

A Performance Benchmark of Formulated Methods for Forecast or Reconstruct Trajectories Associated to the Process Control in Industry 4.0

Davi Neves¹^a and Ricardo Augusto Rabelo Oliveira²^b

¹*Department of Production Engineering, University Federal of Ouro Preto, Ouro Preto, Brazil*

²*Department of Computing Science, University Federal of Ouro Preto, Ouro Preto, Brazil*

Keywords: Dynamic Systems, Koopman Operator, Reinforcement Learning, Neural Networks, Topological Measures.

Abstract: Manufacturing processes are generally modeled through dynamic systems, whose solutions establish a tool for control theory, essential in the elaboration of industrial automation, a pillar of the fourth revolution. Understanding and mastering these technological procedures correspond to the ability to determine and analyze the solutions of a system of differential equations, in order to deploy smart devices in a production line, such as the robotic arm, because this trajectories can be always associated with the running of any equipment. Currently there are many formulated methods to determine (or forecast) these curves, through numerical or stochastic tools, the focus in this work are those capable of reconstructing a state space, such as the Koopman's operator, convolutional neural network and reinforcement learning technique. Therefore, based on the solutions provided by these methods, a benchmark will be assembled to compare them, using topological measures such as Shannon entropy, Lyapunov exponent and Hurst coefficient, thus defining the effectiveness of each one.

1 INTRODUCTION

The automation of manufacturing processes dates back to the first industrial revolution (Clark, 2014), when thermal machines mechanized tasks once performed by hands, expediting and improving industrial production; now, facing the fourth revolution supported by electronic devices, an efficiency growing is elicited and consequently to emerge great challenges for scientists and engineers (Xu et al., 2018a; Prisecaru, 2016; Xu et al., 2018b).

Primary challenge in the automaton process is the fit of its results, considering the input values in system that represents this process; studies that approach this topic are classified into the control theory, an applied area of dynamical systems (Nise, 2020; Rodic, 2009), which are formally represented by coupled differential equations (Haddad and VijaySekhar, 2011; Salle, 1976).


Normally, the control theory of automatized manufacturing processes correspond to the solutions and analysis of the differential equations that model them; these systems are usually named like governing equa-


tions and their solutions are computed with numerical methods (Stuart and Humphries, 1998; Beyn, 1991) or block diagrams (Nise, 2020).

An ordinary differential equation express the behavior of a curve, this way the solution of coupled equations correspond the union of distinct curves, thus forming a n-dimensional surface known as manifold. These structures are studied in topology, a theory that provide tools to the understand of dynamic systems (Akin, 2010; Materassi and Innocenti, 2010).

Curves (trajectories) referring to the solutions of a dynamical system can be illustrated in state space, rather than real space, because in this hidden symmetries are highlighted (Prince, 1982; Levi and Winternitz, 1996). Koopman and von Neumann observed this important detail and based on that they formulated a theory (Koopman, 1931; v. Neumann, 1932a; v. Neumann, 1932b) with enormous effectiveness and usefulness for data-driven analysis (Williams et al., 2015; Mezić, 2013; Proctor et al., 2018).

Using Koopman operator is possible to reconstruct the trajectories in state space, which constitutes a method to determine predictions regarding the system's behavior and hence a control tool for the corresponding process (Li and et al, 2017; Bruder et al.,

^a <https://orcid.org/0000-0002-3144-0207>

^b <https://orcid.org/0000-0001-5167-1523>

2019).

Due to importance of this reconstruction process other methods have been developed, among which it is worth mentioning the Packard theory that use delay coordinates (Packard and et al, 1980) and those that use neural networks, with convolutional (Hauser and et al, 2019; Teng and Zhang, 2019) or autoencoders structures (Otto and Rowley, 2019; Almazova et al., 2021; Champion and et al, 2019).

To complement the control theory approach referent to trajectories analysis, reinforcement learning should also be included, because in this method the most probable trajectory is determined by a Markovian decision process. Although this methodology (stochastic) is not in the same context (deterministic) as the ones mentioned before, its leading objectives are equivalents: compute the path most effective to carry out a process (Sutton and Barto, 2018; Kaelbling et al., 1996).

