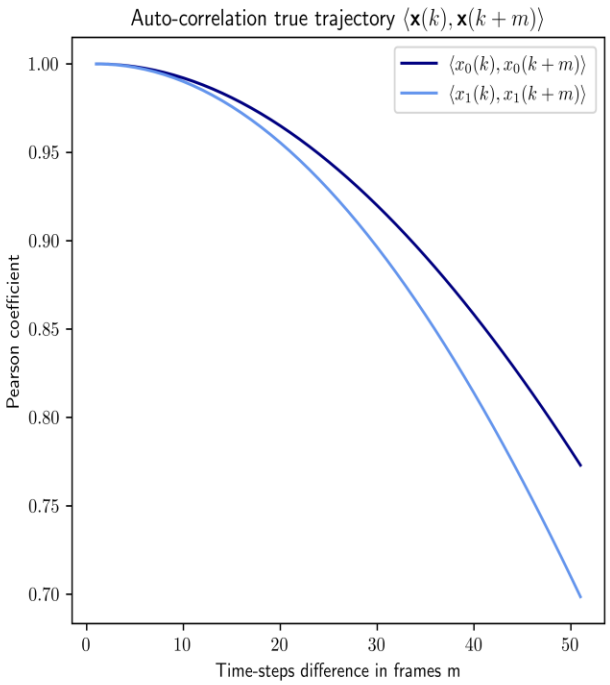


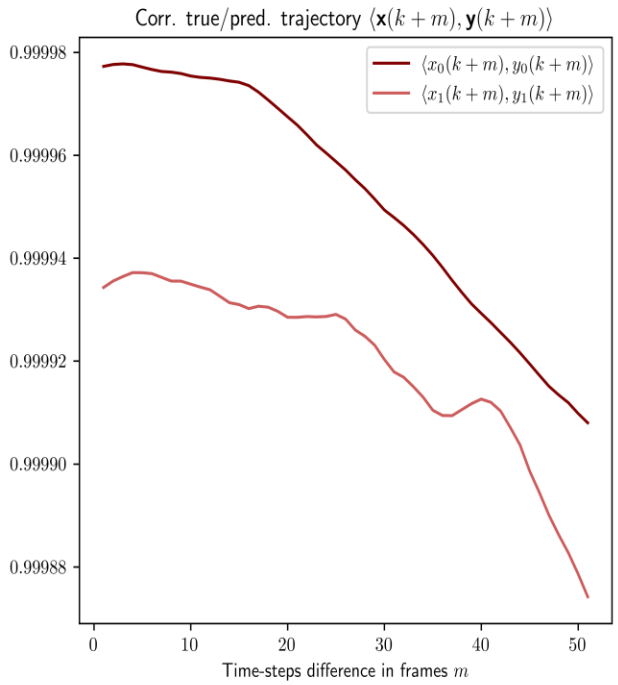



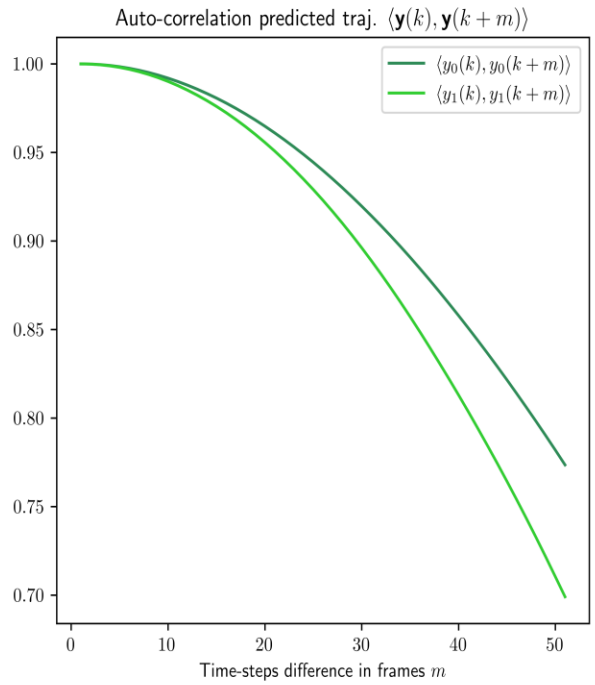
# Results

## Pendulum

### Correlations and auto-correlations

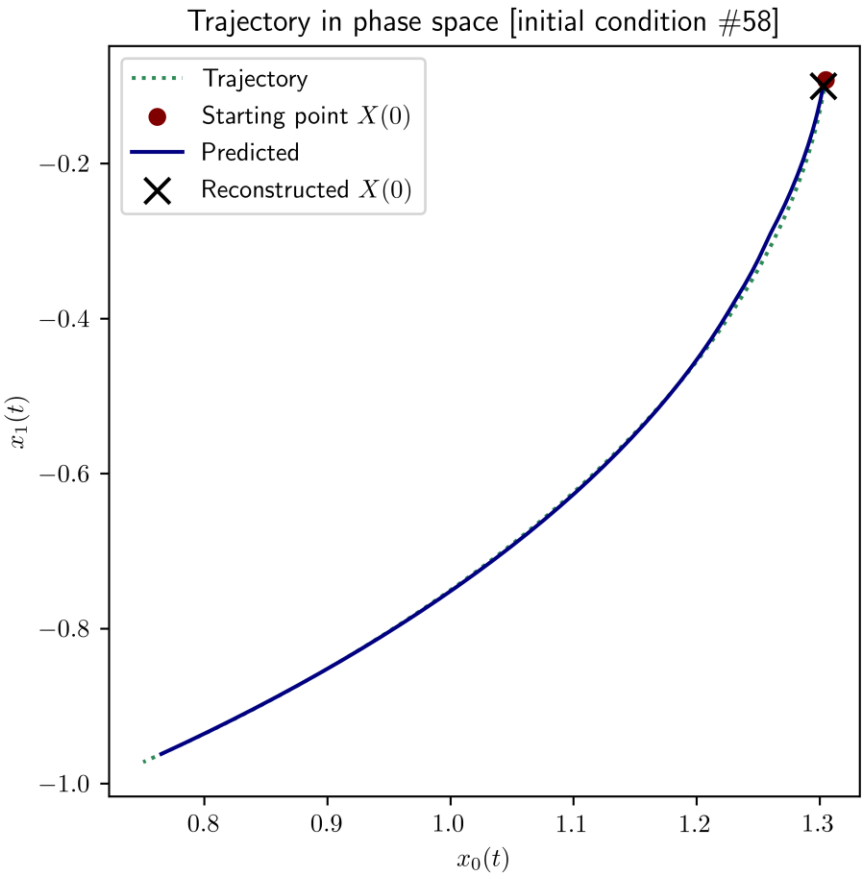


