# Koopman Operator Estimation with Neural Network

Your Name

November 22, 2023

## 1 Introduction

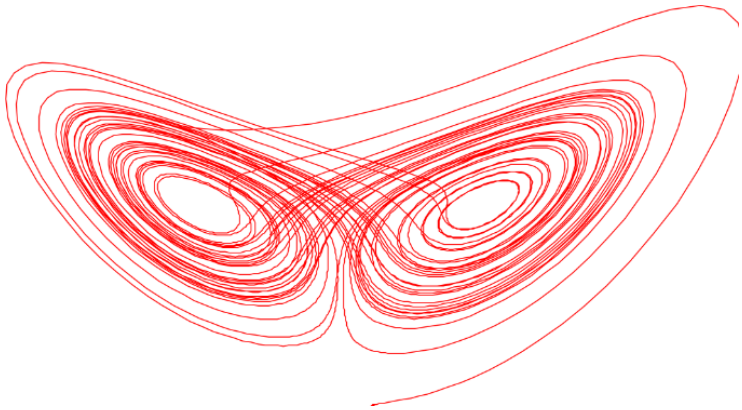

Figure 1: Example for a Non-linear system: The Lorenz system is a set of three coupled ordinary differential equations that describe a chaotic, deterministic system

In everyday scenarios, we often encounter complex relationships that don't follow a straight, predictable path. Consider the movement of a mass attached to a spring or the rhythmic oscillations in a pendulum or heartbeat—these are examples of non-linear dynamics, where relationships among variables are intricate.

To tackle the challenges of understanding and modeling such non-linear systems, recent developments introduce a powerful tool called the Koopman operator. This paper dives into exploring the Koopman operator, a mathematical concept designed to simplify the modeling of non-linear systems. By utilizing the Koopman operator, researchers gain a more accessible framework to comprehend and forecast the behavior of intricate systems. This exploration outlines the practical application of the Koopman operator, highlighting its effectiveness

in overcoming the complexities associated with modeling non-linear functions. Ultimately, this contributes to advancing our scientific understanding in this field, offering new insights into the dynamics of real-world systems.

# 2 Methodology

**Solving the Partial differential equations:**
Solving partial differential equations (PDEs) involves employing numerical methods to approximate solutions, with the Euler and Runge-Kutta methods being notable approaches. The Euler method is a straightforward technique that discretizes the PDE and updates the solution incrementally, based on the local slope at each step. While conceptually simple, the Euler method may suffer from stability issues and reduced accuracy in certain scenarios. On the other hand, the Runge-Kutta method, particularly the fourth-order variant, is a widely used numerical technique. It refines accuracy by evaluating multiple intermediate stages, providing a more robust approximation of the solution.

In practical implementations of these methods, Python libraries such as odeint are often employed. Utilizing odeint streamlines the coding process, offering a convenient and efficient tool for solving PDEs numerically. This integration of numerical methods with Python libraries empowers researchers and engineers to address complex PDEs, facilitating a deeper understanding of dynamic systems and phenomena.
**Koopman operator:**
The Koopman operator is a linear operator that describes the evolution of scalar observables (i.e., measurement functions of the states) in an infinite dimensional Hilbert space. This operator's theoretical point of view lifts the dynamics of a finite-dimensional nonlinear system to an infinite-dimensional function space where the evolution of the original system becomes linear.

In the context of Koopman operators, we will focus on dynamical systems that can be modelled as:

$$\dot{x} = f(x)$$

, where x is an element of the state space $S \in R^n$ (the current state of the system) and

$$x^{t+1} = T(x^t)$$

in which $T(.) : S \to S$ can be referred to as the dynamic mapping which performs time-integration on the state variables. Given that we will be interested in the prediction of time steps of the dynamical system, we will be generally interested in this discrete-time formulation. In 1931, Bernard Koopman proposed the existence of a compositional operator for all dynamical systems that can be expressed as the equations above [1]. Now known as the Koopman operator (or the left-adjoint of the Frobenius-Perron operator), this allows for a discrete dynamical system to be decomposed as follows:

$$Ug(x) = g \circ T(x)$$

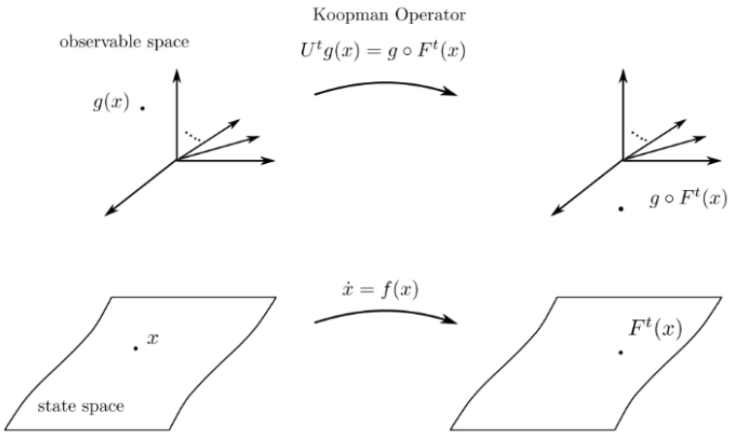


Figure 2: By transforming the state variables to the observable space of $g(x)$, the dynamics is linear but infinite-dimensional