In order to elaborate an analysis that cover all these methods, a robotic arm was selected like object of study, due to your wide bibliographic reference and the inherent complexity its mathematical model, a double pendulum (chaotic behavior), thus warranting the suitable requirements for build the benchmark.

We started with a theoretical review of dynamic systems and their fasten relationship with process control, followed by the Koopman operator approach, in next section we present essential concepts about models that use reinforcement learning and then we complement the description of the methods to trajectories rebuild with a decoder type neural network.

The theoretical overview will be concluded with the presentation of the effectiveness measures, using topological methods such as Lyapunov exponent, Hurst coefficient and Shannon entropy (Akin, 2010); we will also elucidate Pearson's statistical coefficient (Montgomery et al., 2009), which makes it possible to analyze the correlation between real and predicted trajectories.

In results section we will present the simulation environment, in which highlight the hardware and softwares used, proceeding to the proposed analyses, where we will evaluate the effectiveness of each method mentioned, always establishing a relationship with the robotic control.

2 DYNAMICAL SYSTEMS AND CONTROL

Dynamical systems are essential in the control of industrial devices and processes, because their governing equations compose the theoretical formulation of

these events. Analysis can be performed through the corresponding block diagram or by state space (Nise, 2020), for nonlinear equations the latter is the alternative most pertinent (Stuart and Humphries, 1998; Beyn, 1991), since symmetries and topological measures are highlighted, which contribute significantly to the control of projects in industry 4.0 (Xu et al., 2018a).

A trivial case study that provides the elucidation of these controls is robotic arm, whose objective is to reach a target and then move it; this process, or device, can be modeled using equations that govern the motion of a double pendulum, and despite being a three-dimensional system, for simplicity without lose the essence, can be modeled in two dimensions.

According to these considerations, the differential equations must couple the angular variable (θ) and their respective velocity (ω), resulting in complex expressions that can be presented in a concise way:

$$\frac{\partial \vec{\theta}}{\partial t} = \hat{F}(\vec{\theta}, t, \alpha) \quad (1)$$

In (1) θ and ω are coupled into $\vec{\theta}$ vector, while the equations system is represented by functional \hat{F} , named like field, because it's associated with the vector field of the states space corresponding.

Time is illustrated by the dependent variable t and the system parameters are represented by α ; also, regarding (1), is worth noting that this system refers to continuous states, for the discrete case a more adequate formulation would be:

$$\vec{\theta}_{k+1} = \hat{F} \cdot \vec{\theta}_k \quad (2)$$

In the control theory θ_{k+1} represents the output of the system (posteriori signal) and θ_k is called of input (piori signal), while \hat{F} operator promote state evolution (θ); to establish the control this operation must be linearized, making outputs directly associated with input signals, thus elucidating the scope of this theory.

For linear systems, the control methodology is already well established, so there are several tools, theoretical and practical, to adjust input signal until the required output is obtained, within an acceptable margin of error, however, for nonlinear systems the inherent complexities usually affect your control.

3 KOOPMAN OPERATOR METHOD

Nonlinear systems are approached by control theory from linearization methods, such as the Taylor series truncated, that has inspired numerous and effec-

tive procedures, however, in this section, will be presented dimension reduction methods using coordinate transformations, like the Packard's work (Packard and et al, 1980).

Packard's work inspired others (Broomhead and et al, 2020; Schmid, 2010), in which the dimension reduction, such as principal component analysis (PCA), was combined with the fast Fourier transform (FFT) to formulate what has been termed by dynamic mode decomposition (DMD) (Proctor et al., 2018), that therefore was associated with Koopman operator (Koopman, 1931), method developed in 1931 for analyze the time evolution of observables.

$$g(\vec{\theta}_{k+1}) = \hat{K}.g(\vec{\theta}_k) \quad (3)$$

Fundamentally the Koopman operator (\hat{K}) works in the space of observables, relatives to the system's states; considering that the dimension of this space is infinite, the linearization process will be feasible, thus if the measure function was $g(\vec{\theta})$ the evolution of the observables will be given by equation (3).

The key point of this methodology is that even if the operator \hat{F} , which represents the dynamical system, was nonlinear, the Koopman operator will provide a linear way of time evolve states, thus enabling a way to reconstruct the trajectories in respective states space (control).

