Fourier Neural Operators as nonlinear time-varying system models

Tufts UNIVERSITY
Institute for Artificial Intelligence

Rad Haghi[1,2], Bipin Jairaj Gaikwad[1], Babak Moaveni[2] and Abani Patra [1]

[1]Tufts Institute for Artificial Intelligence, Tufts University, Medford, MA 02155
[2]Department of Civil and Environmental Engineering, Tufts University, Medford, MA 02155

Introduction

Fourier Neural Operators (FNOs) have emerged as effective surrogate models for parametric PDEs and nonlinear dynamical systems [4, 6]. Recent theory [5] shows FNO approximation error bounds as a function of grid discretization and solution regularity, also building confidence in the method. Although FNOs have successfully addressed steady-state and time-dependent systems [4, 8, 3], few studies have explored complex nonlinearities and explicitly time-varying parameters. This research fills this gap by applying FNOs to nonlinear systems with time-dependent parameters, demonstrated on a 2-DOF Duffing oscillator with varying stiffness. A spectrogram based loss function [2, 1] significantly enhances prediction accuracy. Results show that integrating this frequency-domain loss notably improves the FNO's performance.

Fourier Neural Operator

The "out-of-the-box" FNO architecture [4] is presented in the figure below.

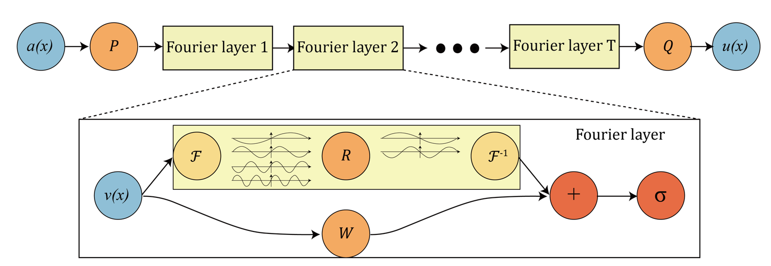


Figure: Schematic FNO architecture [6]

In common practice, the mean squared error (MSE) or L_2 loss is utilized for training.

What Has Worked? Simple and Complicated Linear Systems

We tested a four-layer FNO on a 2-DOF undamped mass-spring system.

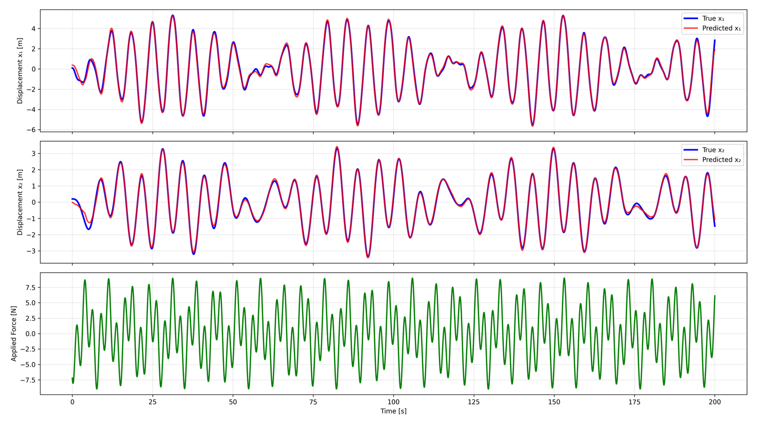
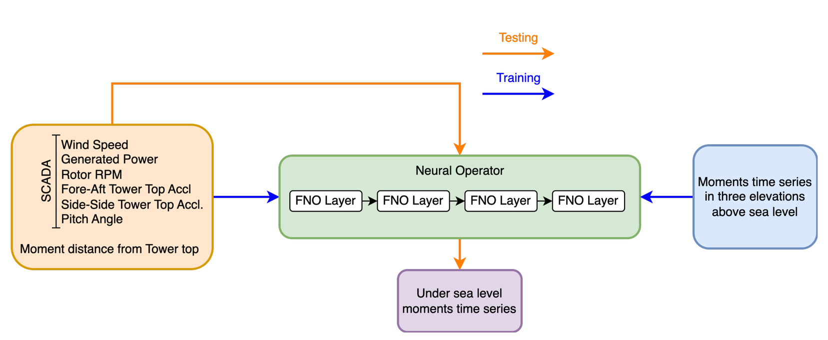
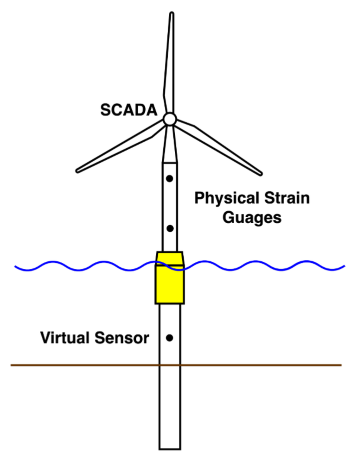


