ISBN: 978-3-95450-247-9 ISSN: 2673-5490 doi: 10.18429/JACoW-IPAC2024-MOPS10

KOOPMAN OPERATOR METHOD FOR NONLINEAR DYNAMICS ANALYSIS USING SYMPLECTIC NEURAL NETWORKS *

K. Anderson[1] †, Y. Hao[1,2]
[1]Facility for Rare Isotope Beams, Michigan State University, East Lansing, MI, USA
[2]Brookhaven National Lab, New York, USA

Abstract

Data driven methods have proved to be a useful tool for analyzing Hamiltonian systems. The symplectic condition is a strong constraint on Hamiltonian systems and it is therefore useful to implement this constraint into neural networks to ensure the accuracy of long term predictions about the system. One such method is the use of SympNets [1], linear, activation, and gradient layers that guarantee the symplectic condition is met without the use of symplectic integration or extra gradient calculations. Data driven methods are also useful for calculating Koopman operators [2] which aim to simplify nonlinear dynamical systems into linear ones. By using SympNets, one can ensure that the transformation described by the Koopman operator is symplectic, reversible, and more easily trained.

THE PROBLEM SETUP

We tested the network on two different 2D nonlinear maps, the Standard map, and the Octopole McMillan map. The Standard map is described by the following:

$$p_{n+1} = p_n + K \sin\theta_n \tag{1}$$
$$\theta_{n+1} = \theta_n + p_{n+1} \tag{2}$$

where θ and p are then acted on by modulus 2π. The Octopole McMillan map [3], which will be called the McMillan map for simplicity, is described by the following:

$$q_{n+1} = p_n \tag{3}$$
$$p_{n+1} = -q_n - \frac{2\epsilon p_n}{p_n^2 + \lambda} \tag{4}$$

We are seeking a transformation of variables through the Koopman Operator [2] which has the following properties:

$$K : \mathbb{R}^2 \to \mathbb{C}^2 \tag{5}$$
$$\mathbf{W} = K(\mathbf{X}) \tag{6}$$
$$\mathbf{W}_{n+1} = \begin{pmatrix} e^{2\pi i\nu} & 0 \\ 0 & e^{-2\pi i\nu} \end{pmatrix} \mathbf{W}_n \tag{7}$$

where $\mathbf{X} \in \mathbb{R}^2$ is the original phase space coordinates and $\nu \in [0, 1]$ is the betatron tune of the system.

Since many nonlinear systems that describe accelerators are non-integrable, there is not a transformation that can perfectly transform the entire phase space into a pure rotation. It is possible however, to find such a transformation in a small amplitude range. We aimed to find such a transformation and the area that it could reasonably cover.

NETWORK SETUP

Data

The models described in the following sections were trained on a single initial condition. The idea being that if all the data comes from a single orbit, then the model would have an easier time converging to a solution. This was done by doing a number of iterations of the given map, then the initial conditions would be the phase space coordinates at iteration n and the output would be the positions at iteration $n + 1$. The same tracking data would be used for additional data points a second time with the difference being the input would be iteration $n+1$ and the output would be iteration n.

Model

Similar to a traditional neural network, the Koopman operator network consists of a series of linear and activation layers. The layers are split into three main sections, one is the encoder which aims to find the transformation from the phase space of the tracking data into a space where the motion is a pure rotation. The second is the frequency layers, these layers aim to predict the frequency of the pure rotation. They are composed of traditional network layers as the symplectic condition is not needed. Then there is the decoder which aims to be an inverse of the encoder.

With the data being nonlinear tracking data, the input being an initial position in phase space with an additional feature of ± 1 which indicates if the output vector should be one iteration of the map forwards, or backwards. This feature was not passed into the encoder or the frequency network, it would instead be used to determine if the linear transformation would be clockwise (forward in time) or counterclockwise (backwards in time). The full forward function takes the input vector, puts it through the encoder and frequency layers, the coordinates from the encoder output are rotated by the angle given by the output of the frequency layers, and the rotated point is inputted into the decoder. The output should be either one iteration of the map forward or backward.

* This work is supported by DOE office of science, with award number DE-SC0023722.
† anderske@frib.msu.edu

ISBN: 978-3-95450-247-9 ISSN: 2673-5490 doi: 10.18429/JACoW-IPAC2024-MOPS10

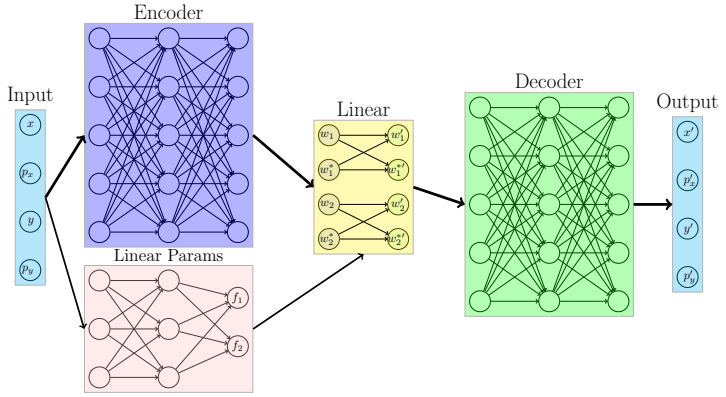

Figure 1: An illustration of a 4-D representation of the neural networks that were used.

