Mathematical Foundations of the Dynamical Koopman Neural Operator (KNO) Framework

Ameir Shaa, Claude Guet

April 1, 2025

Contents

1 Introduction to Koopman Theory

The DynamicalOperator framework is built upon Koopman operator theory, which provides a way to transform nonlinear dynamical systems into linear operators in a higher-dimensional function space.

1.1 Dynamical Systems Framework

Consider a discrete-time dynamical system:

$$\mathbf{x}_{t+1} = \mathbf{F}(\mathbf{x}_t) \tag{1}$$

where $\mathbf{x}_t \in \mathcal{M} \subset \mathbb{R}^n$ is the state of the system at time t, and $\mathbf{F} : \mathcal{M} \to \mathcal{M}$ is a (potentially nonlinear) function describing the evolution of the system.

1.2 The Koopman Operator

The Koopman operator $\mathcal{K}$ is an infinite-dimensional linear operator that acts on observable functions $g : \mathcal{M} \to \mathbb{R}$ as:

$$\mathcal{K}g = g \circ \mathbf{F} \tag{2}$$

More explicitly, for any observable function g, the Koopman operator gives:

$$(\mathcal{K}g)(\mathbf{x}_t) = g(\mathbf{F}(\mathbf{x}_t)) = g(\mathbf{x}_{t+1}) \tag{3}$$

The key insight is that while $\mathbf{F}$ may be nonlinear, the Koopman operator $\mathcal{K}$ is always linear, albeit infinite-dimensional.

2 Neural Operator Formulation

2.1 Neural Networks as Function Approximators

In practice, we approximate the Koopman operator using neural networks. Specifically, we use encoder-decoder architectures:

$$\begin{aligned} \mathbf{\Phi} &: \mathcal{M} \to \mathcal{H} \quad \text{(Encoder)} \\ \mathbf{\Psi} &: \mathcal{H} \to \mathcal{M} \quad \text{(Decoder)} \end{aligned} \tag{4}$$

where $\mathcal{H}$ is a higher-dimensional latent space in which we approximate the Koopman dynamics.

2.2 Spectral Representation in Fourier Space

The DynamicalOperator leverages a spectral representation in Fourier space. For a function $u(\mathbf{x})$, its Fourier transform is:

$$\hat{u}(\mathbf{k}) = \mathcal{F}[u](\mathbf{k}) = \int_{\mathbb{R}^d} u(\mathbf{x}) e^{-i\mathbf{k}\cdot\mathbf{x}}\, d\mathbf{x} \tag{5}$$

In discrete settings, we use the Discrete Fourier Transform (DFT):

$$\hat{u}_{\mathbf{k}} = \sum_{\mathbf{x}} u_{\mathbf{x}} e^{-i\mathbf{k}\cdot\mathbf{x}} \tag{6}$$

3 The Dynamical Neural Operator Architecture

3.1 1D Spectral Operator

For one-dimensional systems, the spectral representation of the Koopman operator in Fourier space is:

$$\hat{K}(k) = \mathcal{F}[\mathcal{K}](k) \tag{7}$$

In our parameterization, we approximate this with a learnable complex-valued parameter matrix $\mathbf{W} \in \mathbb{C}^{d\times d\times m}$ where d is the latent dimension and m is the number of Fourier modes.

For an input $\mathbf{u} \in \mathbb{R}^{b\times d\times n}$ (batch size b, latent dimension d, spatial dimension n), the transformation in Fourier space is:

$$\hat{\mathbf{v}}_{b,f,k} = \sum_{t} \hat{\mathbf{u}}_{b,t,k} \cdot \mathbf{W}_{t,f,k} \tag{8}$$

using Einstein summation convention where f indicates the output feature dimension.

3.2 2D Spectral Operator

For two-dimensional systems, we extend to a 2D Fourier transform. The spectral operator is parameterized by a tensor $\mathbf{W} \in \mathbb{C}^{d\times d\times m_x\times m_y}$ where m_x and m_y are the numbers of Fourier modes in the x and y directions.

The spectral transformation in 2D is:

$$\hat{\mathbf{v}}_{b,f,k_x,k_y} = \sum_{t} \hat{\mathbf{u}}_{b,t,k_x,k_y} \cdot \mathbf{W}_{t,f,k_x,k_y} \tag{9}$$

3.3 Low-Rank Truncation

To make computation tractable, we truncate the Fourier representation to the lowest m modes:

$$\hat{\mathbf{v}}_{b,f,k} = \begin{cases} \sum_t \hat{\mathbf{u}}_{b,t,k} \cdot \mathbf{W}_{t,f,k}, & \text{if } |k| \leq m \\ 0, & \text{otherwise} \end{cases} \tag{10}$$

This approach is motivated by the fact that in many physical systems, the dynamics are dominated by the low-frequency modes.