In its primordial practical formulation, the determination of the elements of this operator required an adequate choice of basis functions for the proper representation of the \hat{K} matrix, which was often infeasible, but currently the construction of these base functions is referred to deep learning methods (Li and et al, 2017), using neural networks to estimate their eigenvectors and eigenvalues.

Addition to the use of deep learning, which expanded the applicability of this method (Proctor et al., 2018), in industry 4.0 context the control usually can be data-driven (Li and et al, 2017), thus referring to the DMD method, which enables the construction of the Koopman operator using data series (Williams et al., 2015):

$$\hat{K} \approx \hat{T}'\hat{T}^+ \quad (4)$$

In (4) \hat{T}' is the matrix of posteriori states and \hat{T}^+ is the Moore-Penrose pseudoinverse matrix of previous states (\hat{T}) (Mezić, 2013; Li and et al, 2017).

As the focus of this work is to evaluate methodologies for robot's arm control, the Koopman operator will then be built using data from double pendulum simulations, from ordinary differential equations integration referring to this model.

4 REINFORCEMENT LEARNING

Among many machine learning techniques, control theory can be meet on the reinforcement learning, in which an agent interacts with the environment and as result a learning is constituted, represented by a table of rewards referred like Q-table (Sutton and Barto, 2018; Kaelbling et al., 1996).

Formally this learning process can be analyzed from Bellman's equation (Sutton and Barto, 2018), referring to the current state-value $v(\theta)$:

$$v(\theta) = \max_a (R(\theta, a) + \gamma.v(\theta')) \quad (5)$$

Agent's action must maximize the sum between the archived state-value $v(\theta')$ for all possibles next states θ' and the current reward considering all possibles actions at state θ . Discount factor (γ) was introduced to $v(\theta)$ focus on immediate instead of future rewards.

After this training, the most efficient trajectory is performed, or in other words the most probable path, considering a policy $\pi(\theta/a)$ that defines the probabilities of each action a in a respective state θ , then, setting this methodology like this, your objective is find the optimum policy π which leads to the build of the most probable trajectory for the robotic arm.

Although this description elucidates the similarities between the methods discussed in this text, it should be noted that in this case the essence of the approach is stochastic, different from deterministic character adopted by dynamical systems theory.

Currently, there have been improvements with the use of neural networks, which replace the Q-table in the agent's learning process, that is, in the determination of the most probable trajectory. In this approach, each agent state will be used as an input signal in the neural network and the output will be the reward referring to an action in this state (Kaelbling et al., 1996).

In this work, trajectories for a robotic arm will be constructed considering the previous knowledge of the object's position to build the Q-table, methods using the neural networks will also be evaluated, with soft actor critic algorithm (SAC) (Sutton and Barto, 2018), that optimizes a stochastic off-policy, in order to carry out the integration of this technique with others, thus providing its improvement and expanding its applicability.

5 DECODER NEURAL NETWORKS

Trajectories are essentially graphical representations of a movement, due to this the most suitable neural networks to deal with these are convolutional ones, as will be explained in this section.

Without dodging of the work's objective, the convolutional neural network (CNN) corresponds to a structure in which an image will be initially polished and flattened, thus forming a first-order tensor, fitted like an input signal of this neural network whose logistical output normally results in the classification of this image (Teng and Zhang, 2019).

Although convolutional neural networks can be used to analyze dynamical systems (Teng and Zhang, 2019; Otto and Rowley, 2019; Almazova et al., 2021), the reconstruction of paths in the state space is more consistent with the structure called like autoencoder (fig. 1-A), in which a neural network (encoder) compresses an image in a latent space to that then a second network (decoder) make another image, corresponding to the one used as input (Otto and Rowley, 2019; Almazova et al., 2021).

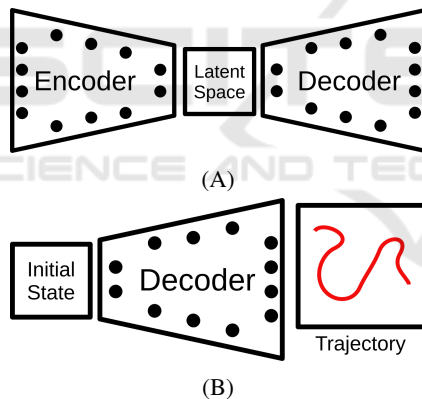


Figure 1: Representation of neural networks structures, in (A) an autoencoder is illustrated and in (B) the structure used in this work.