Figure: FNO test on linear 2-DOF mass-spring system

We also utilized a similar architecture to build a virtual sensor for an offshore wind turbine model. For testing the FNO on an offshore wind turbine (OWT), we utilized publicly available OWT model simulation results [7]. We propose the use of the FNO model as a virtual sensor for the moment time series measurements below the water line, which are not accessible.



(a) FNO as a virtual sensor for an Offshore Wind Turbine



(b) Offshore Wind Turbine virtual sensor

We show that the FNO can accurately predict the moment time series for the location along the foundation that it is not trained on, indicating successful operator learning and generalization in function space.

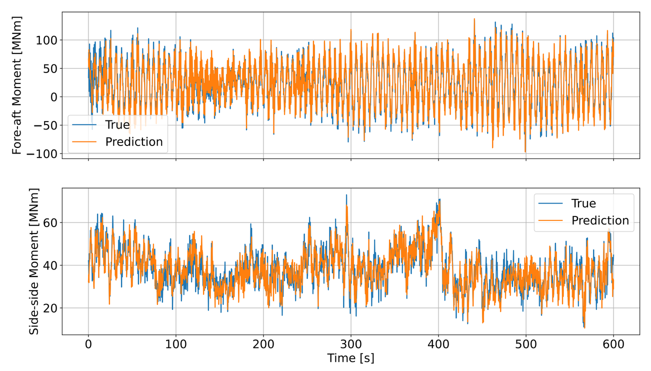
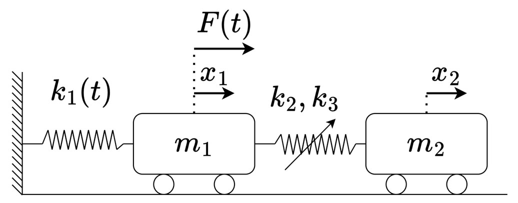


Figure: FNO as a virtual sensor for an OWT simulation results

Test Problem



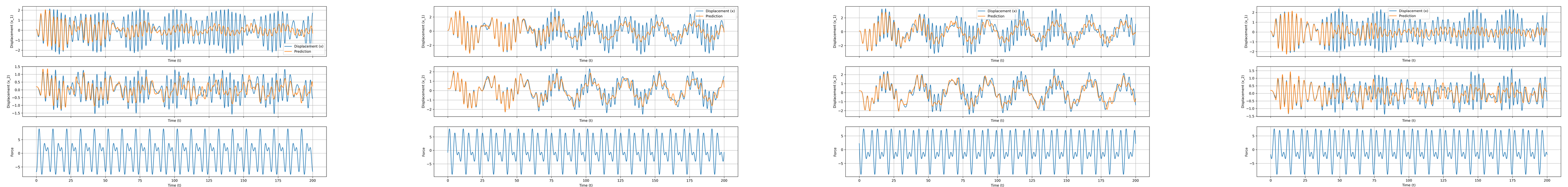
$$
\begin{aligned}
&m_1\ddot{x}_1 + k_1(t)x_1 + k_2(x_1 - x_2) + k_3(x_1 - x_2)^3 = F(t)\\
&m_2\ddot{x}_2 + k_2(x_2 - x_1) + k_3(x_2 - x_1)^3 = 0\\
&k_1(t) = k_0 + k_{initial}e^{-\alpha t}\\
&F(t) = A\sin(2\pi\omega_1 t + \phi_1) + B\cos(2\pi\omega_2 t + \phi_2)
\end{aligned}
$$