in which $U$ denotes the Koopman operator which is nothing more than a linear transformation. The continuous time version of the Koopman operator contains a single time parameter, but essentially follows the same form:

$$U(t)g(x) = g \circ F(t, x)$$

in which $F$ is the flow map between the initial state to time $t$ in continuous space. Now you may be thinking "Wow, any dynamical system can be evolved using a linear transformation! That sounds fantastic!" Indeed it does, however the catch is that the Koopman operator updates some unknown vector of real valued observables of this dynamical system denoted by $g : S \to R$. Typically, these values are referred to as the Koopman observables. To make matters even more difficult, theoretically, the vector space of these observables is infinite which practically speaking will need to be approximated. We can view the use of Koopman operators as a trade off from simple observables and complex dynamics to complex observables and simple dynamics.

**Machine Learning Koopman Embeddings:**
In understanding the Koopman operator and its valuable physical insights, the goal is to formulate a method for efficiently learning Koopman observables $g(x)$ and the Koopman operator itself. In the era of big data, machine learning methods have become effective tools for this task, sparking a renewed interest in Koopman operators. Traditional methods like dynamic mode decomposition (DMD) or its variants (e.g., EDMD, SINDy) have been common, but they require a predefined library of potential observables, posing a drawback.

Enter deep learning, where neural networks offer a solution. Unlike traditional methods, a neural network learns a function from the system's state variables to Koopman observables and vice versa. This eliminates the need for a predefined library. Using a neural network simplifies the process, allowing

for highly complex Koopman observables without the need for expert intuition. Essentially, the deep neural network, through gradient descent, takes on the task of figuring out the observables, making the process more accessible and automated.

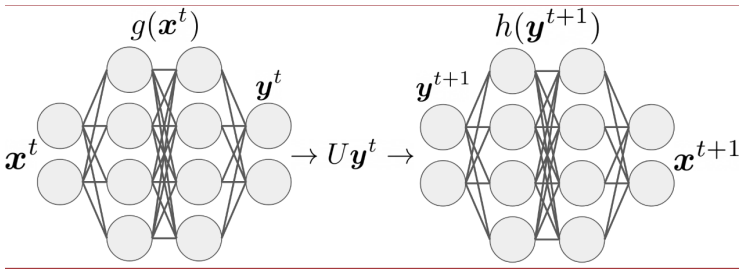


Figure 3: The common neural networks encoder/decoder structure for learning Koopman observables, $y^t = g(x^t)$, as a function of the systems state variables.

## 2.1 Non-Linear Function Taken

The non-linear system I used is known as the Van del Pol oscillator. The Van der Pol oscillator, a second-order nonlinear dynamical system, is renowned for its distinctive behavior, demonstrating self-sustained oscillations with a tendency towards limit cycle behavior and finding applications in various fields such as electronics and biology. The equation is given by:

$$\frac{d^2x}{dt^2} - \mu(1 - x^2)\frac{dx}{dt} + x = 0 \tag{1}$$

where:

- $x$ represents the displacement from the equilibrium position,
- $t$ is time,
- $\mu$ is a scalar parameter indicating the nonlinearity and the strength of the damping.

## 2.2 Neural Network Architecture

My neural network architecture consists of eight hidden layers. The input layer expects data with a shape of (2,), indicating two input features per sample. The subsequent four dense layers employ Rectified Linear Unit (ReLU) activation functions with varying numbers of neurons (128, 64, 32, and 4, respectively). Following these, a layer with four neurons and a ReLU activation is present, followed by another layer with four neurons but a linear activation function, commonly used for regression tasks. The network then expands through three

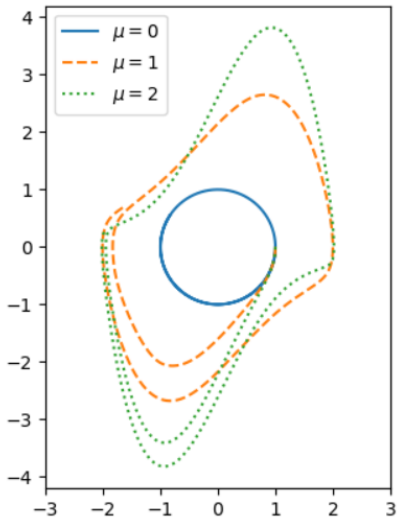


Figure 4: Evolution of the limit cycle in the phase plane. The limit cycle begins as a circle and, with varying $\mu$, becomes increasingly sharp. An example of a relaxation oscillator.

more dense layers, mirroring the structure of the initial layers in reverse order, both utilizing ReLU activation. The final output layer consists of two neurons, and although no explicit activation function is specified, it defaults to linear activation. This architecture suggests a model designed to predict a two-dimensional output, with the Koopman operator predicting the state at the next time step.