As the purpose of this work is to analyze techniques capable of reconstructing trajectories in the state space and considering the high cost of autoencoders, a reduced structure was elaborated, constituted only by a network similar to the decoder (fig. 1-B), using as latent space the initial conditions of the problem in question, that is, the initial positions of the object and robotic arm, producing as output an image that illustrates a trajectory in state or real spaces.

Using several trajectories referring to different initial positions, for the target and robotic arm, a decoder neural network was trained to construct (not reconstruct) the trajectories relevant to the aforementioned

control. In the training, initial conditions were put in the format of a 2x2 matrix, whose elements represented the positions of the arm and the object, then this matrix was converted into the image of a trajectory consistent with the pendulum solution.

6 TOPOLOGICAL MEASURES

Visual similarity (or difference) between two curves can lead to hasty and wrong conclusions, with regard to the evaluation of the reconstruction capacity of a method, however topological analysis provides tools to perform the appropriate analysis, with precision and efficiency.

Pearson's correlation coefficient may be the first tool used (Montgomery et al., 2009), because despite its statistical formulation, its value indicates the coherence between two curves, that is, positive values close to unity correspond to similar curves, on the other hand, negative values close to unity indicate an inverse behavior between two curves, finally, to the other values no correspondence is verified.

$$\rho = \frac{COV(\theta_1, \theta_2)}{\sqrt{VAR(\theta_1) \cdot VAR(\theta_2)}} \quad (6)$$

The previous expression represents the calculation of Pearson's coefficient, using the covariance between two data series, the first (θ_1) correspond to original trajectory and the second (θ_2) to forecast (rebuild). The denominator is defined by the square root of the variances referring to these trajectories.

Another important measure is Shannon entropy, which defines the amount of information contained in a data series, considering the occurrence of an event, its basic formulation is as follows (Materassi and Innocenti, 2010):

$$S = \log \left(\frac{P_{i+1}}{P_{i-1}} \right) \quad (7)$$

In (7) P_{i-1} is the priori probability of the event's occurrence and P_{i+1} is the posteriori probability.

According to this formulation, is evident that this value indicates the complexity of respective data series, as it quantifies the information (bits) needed to describe the occurrence of this event, however, it should be noted that, in this work only the similarity of these measures will be evaluated, as this qualitative assessment is sufficient for the proposed objectives.

Lastly, two other values will also be determined, the larger Lyapunov exponent and the Hurst coefficient (Materassi and Innocenti, 2010), the first being a measure of data's chaoticity and the second a measure

of its fractal dimension, which refers to the complexity of this data series, however, in this work these values will be evaluated only qualitatively, that is, similar values indicate similar trajectories.

7 RESULTS AND DISCUSSIONS

7.1 Simulation Environment

The results presented in this section were obtained in a cluster assembled using the Docker engine software, with hardware composed by Intel i7 processor (3.8 GHz) and 16 GB of RAM, aided in graphics processing by an Nvidia Jetson Nano card with 128 cuda cores (1.4 GHz) and 4GB of RAM.

Codes developed in this work used a series of python libraries, the standards Numpy, Scipy and Matplotlib, for numerical and graphical functionality. Koopman method was deployed using PyKoopman, neural networks were elaborated with PyTorch, while reinforcement learning simulations it used Gym and Stable-Baselines3. The topological measures was computed with Nolds (Harris and et al, 2020; Virtanen and et al, 2020; Hunter, 2007; Paszke and et al, 2019; Schölzel, 2019).

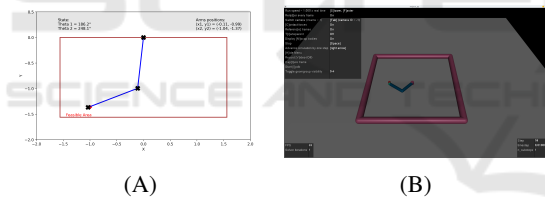


Figure 2: Simulation environment used, (A) the first is an elaboration with Numpy and Matplotlib. (B) The second is called Reacher-v2, deployed in Gym using the Mujoco.

All trajectories that emulate the behavior of a robotic arm were produced in two simulation environments, the first (fig. 2-A) created by the authors only with standard libraries, while the figure B was elaborated into Gym library (with Mujoco (Todorov et al., 2012)); in both there is a area that delimits the target's position.