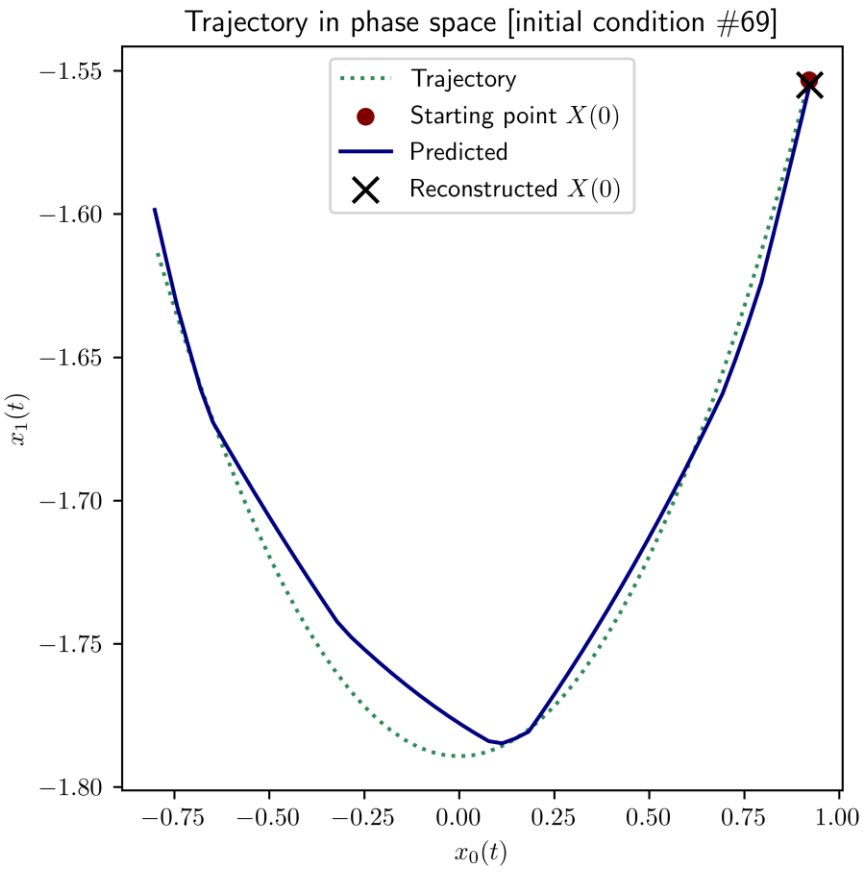



# Trajectories

BEST and WORST

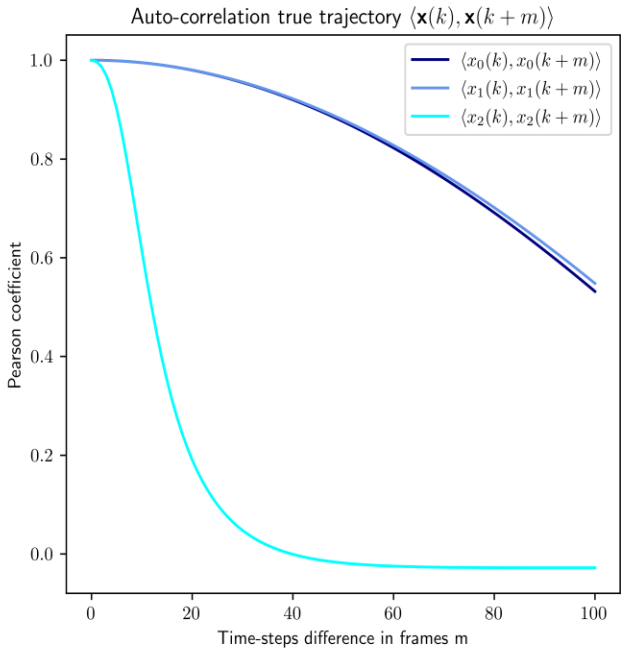


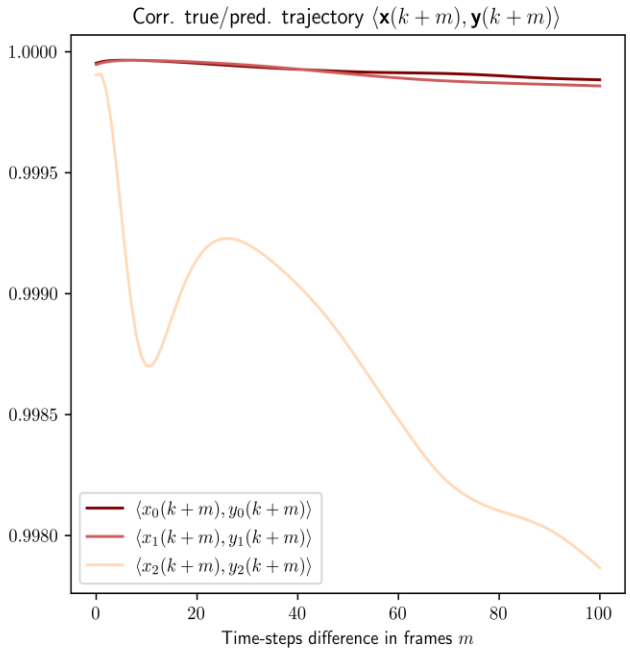



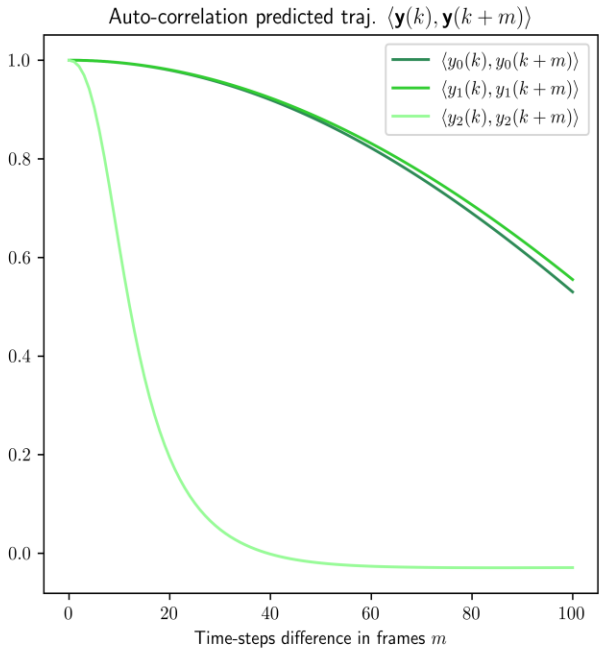