$$
\begin{aligned}
&x_1(0) = 0.1,\ \dot{x}_1(0) = 0\\
&x_2(0) = 0.2,\ \dot{x}_2(0) = 0
\end{aligned}
$$

Where m_1 and m_2 are 5 and 10 kg, k_0 is 1 N/m, $k_{initial}$ is 3 N/m while α is 0.01. k_2 and k_3 are 3 N/m and 2 N/m^3. In the forcing function ω_1 and ω_2 are 0.2 and 0.5 Hz. A and B are sampled from $\mathcal{U}(0,4)$ and $\mathcal{U}(0,5)$ utilizing Sobol's sampling method [9], while ϕ_1 and ϕ_2 are sampled from $\mathcal{U}(0,2\pi)$. We build the dataset by taking 2^{11}, 2^{12}, 2^{13}, and 2^{14} samples from these random variables and then generating the force time series. Afterwards, we solve the 2-DOF system based on those force time series for the displacement of m_1 and m_2. The equation of motion was solved utilizing `scipy` Runge-Kutta-Fehlberg Method (RKF45).

What Has Not Worked? Nonlinear, Non-stationary Systems

We tested the FNO with a nonlinear cubic stiffening 2-DOF mass-spring system and time-dependent loading/stiffness parameters. The goal was to test the ability of FNO to predict the displacement time series of two masses when it receives the force time series as input. The FNO trained with MSE loss did not function properly in our test. One solution is increasing the size of the datasets. We tested this on 2^{11},2^{12},2^{13} and 2^{14} dataset sizes. The results show no significant improvement in the results.



(a) 2^{11} dataset testing result (b) 2^{12} dataset testing result (c) 2^{13} dataset testing result (d) 2^{14} dataset testing result

FNO is able to capture the signal during the initial steps. However, as the system evolves, the prediction worsens. We solve this by designing a novel loss function that can capture the dynamic behavior of the system.

Spectrogram Loss

The Spectrogram Loss compares the frequency content in time of the prediction and ground truth, utilizing short-time Fourier transform (STFT). The loss is calculated for the magnitude of the STFT output. Mathematically, it can be defined as:

$$
\mathcal{L}_{\text{Spectrogram}} = \frac{\|\boldsymbol{M}_{\text{pred}} - \boldsymbol{M}_{\text{true}}\|_F}{\|\boldsymbol{M}_{\text{true}}\|_F + \epsilon}
$$