Movement of both emulators results in chaotic paths that start at the initial position of the robotic arm until it finds the target, whose position constitutes the final point, as is showed in figure 3.

7.2 Koopman Operator

Simulations referring to the Koopman method use numerical solutions of the double pendulum like input,

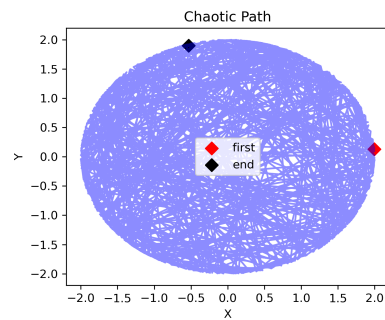


Figure 3: Representation of a not optimized and chaotic path generated by arm's movement, simulated with double pendulum equations.

computed with the Scipy module, from these trajectories and using PyKoopman module one can determine the matrix corresponding to the operator.

Koopman eigenvalues are demonstrated in figure 4, which indicate that the eigenvectors neither grows nor decay, that is, operator is stable, once it's in your adequate form to make forecast calculus.

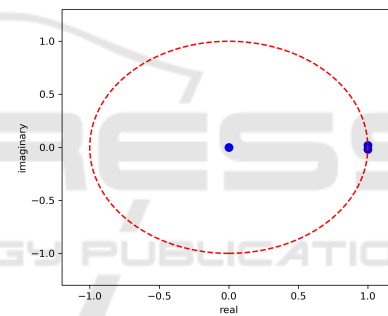


Figure 4: Koopman operator eigenvalues, three are on the unit circle and one is approximately zero.

The results illustrated in figure 5 represent trajectories in the state spaces for the second joint angle (θ_2) and its respective velocity (ω_2). In figure 5-A is the result of the numerical integration, with mass and length equal to one and using the initial conditions: $[0.27, 0.0, 0.42, 0.0]$. The second figure (5-B) is the prediction using the Koopman operator.

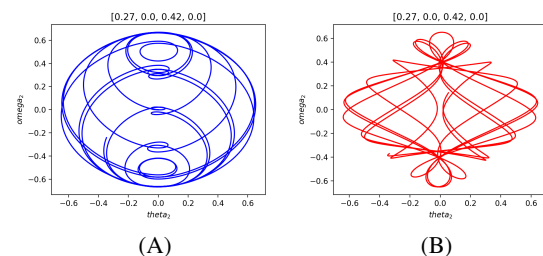


Figure 5: State space corresponding to the angle and its velocity of the second joint. In (A) is the numerical solution and in (B) the prediction respective.

The Pearson correlation coefficient for these two state space trajectories is 0.94 (high), which indicates a strong coherence between these curves, furthermore it is possible to illustrate this result using a correlation matrix (figure 6).

A particular result like the one illustrated in figure 6-A can lead to wrong conclusions, then it is important to remember that a chaotic system is sensitive to the initial conditions, as demonstrated by figure 6-B, which is presented as a correlation matrix for the system that starts at $[-0.32, 0, 0.64, 0]$. In this case the Pearson coefficient is 0.36.

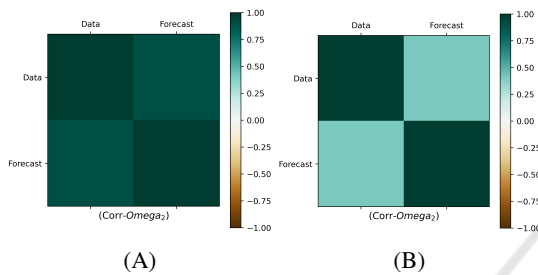


Figure 6: (a) Correlation matrix for the trajectories illustrated in figure 5. (B) Correlation matrix for trajectories with other initial conditions.

The curve with the lowest Pearson coefficient, presented a Hurst coefficient of 0.93 for the actual data and 0.94 for forecasts. The Shannon entropy for the real data was 0.11 and for the predictions 0.15, and finally the Lyapunov exponent in both cases was slightly positive, confirming the chaotic nature of this system.