# Results

## Fluid box

### Correlations and auto-correlations

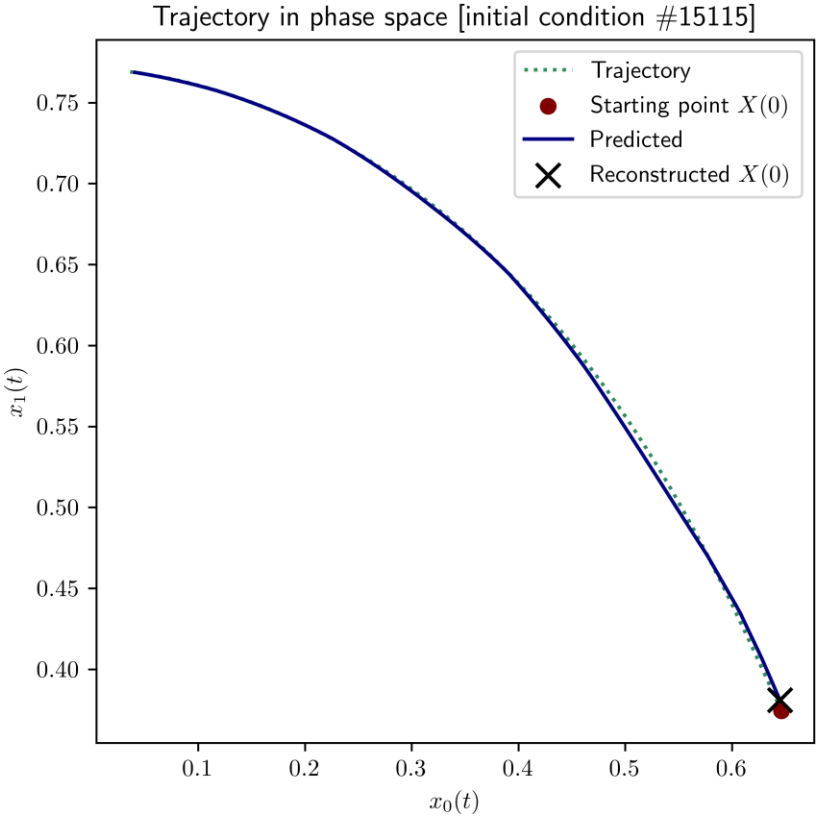


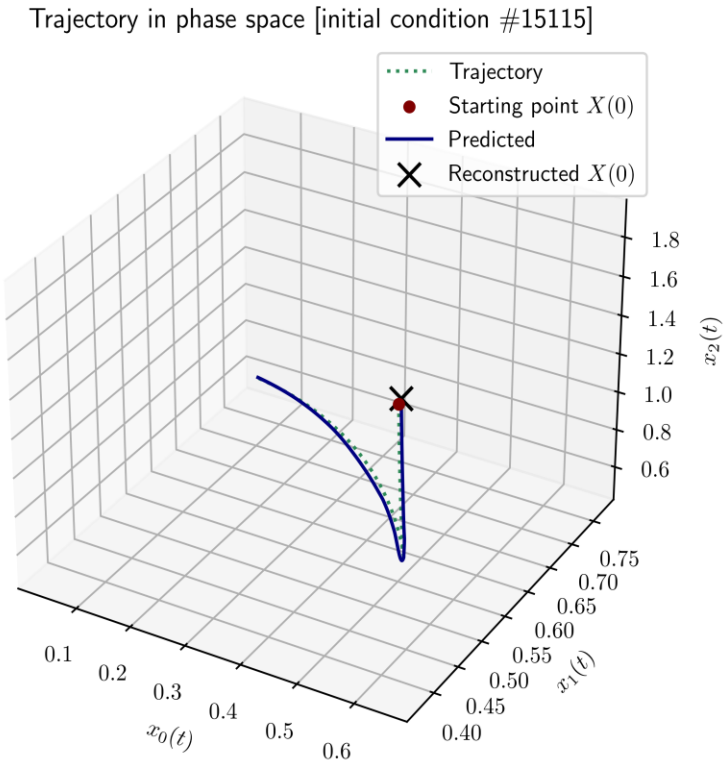



# Trajectories

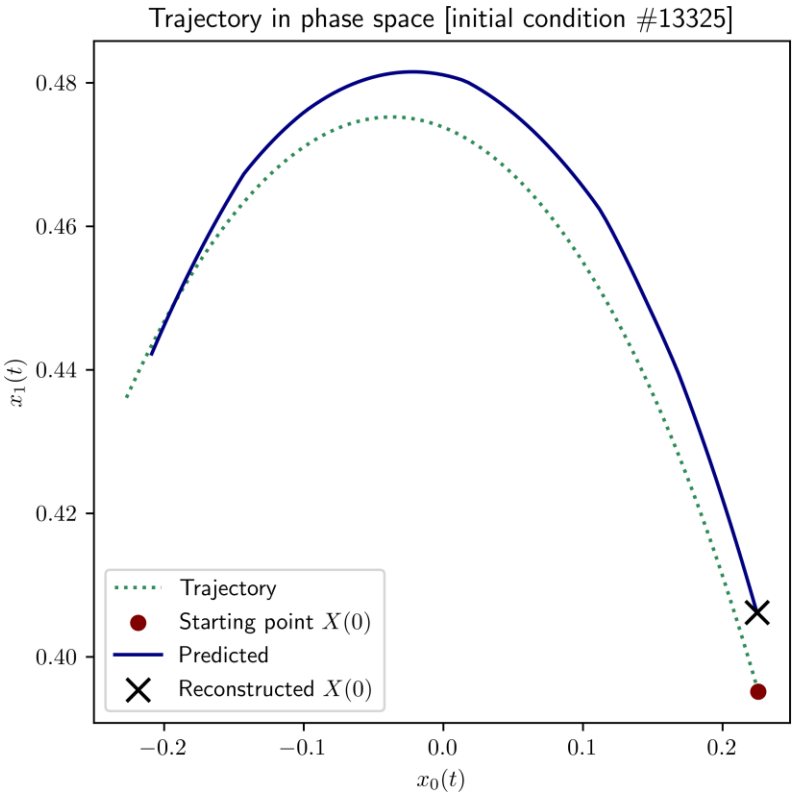
BEST