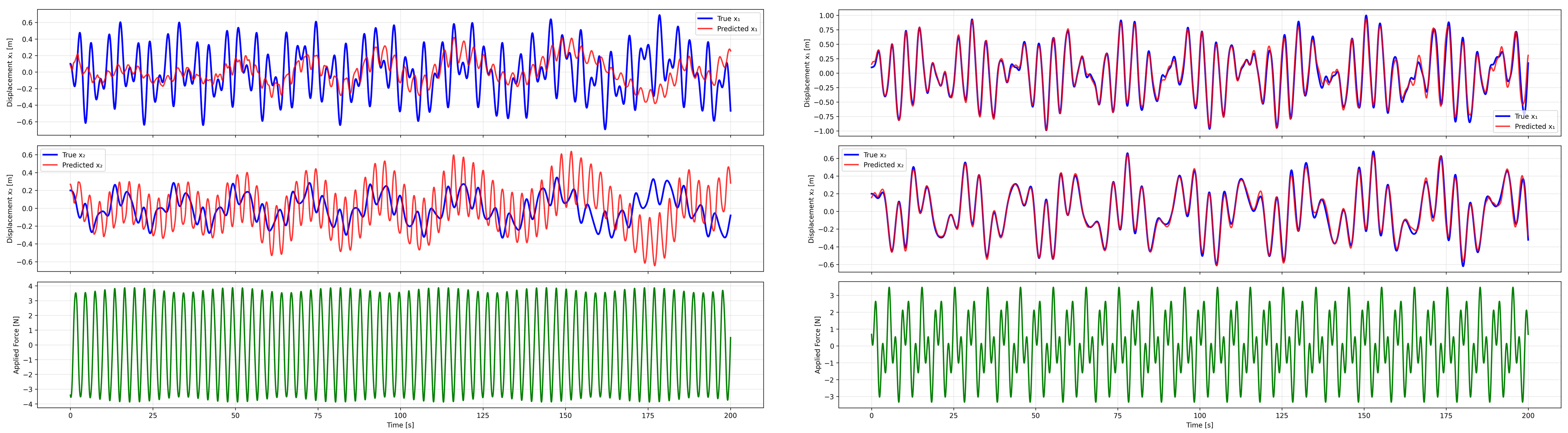
where $\|\cdot\|_F$ is the Frobenius norm.

Results

We train the 4-layer FNO configurations with two loss configurations on the dataset:

- **FNO1**: four layers FNO, 512 modes, batch size 64, $\mathcal{L} = \mathcal{L}_{\text{MSE}}$
- **FNO2**: four layers FNO, 512 modes, batch size 64, $\mathcal{L} = \lambda_{spectrogram}\mathcal{L}_{\text{Spectrogram}} + \lambda_{data}\mathcal{L}_{\text{MSE}}$

where $\lambda_{spectrogram}$ is 0.2, and λ_{data} is 0.8. The dataset is divided into 80% for training and 20% for testing. The results presented below are testing results after 100 epochs.



(a) FNO1: $\mathcal{L}_{\text{MSE}}$ (b) FNO2: $\mathcal{L} = \lambda_{spectrogram}\mathcal{L}_{\text{Spectrogram}} + \lambda_{data}\mathcal{L}_{\text{MSE}}$

The FNO with the spectrogram loss has a clear advantage in capturing the changes in the system over time. We compare the frequency content of the prediction vs. the actual by running the spectrogram on all the 20% test part of the dataset, and also on the corresponding predictions.