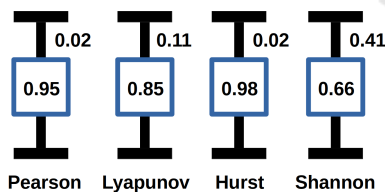


Figure 7: Normalized means and errors referring to topological and statistical measures.

An assemble to evaluate the data roughness was elaborated from a sample with fifty arbitrary initial conditions, which was used to measure the efficiency of the Koopman method, the statistical results are illustrated in figure 7.

Analyzing figure 7, it was observed that the Pearson correlation coefficient is positive and presents an error (standard deviation) of 2%, while the Lyapunov exponent presents an error of 11%, less stable. The Hurst coefficient is very stable, with the same error as the Pearson coefficient, lastly the Shannon entropy was the most unstable, with a big error of 40%, which differs from

previous results.

7.3 Reinforcement Learning

To perform the reinforcement learning method, a discrete state space was elaborated considering a fixed angle step ($d\theta = 0.04$) for each pendulum joint (θ_1, θ_2), then a Q-table was determined, which induced the deployment of a space engine (DC motor) with constrained movements, like a pendulum (fig. 8), thus emulating a deterministically programmed automaton system.

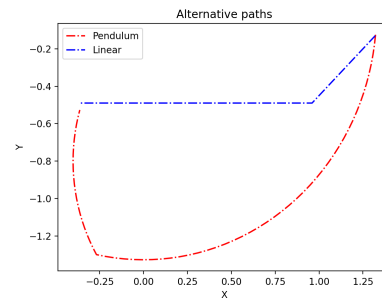


Figure 8: Trajectories for a robotic arm using reinforcement learning (Linear/Blue) and the path like made by a DC motor (Pendulum/Red).

In each simulation was used the environment called Reacher-v2, implemented in the Gym framework (using Mujoco library), as this system is a continuous state space the SAC model was selected, so that the SAC3 method, also deployed in this framework, can be used too.

The neural network associated with the reinforcement learning method (SAC) was trained with 106 steps (workouts), during this process were observed a convergence in the loss function from 20% of the training, then 1000 tests were performed with 100% accurate results.

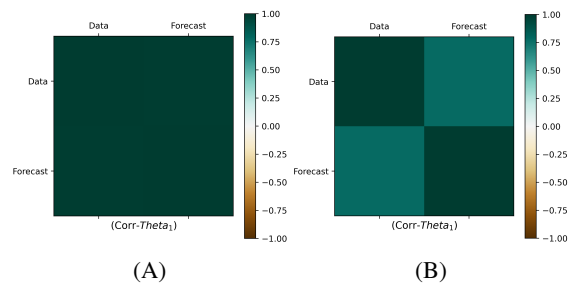


Figure 9: Correlation matrix for the trajectories of reinforcement learning, with (A) high and (B) low coherence, considering different initial conditions.

Similarly to the Koopman method, in this case the Pearson coefficient presented a small variation (1%), thus demonstrating a strong coherence in the trajectories generated by this method; the other topological

measures are shown in figure 10.

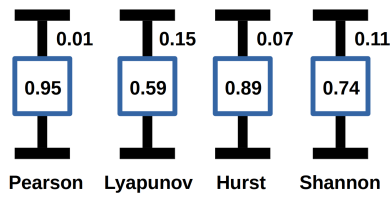


Figure 10: Normalized means and errors for topological and statistical measures.

The Hurst coefficient in this case varied more than in Koopman method, between 0.85 to 0.92, i.e. 7%, as illustrated in figure 10. Shannon entropy varied less than before, around 11%, from 0.71 to 0.78. Lastly, Lyapunov exponent again presented positives values, but with the biggest error, around 15%. Overall, this method performed better than the Koopman method.

7.4 Decoder Neural Network

The last method analyzed is a decoder neural network, used to build trajectories from matrices that represent the initial states, in this case the target and robot arm positions:

$$\begin{bmatrix} X_{target} & Y_{target} \\ X_{arm} & Y_{arm} \end{bmatrix} \quad (8)$$

This initial matrix is firstly flattened to one order tensor, then passed through a neural network with 6 hide layers, the last one being an output with 3969 values. The output then goes through a process of unpolishing and reshape for a 128x128 matrix, which represents the trajectory predicted (fig. 11-B).