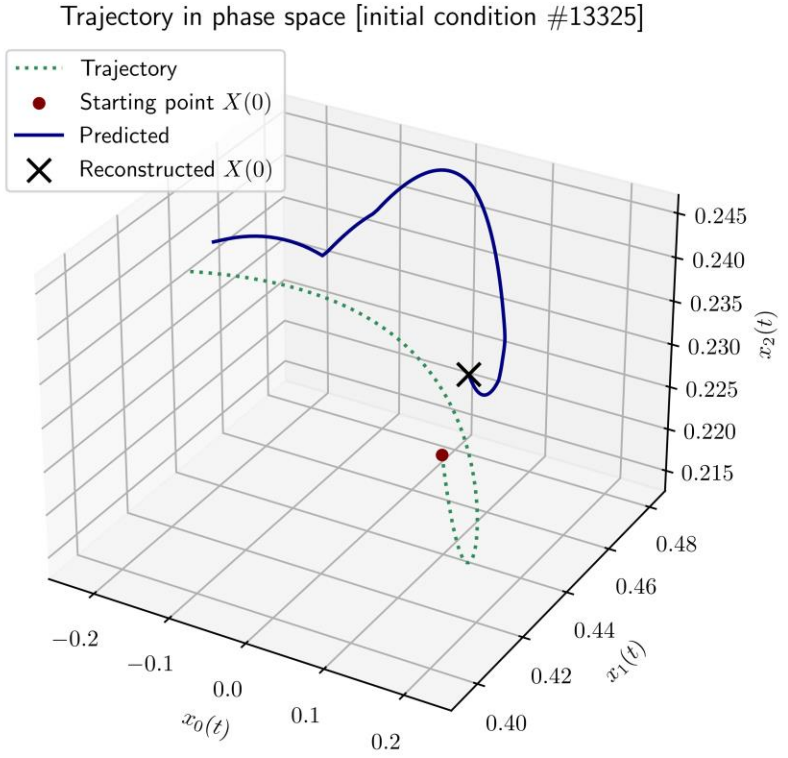


# Trajectories

WORST

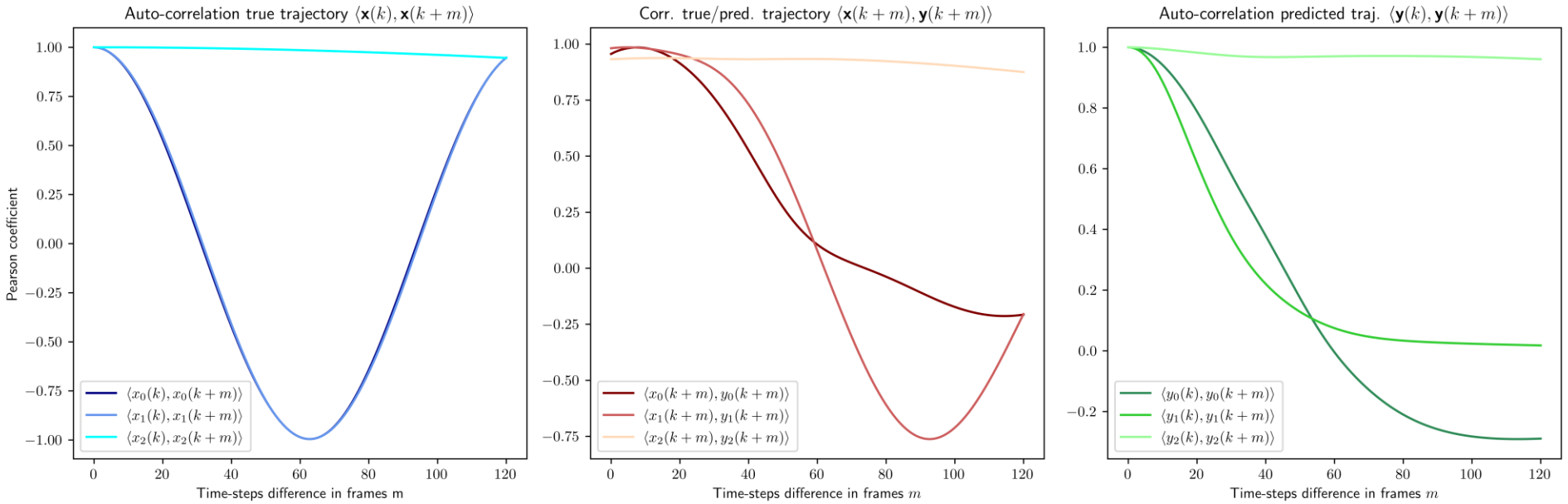


# Results

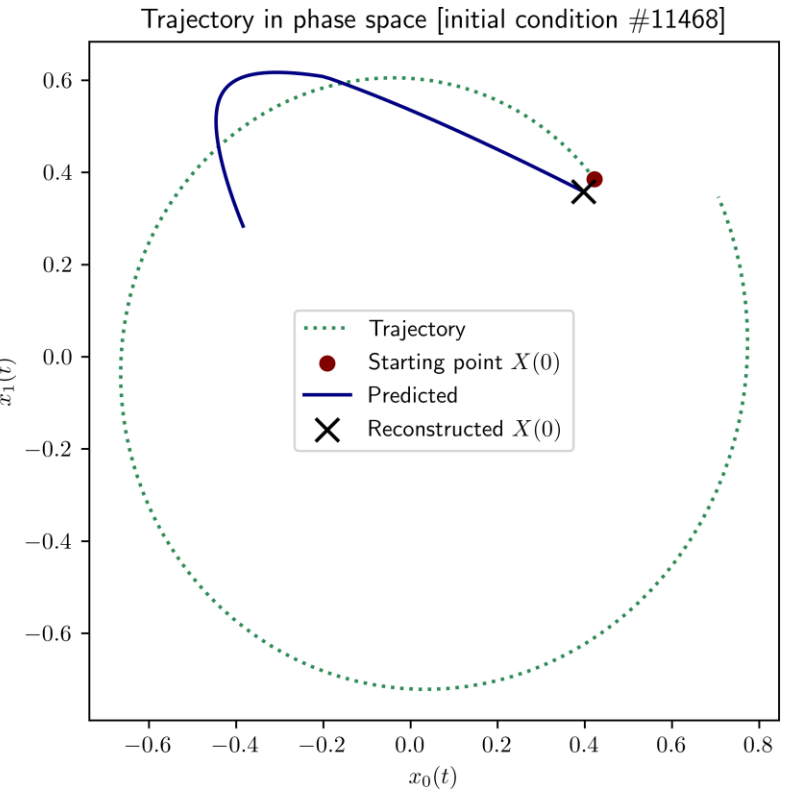
## Fluid on attractor

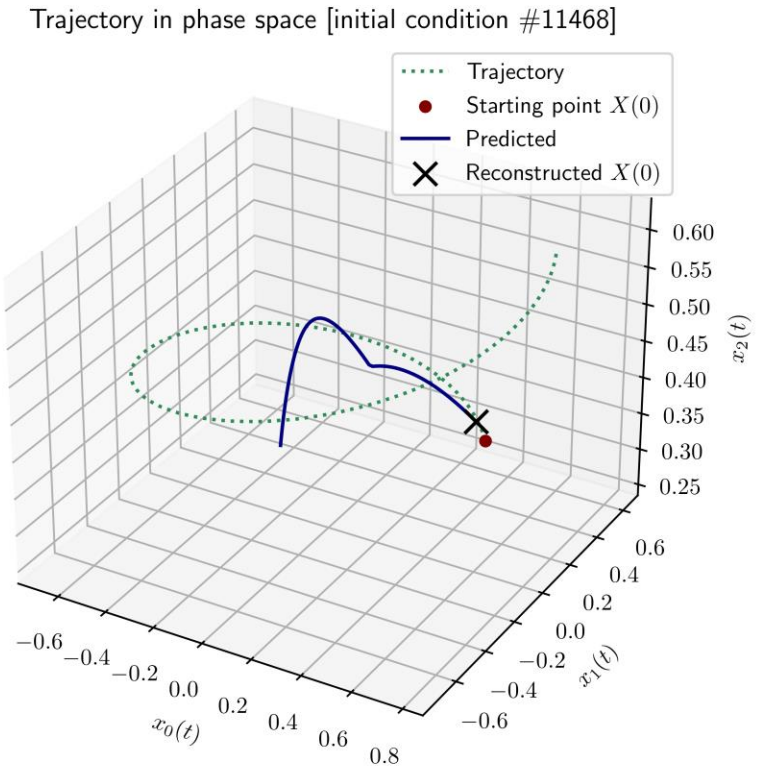