$$\mathcal{L}_n^{\text{up}} = \begin{pmatrix} I & 0/S_n \\ S_n/0 & I \end{pmatrix} \cdots \cdots \begin{pmatrix} I & 0 \\ S_2 & I \end{pmatrix} \begin{pmatrix} I & S_1 \\ 0 & I \end{pmatrix} \begin{pmatrix} p \\ q \end{pmatrix} \quad (8)$$

$$\mathcal{N}^{\text{low}} \begin{pmatrix} p \\ q \end{pmatrix} = \begin{pmatrix} p \\ \text{diag}(a)\sigma(p) + q \end{pmatrix} \quad (9)$$

The difference comes from using SympNets [1] versions of these layers instead of the traditional ones. This gives a few advantages, one being that the transformation found in training obeys the symplectic condition required for Hamiltonian systems. Another is that due to the configuration of these layers, an exact inverse of the encoder can be systematically constructed from the weights of the layers. This means that there is no need to have additional network layers to create a decoder, which would slow training, and no approximate methods need to be used to decode the data either. Figure 1 shows an example of this structure in 4-D.

One assumption made was that the coefficients of the transformation would not be very large. So a weight decay was added to the Adam optimization method in PyTorch [4]. This adds the L2 norm of the model parameters into the loss function to encourage smaller sized parameters. This prevented a previous issue where the parameters would grow exponentially.

Equations 8 and 9 show examples of a SympNet upper linear layer of size n and a lower activation layer with function σ. a is an unbounded parameter of the diagonal of an $m \times m$ matrix and S_j are $m \times m$ symmetric matrices, where m is the number of spacial dimensions and p and q are $m \times 1$ vectors with momentum and position data respectively. For our models, the encoder was composed as a lower linear layer, an upper activation layer, an upper linear layer, a lower activation layer, and an upper linear layer. PyTorch [4] dropout layers were included to prevent overfitting.

The frequency is retrieved from a traditional network comprised of a linear layer to go from 2 input parameters to 100 output parameters, a hyperbolic tangent activation layer, another linear layer with 100 inputs and outputs, another hyperbolic tangent activation layer, and a final linear layer to go from 100 inputs to 1 output, the tune.

RESULTS

Standard Map

For the standard map, a smaller network was chosen to reduce training time. The encoder was a SympNet network, the linear layers were all 8 alternating upper and lower symplectic matrices as described in [1]. The activation layers were all hyperbolic tangent layers, again described in [1]. This comes out to 28 parameters for the encoder / decoder.

An approximate constant of motion was calculated based on a single small amplitude orbit with $K = -0.5$. Figure 2 (a) shows multiple orbits in the standard map phase space. Figure 2 (b) shows the same orbits after being acted on by the encoder. It is easy to see that the lower amplitude orbits are very close to pure rotations and variations are seen as the amplitude increases. This is not surprising as a singular transformation that would linearize the motion cannot exist for the whole phase space. The amplitudes are plotted by turn in Fig. 2 (c) which shows a more clear picture of this effect. Ideally the line width would be zero but we can see the variations in the constant of motion increasing by amplitude. The widths of each constant as well as their variances is plotted by amplitude in Fig. 2 (d).

One issue that still needs to be addressed is the inaccurate tune predictions. The predicted tune very quickly deviates from the numerical analysis of fundamental frequencies algorithm (NAFF) [5] when away from the small orbit it was trained on. This is not surprising but needs to be addressed when making future improvements to the model such as using multiple orbits to have the training data cover a larger area of the phase space.

McMillan Map

For the McMillan map, the network had the same structure as the standard map, but with 9 matrices on the linear layers in the encoder. This comes out to 31 parameters for the encoder/decoder. The parameters for the map were $\lambda = 1$ and $\epsilon = 0.1$ and the model was trained on 1,500 iterations