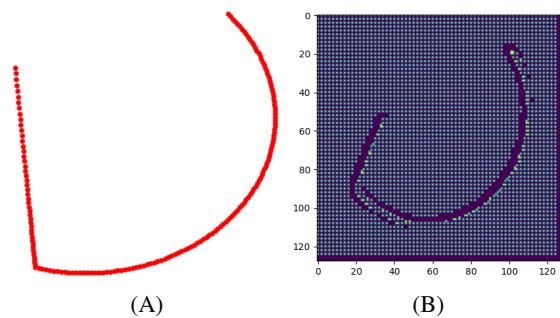


Figure 11: (a) Original pendulum trajectory, path from arm (initial) to the target. (B) Results of the decoder neural network after your training, with the same initial conditions.

In network training 5000 data were used, the supervised learning was performed with one thousand epochs and the loss function converged with approximately 600 steps. Next, the decoder then transformed matrices like (8) in images as the figure 11-B, which

turn represent original pendulum trajectory, like illustrated in figure 11-A.

Fifty tests was performed using this neural network, the results were further processed so that we could determine the topological measurements; using the trajectories, predicted and original, was determined the topological measures for this method, thus defining its ability to reconstruct paths in state space.

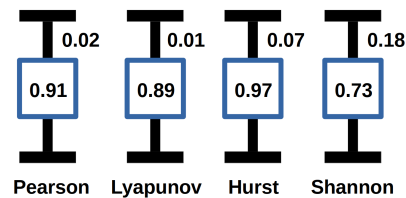


Figure 12: Normalized means and standard deviations for topological and statistical measures.

Analyzing the results (fig. 12), can be observe that the Pearson coefficient is positive and with a very small error (2%). The other values also presented small errors, the Shannon entropy with a error of 18% was the bigger. Lyapunov exponent and Hurst coefficient with 1% and 7% respectively, deviated less then 10%, which confirm the effectiveness of this method.

8 CONCLUSIONS

According to the results presented in the previous figures, it can be concluded that the topological measurements showed that the methods evaluated are effective and despite the amplitude of the results the errors were generally moderate.

Looking at figures 7, 10 and 12, it is worth noting that the average Pearson coefficient in all cases is positive, indicating coherence between the original and predicted curves. The errors of the other measurements corroborate this statement, except for Shannon entropy, which always exceeds 10%.

The chaoticity of this system justifies the discrepancies that occurred, it should still be noted that the decoder neural network results were excellent, however this benchmark does not intend to point out the best method.

Finally, as all methods were effective, it can be concluded that all these are able to reconstruct a trajectory for machine learning, thus stimulating their association.