Correlations and auto-correlations



# Trajectories

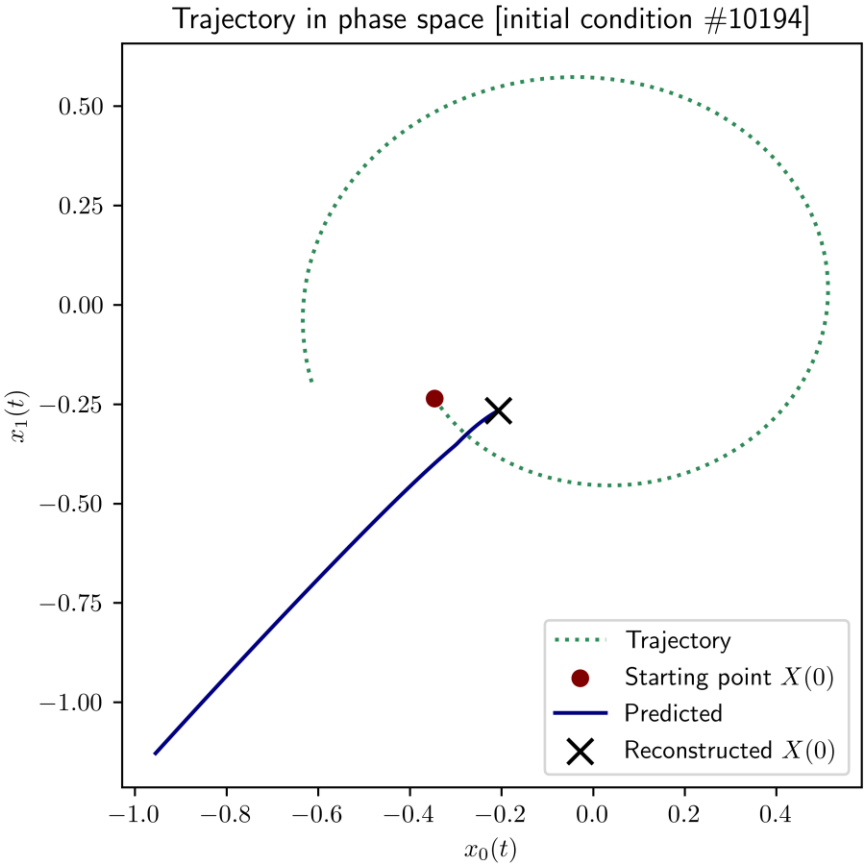
BEST

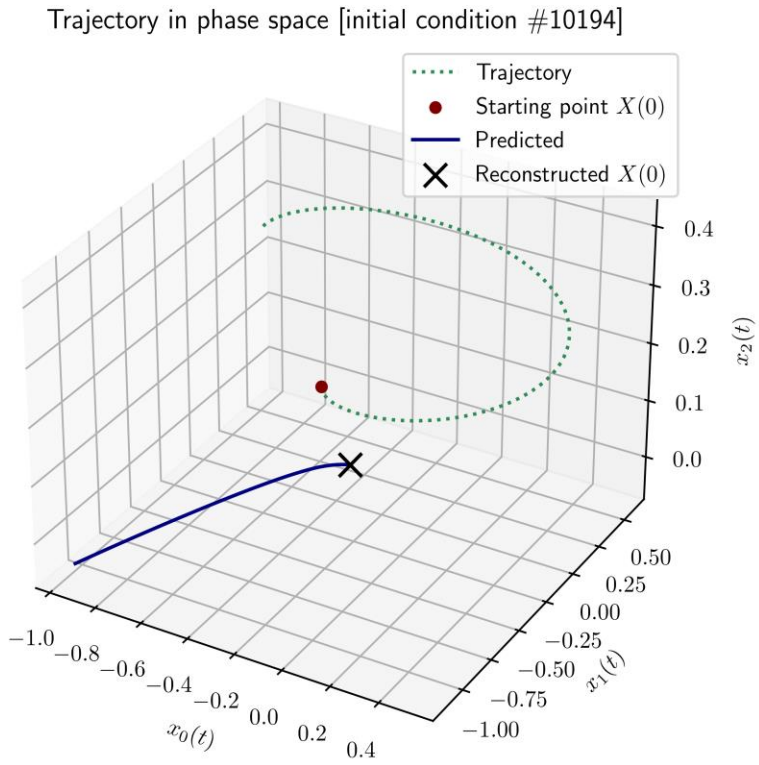


# Trajectories

WORST



# Digression

- There are **many ongoing challenges and promising directions** that motivate future work (new symmetries and conservation laws, turbulence, epidemiology, neuroscience etc.), all facilitated by more powerful network representations
- There are still several **limitations** associated with deep learning, including the need for vast and diverse data and long computation to train models.
- Even more concerning is the dubious **generalizability** and **interpretability** of the resulting models, as deep learning architecture.
- There are also more specific limitations to the current proposed architecture, foremost, choosing the dimension of the autoencoder coordinates, **y**.