ISBN: 978-3-95450-247-9 ISSN: 2673-5490 doi: 10.18429/JACoW-IPAC2024-MOPS10

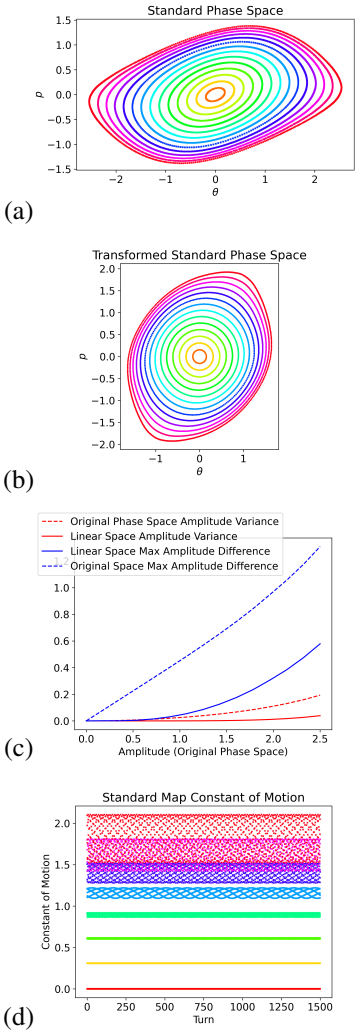

Figure 2: (a) Tracking data for orbits of the standard map with different initial θ. (b) Shows the same orbits from (a) after being passed through the encoder. (c) Shows the difference between the maximum and minimum amplitudes (approx constant) as well as the amplitude variances plotted against the initial amplitude. (d) Shows the amplitudes (approx constants) plotted by iteration for every other orbit shown in (a)/(b).

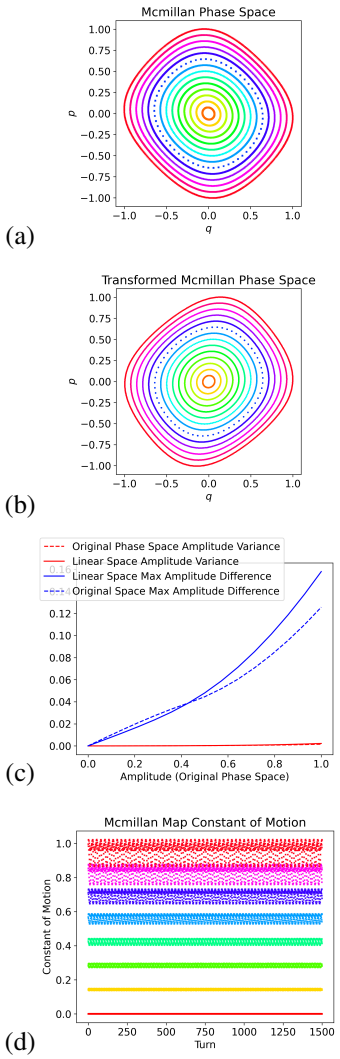

Figure 3: (a) Tracking data for orbits of the McMillan map with different initial q. (b) Shows the same orbits from (a) after being passed through the encoder. (c) Shows the difference between the maximum and minimum amplitudes (approx constant) as well as the amplitude variances plotted against the initial amplitude. (d) Shows the approximate constant plotted by iteration for every other orbit shown in (a)/(b).

of the orbit of $q_o = 0.3$ and $p_o = 0$. We see results similar to the standard map, but the encoder diverges from an ideal transformation faster as the amplitude increases. So a more complicated network may be needed to account for the amplitude dependency. This could be something like using multiple operators for different amplitudes and having some continuity constraint between them. The same plots given for the standard map are given for the McMillan map in Fig. 3.

CONCLUSION

The use of machine learning models can aid in finding approximate constants of motion in a Hamiltonian system. The use of SympNets in the encoder / decoder allows for more efficient training. Future work needs to be done to find how many parameters need to be used to cover a larger area as well as account for amplitude dependencies in the transformation. It appears that while this number of parameters may be adequate to predict a single orbit, a more complicated network would be needed to cover a larger area of amplitudes. The significant error in the frequency predictions also needs to be addressed.

REFERENCES

[1] P. Jin, A. Zhu, G. E. Karniadakis, and Y. Tang, "SympNets: Intrinsic structure preserving networks for identifying Hamil-

ISBN: 978-3-95450-247-9 ISSN: 2673-5490 doi: 10.18429/JACoW-IPAC2024-MOPS10

tonian systems", *arXiv*, 2020. doi:10.48550/arXiv.2001.03750

[2] S. L. Brunton *et al.*, "Modern Koopman theory for dynamical systems", *arXiv*, 2021.doi:10.48550/arXiv.2102.12086

[3] T. Zolkin, S. Nagaitsev, and I. Morozov, "McMillan map and nonlinear Twiss parameters", *arXiv*, 2022. doi:10.48550/arXiv.2204.12691

[4] A. Paszke *et al.*, "PyTorch: An Imperative Style, High-Performance Deep Learning Library", *arXiv*, 2019. doi:10.48550/arXiv.1912.01703

[5] J. Laskar, "The chaotic motion of the solar system: A numerical estimate of the size of the chaotic zones", *Icarus*, vol. 88, no. 2, pp. 266–291, 1990. doi:10.1016/0019-1035(90)90084-M