Spectrogram Comparison

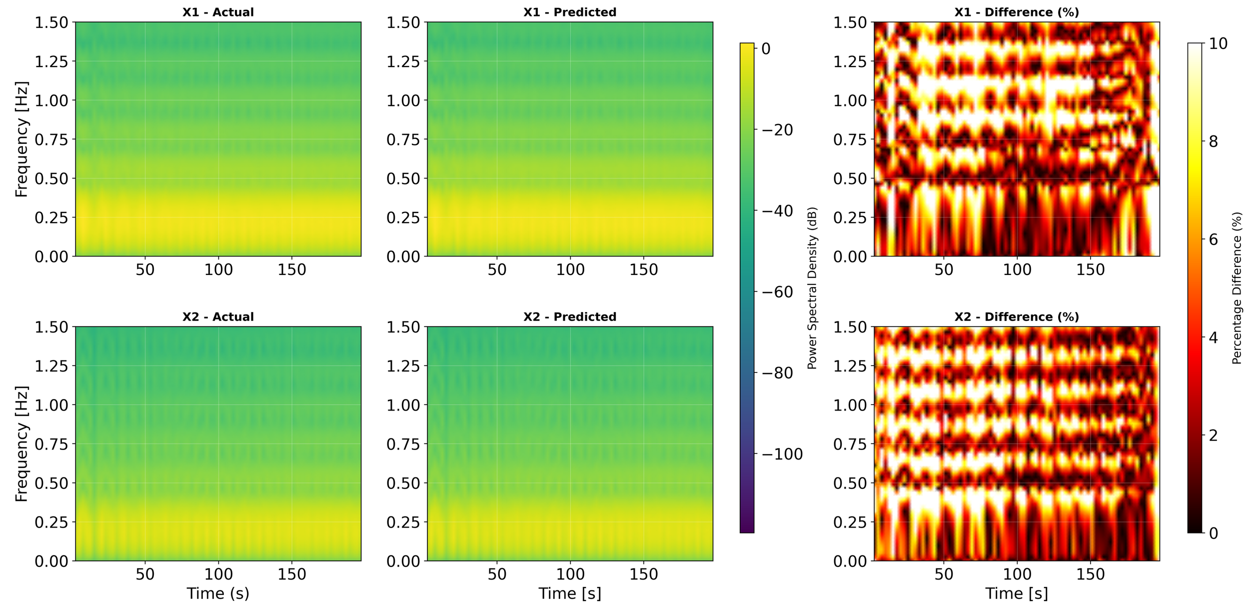


Figure: The average spectrogram on the 20% test

The presented error is based on the difference in the power spectrum. This shows a good match between the prediction and actual frequency content. The spectrogram loss works here as it pushes the training algorithm to consider the frequencies in the Fourier domain, where they are not considered with the MSE loss. This gives the FNO the ability to digest the nonlinear time-varying system.

Conclusion and Discussion

- Out-of-the-box FNO with MSE loss works well for a linear system. We showed the capabilities of FNO on a linear 2-DOF and an OWT simulation output.
- The same architecture could not achieve desirable accuracy regardless of the training dataset size when the model is nonlinear. We showed this by training the FNO on four different sizes of datasets.
- As the option of increase in the dataset size did not work, and the results show further inclusion of the frequency is required, we added the spectrogram loss to the FNO.
- The FNO with spectrogram loss showed improvement in predicting the nonlinear time-varying 2-DOF mass-spring system.
- Our tests show that if the input force has a changing frequency in the dataset, the training and testing errors will not reach an acceptable level. This needs further investigation.
- Calculating spectrograms is computationally expensive, especially when it adds up in a training process. Improvement on that front needs further investigation.

Future Work Questions

- How can the FNO setup with spectrogram loss performance be improved for systems where the frequency content of the forcing function is changing?
- What is the prediction of the trained FNO model when the forcing function is out of the training dataset? How can we improve it?
- What other loss functions can help FNO capture the dynamic frequency content? Which are computationally less expensive?

Acknowledgement

The authors acknowledge the Tufts University High Performance Compute Cluster `https://it.tufts.edu/high-performance-computing`, which was utilized for the research reported in this poster.

References

[1] Q. Cao and et al. *Comput. Struct.*, 291:107228, 2024. doi: 10.1016/j.compstruc.2023.107228.
[2] D. Chakraborty and et al. *arXiv:2502.00472*, 2025.
[3] T. J. Grady II and et al. *Comput. Geosci.*, 178:105402, 2023. doi: 10.1016/j.cageo.2023.105402.
[4] N. Kovachki and et al. *arXiv:2108.08481*, 2024. doi: 10.5555/3648699.3648788.
[5] S. Lanthaler and et al. doi: 10.48550/arXiv.2405.02221.
[6] Z. Li and et al. Fourier Neural Operator for Parametric Partial Differential Equations. *International Conference on Learning Representations*, 2021. URL `https://openreview.net/forum?id=c8P9NQVtmn0`.
[7] M. G. Pedersen and et al. 4 2024. doi: 10.11583/DTU.24460090.v1.
[8] M. M. Rahman and et al. *Phys. Plasmas*, 31(12):123902, 2024. doi: 10.1063/5.0232503.
[9] I. M. Sobol'. *Zhurnal Vychislitel'noi Matematiki i Matematicheskoi Fiziki*, 7(4), 1967.