REFERENCES

- Akin, E. (2010). *The general topology of dynamical systems*. American Mathematical Society, Rhode Island, USA.
- Almazova, N., Barmparis, G. D., and Tsironis, G. P. (2021). Analysis of chaotic dynamical systems with autoencoders. *Chaos: An Interdisciplinary Journal of Nonlinear Science*, 31.103109:1–10.
- Beyn, W.-J. (1991). Numerical methods for dynamical systems. *Advances in numerical analysis*, 1:175–236.
- Broomhead, D. S. and et al (2020). Singular system analysis with application to dynamical systems. *Chaos, noise and fractals*, pages 15–27.
- Bruder, D., Remy, C. D., and Vasudevan, R. (2019). Nonlinear system identification of soft robot dynamics using koopman operator theory. *2019 International Conference on Robotics and Automation. IEEE*, pages 6244–6250.
- Champion, K. and et al (2019). Data-driven discovery of coordinates and governing equations. *Proceedings of the National Academy of Sciences*, 116.45:22445–22451.
- Clark, G. (2014). The industrial revolution. *Handbook of economic growth*, 2:217–262.
- Haddad, W. M. and VijaySekhar, C. (2011). *Nonlinear dynamical systems and control*. Princeton university press, New Jersey, USA.
- Harris, C. R. and et al (2020). Array programming with numpy. *Nature*, 585:357–362.
- Hauser, M. and et al (2019). State-space representations of deep neural networks. *Neural computation*, 31:538–554.
- Hunter, D. (2007). Matplotlib: A 2d graphics environment. *Computing in Science & Engineering*, 9.3:90–95.
- Kaelbling, L. P., Littman, M. L., and Moore, A. W. (1996). Reinforcement learning: A survey. *Journal of artificial intelligence research*, 4:237–285.
- Koopman, B. O. (1931). Hamiltonian systems and transformations in hilbert space. *Proceedings of the National Academy of Sciences*, 17.5:315–318.
- Levi, D. and Winternitz, P. (1996). Symmetries of discrete dynamical systems. *Journal of Mathematical Physics*, 37.11:5551–5576.
- Li, Q. and et al (2017). Extended dynamic mode decomposition with dictionary learning: A data-driven adaptive spectral decomposition of the koopman operator. *Chaos: An Interdisciplinary Journal of Nonlinear Science*, 27.103111:1–10.
- Materassi, D. and Innocenti, G. (2010). Topological identification in networks of dynamical systems. *IEEE Transactions on Automatic Control*, 55.8:1860–1871.
- Mezić, I. (2013). Analysis of fluid flows via spectral properties of the koopman operator. *Annual Review of Fluid Mechanics*, 45:357–378.
- Montgomery, D. C., Runger, G. C., and Hubele, N. F. (2009). *Engineering statistics*. John Wiley & Sons, USA.
- Nise, N. S. (2020). *Control systems engineering*. John Wiley & Sons, New York, USA.
- Otto, S. E. and Rowley, C. W. (2019). Linearly recurrent autoencoder networks for learning dynamics. *SIAM Journal on Applied Dynamical Systems*, 18.1:558–593.
- Packard, N. H. and et al (1980). Geometry from a time series. *Physical review letters*, 45.9:712–716.
- Paszke, A. and et al (2019). Pytorch: An imperative style, high-performance deep learning library. *Advances in Neural Information Processing Systems*, pages 8024–8035.
- Prince, G. E. (1982). Lie symmetries of differential equations and dynamical systems. *Bulletin of the Australian Mathematical Society*, 25.2:309–311.
- Prisecaru, P. (2016). Challenges of the fourth industrial revolution. *Knowledge Horizons - Economics*, 8.1:57–62.
- Proctor, J., Brunton, S. L., and Kutz, J. N. (2018). Generalizing koopman theory to allow for inputs and control. *SIAM Journal on Applied Dynamical Systems*, 17.1:909–930.
- Rodic, A. (2009). *Automation and Control: Theory and Practice*. InTech, Rijeka, Croatia.
- Salle, J. P. L. (1976). *The stability of dynamical systems*. Society for Industrial and Applied Mathematics, Philadelphia, USA.
- Schölzel, C. (2019). *Nonlinear measures for dynamical systems*, volume 1ed. Zenodo.
- Schmid, P. J. (2010). Dynamic mode decomposition of numerical and experimental data. *Journal of fluid mechanics*, 656:5–28.
- Stuart, A. and Humphries, A. R. (1998). *Dynamical systems and numerical analysis*. Cambridge University Press, Cambridge, UK.
- Sutton, R. S. and Barto, A. G. (2018). *Reinforcement learning*. Elsevier, Mountain View, USA.
- Teng, Q. and Zhang, L. (2019). Data driven nonlinear dynamical systems identification using multi-step cldnn. *AIP Advances*, 9.085311:1–8.
- Todorov, E., Erez, T., and Tassa, Y. (2012). Mujoco: A physics engine for model-based control. *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 5026–5033.
- v. Neumann, J. (1932a). Zur operatorenmethode in der klassischen mechanik. *Annals of Mathematics*, 33.3:587–642.
- v. Neumann, J. (1932b). Zusätze zur arbeit „zur operatorenmethode... *Annals of Mathematics*, 33.4:789–791.
- Virtanen, P. and et al (2020). Scipy 1.0: Fundamental algorithms for scientific computing in python. *Nature Methods*, 17.3:261–272.
- Williams, M. O., Kevrekidis, I. G., and Rowley, C. W. (2015). A data-driven approximation of the koopman operator: Extending dynamic mode decomposition. *Journal of Nonlinear Science*, 25.6:1307–1346.
- Xu, L. D., Xu, E. L., and Li, L. (2018a). Industry 4.0: state of the art and future trends. *International Journal of Production Research*, 56.8:2941–2962.
- Xu, M., David, J. M., and Kim, S. H. (2018b). The fourth industrial revolution: Opportunities and challenges. *International journal of financial research*, 9.2:90–95.