4 Iterative Application and Nonlinear Activation

In our implementation, we apply the spectral operator iteratively:

$$\mathbf{z}^{(0)} = \mathbf{\Phi}(\mathbf{x}) \tag{11}$$

For $i = 1, 2, \ldots, r$ iterations:

$$\tilde{\mathbf{z}}^{(i)} = \mathcal{F}^{-1}\left[\hat{K} \cdot \mathcal{F}[\mathbf{z}^{(i-1)}]\right] \tag{12}$$

If we use nonlinear activation (when use_linear=False):

$$\mathbf{z}^{(i)} = \tanh(\mathbf{z}^{(i-1)} + \tilde{\mathbf{z}}^{(i)}) \tag{13}$$

Otherwise (when use_linear=True):

$$\mathbf{z}^{(i)} = \mathbf{z}^{(i-1)} + \tilde{\mathbf{z}}^{(i)} \tag{14}$$

5 Residual Connections and Feature Mixing

We also include a residual connection with a learnable weight matrix:

$$\mathbf{z}^{(r+1)} = \tanh(\mathbf{W}_0\mathbf{z}^{(0)} + \mathbf{z}^{(r)}) \tag{15}$$

where $\mathbf{W}_0$ is implemented as a 1×1 convolution.

6 Overall Forward Process

The complete forward process can be summarized as:

$$\mathbf{z}^{(0)} = \tanh(\mathbf{\Phi}(\mathbf{x})) \tag{16}$$

$$\tilde{\mathbf{z}}^{(i)} = \mathcal{F}^{-1}\left[\hat{K} \cdot \mathcal{F}[\mathbf{z}^{(i-1)}]\right] \tag{17}$$

$$\mathbf{z}^{(i)} = \begin{cases} \tanh(\mathbf{z}^{(i-1)} + \tilde{\mathbf{z}}^{(i)}), & \text{if use\_linear} = \text{False} \\ \mathbf{z}^{(i-1)} + \tilde{\mathbf{z}}^{(i)}, & \text{if use\_linear} = \text{True} \end{cases} \tag{18}$$

$$\mathbf{z}^{(r+1)} = \begin{cases} \tanh(\text{BN}(\mathbf{W}_0\mathbf{z}^{(0)}) + \mathbf{z}^{(r)}), & \text{if use\_batch\_norm} = \text{True} \\ \tanh(\mathbf{W}_0\mathbf{z}^{(0)} + \mathbf{z}^{(r)}), & \text{if use\_batch\_norm} = \text{False} \end{cases} \tag{19}$$

$$\mathbf{y} = \mathbf{\Psi}(\mathbf{z}^{(r+1)}) \tag{20}$$

where BN denotes batch normalization.

7 Loss Function and Training

7.1 Dual Objective Function

During training, we optimize a dual objective:

$$\mathcal{L} = \alpha \mathcal{L}_{\text{pred}} + \beta \mathcal{L}_{\text{recon}} \tag{21}$$

where:

$$\mathcal{L}_{\text{pred}} = \frac{1}{B} \sum_{b=1}^{B} \|\mathbf{y}_b - \mathbf{x}_b^{\text{future}}\|_2^2 \tag{22}$$

$$\mathcal{L}_{\text{recon}} = \frac{1}{B} \sum_{b=1}^{B} \|\mathbf{\Psi}(\mathbf{\Phi}(\mathbf{x}_b)) - \mathbf{x}_b\|_2^2 \tag{23}$$

with α and β being weighting coefficients (typically $\alpha = 5$ and $\beta = 0.5$).

7.2 Autoregressive Prediction

During inference, we use autoregressive prediction:

$$\mathbf{x}_{t+1} = \mathcal{G}(\mathbf{x}_{t-k+1:t}) \tag{24}$$

$$\mathbf{x}_{t+2} = \mathcal{G}(\mathbf{x}_{t-k+2:t+1}) \tag{25}$$

$$\vdots \tag{26}$$

$$\mathbf{x}_{t+\tau} = \mathcal{G}(\mathbf{x}_{t+\tau-k:t+\tau-1}) \tag{27}$$

where $\mathcal{G}$ represents the full DynamicalOperator model and k is the input time horizon.

8 Takens' Delay Embedding

For systems with partial observations, we can use Takens' embedding theorem. Given a time series $\{s_t\}$, the delay embedding is:

$$\mathbf{x}_t = (s_t, s_{t-\tau}, s_{t-2\tau}, \ldots, s_{t-(d-1)\tau}) \tag{28}$$

where τ is the embedding delay and d is the embedding dimension.

This allows us to reconstruct the phase space of the underlying dynamical system, even when we only observe a single variable from that system.

9 Application to the Lorenz System

The Lorenz system is defined by:

$$\frac{dx}{dt} = \sigma(y - x) \tag{29}$$

$$\frac{dy}{dt} = x(\rho - z) - y \tag{30}$$

$$\frac{dz}{dt} = xy - \beta z \tag{31}$$

with typical parameters $\sigma = 10$, $\rho = 28$, and $\beta = 8/3$.

In our framework, we treat the Lorenz variables as a 3-channel 1D signal with a single spatial dimension. The DynamicalOperator learns to predict future states directly from past states without requiring explicit knowledge of the governing equations.