Listing 1: Neural Network Architecture

```
# Define the Koopman operator neural network
model = tf.keras.Sequential([
    layers.Input(shape=(2,)),
    layers.Dense(128, activation='relu'),
    layers.Dense(64, activation='relu'),
    layers.Dense(32, activation='relu'),
    layers.Dense(4, activation = 'relu'),
    layers.Dense(4, activation = 'linear'),
    layers.Dense(32, activation='relu'),
    layers.Dense(64, activation='relu'),
    layers.Dense(128, activation='relu'),
    layers.Dense(2),
])
```

### 2.3 Loss Function

The loss function is a mathematical function that calculates the difference between the predicted values and the true values. In this case, the 'mean-squared error' (MSE) is used. MSE is a common choice for regression problems, where the goal is to predict continuous values. It calculates the average of the squared differences between predicted and true values. Minimizing the MSE during training aims to make the model's predictions as close as possible to the actual values.

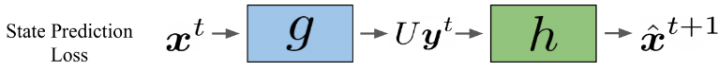


Figure 5: Loss Function.

$$\text{Loss} = MSE(x^{t+1}, \hat{x}^{t+1}) \tag{2}$$

- $x^t$ state of the system at t,
- $\hat{x}^{t+1}$ predicted sate of the system at time t+1,
- $U$ is the koopman operator matrix.

## 3 Results

Below are the results obtained after training the model.

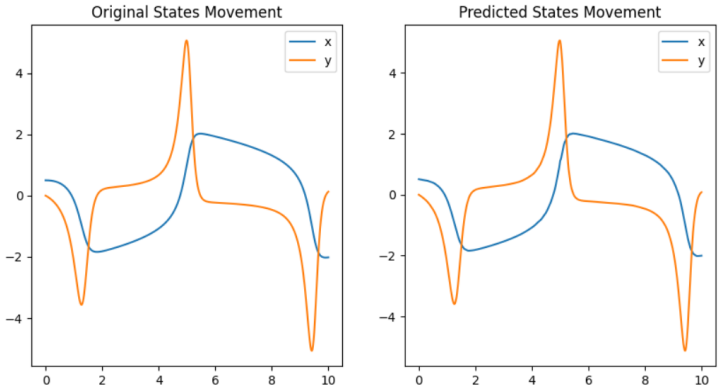


Figure 6: Graphs showing original data vs predicted data of the states.

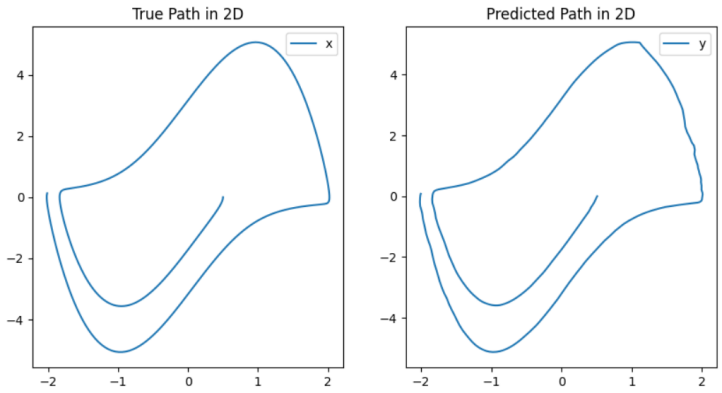


Figure 7: Two dimension path of the given oscillator and predicted path

```
Linear Layer Weights (Koopman Operators):
 [[ 0.78117687 -0.4005713   0.32135096  0.7450857 ]
 [ 0.37719056  0.6350642   0.43564186  0.47469696]
 [-0.41941515  0.61414814  0.539199    0.6412569 ]
 [-0.37140524  0.14613983  0.78170574 -0.7095174 ]]
```

Figure 8: Koopman Operator Weights.

# 4 Conclusion

In conclusion, this paper successfully implements the Koopman operator coupled with neural networks to simulate the dynamics of a nonlinear system, exemplified by the Van der Pol oscillator. The integration of these mathematical and computational tools proves effective in capturing the intricate behavior of the chosen system. This research contributes to the synergy of Koopman operator theory and modern machine learning, offering potential applications in predictive modeling, control systems, and understanding complex dynamical phenomena. While demonstrating success with the Van der Pol oscillator, future work may explore broader applications and optimize the methodology for enhanced performance and interpretability. This study represents a noteworthy step in leveraging advanced mathematical techniques and artificial intelligence for studying and predicting nonlinear dynamics.