Controlling Non-Linear Dynamic Systems Through the Use of RNNs A Generic Methodology

Amirhossein Nayebiastaneh (aminay-2), Fernando Labra Caso (ferlab-1)

***Abstract*— This paper presents a generic methodology for controlling non-linear dynamic systems using Recurrent Neural Networks (RNNs) [1]. Non-linear dynamic systems pose significant challenges for control, as their behavior is highly complex and often unpredictable. Traditional control techniques struggle to effectively handle the non-linearities and dynamic nature of such systems. In recent years, RNNs have emerged as a promising approach for modeling and controlling non-linear dynamics.**

The proposed methodology leverages the power of RNNs to capture and learn the underlying dynamics of the system. The RNN architecture, specifically designed for this purpose, incorporates memory elements that enable it to capture temporal dependencies and make predictions about future states based on past observations.

The paper outlines different steps involved in applying the generic methodology to control non-linear dynamic systems. It discusses the process of data collection, the design of the RNN architecture, and the training procedure. Additionally, the paper addresses the challenges and considerations specific to controlling non-linear dynamic systems using RNNs, including model selection and generalization. To evaluate the effectiveness of the proposed methodology, experiments are conducted on the Van der Pol Oscillator. The paper also discusses the limitations of the methodology and suggests avenues for future research and improvement.

In conclusion, this paper presents a generic methodology for controlling non-linear dynamic systems using RNNs. The proposed approach harnesses the power of RNNs to model complex dynamics, enabling more effective and adaptable control.

I. INTRODUCTION

A dynamic system can be considered from two different perspectives, towards mathematics it is a system in which a function describes the time dependence of a point in an ambient space, such as a parametric curve. In physics, a dynamical system is described as a "particle or ensemble of particles whose state varies over time and thus obeys differential equations involving time derivatives".

Control theory [2] is a field of control engineering and applied mathematics that deals with the control of dynamical systems in engineered processes and machines. Its objective is to develop a model or algorithm governing the application of system inputs to drive the system to a desired state.

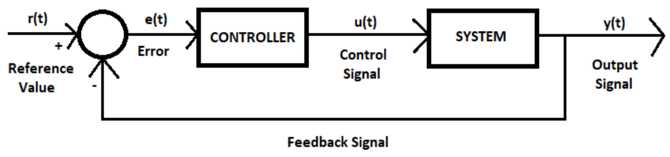

Fig. 1. CLF with a Controller and Dynamic Systems

To do this, a controller monitors the controlled Process Variable (PV) and compares it with the reference or Set Point (SP). The difference between actual and desired value of the process variable, called the Error signal (E), is applied as feedback to generate a Control Action (CA) to bring the controlled process variable to the same value as the set point. We can see an example of a Closed Loop Feedback (CLF) on Figure 1 with a controller and dynamic systems.

II. EXPERIMENTAL SCENARIO

One of the dynamic systems that shows non-linearity is the Van der Pol Oscillator [3], which can be described by the following second-order ordinary differential equation, where $u(t)$ and $y(t)$ are the input and the output of the system respectively and μ is a parameter that controls the strength of the damping and the nonlinearity.

$$\frac{d^2y}{dt^2} = \mu(1 - y^2)\frac{dy}{dt} - y + u(t)$$

The most significant but fairly simple controller is the Proportional-Integrative-Derivative (PID) controller [4], a control loop mechanism employing feedback. A PID controller continuously calculates an error value E as the difference between the desired SP and a measured PV applying a correction CA.

We performed a simple experiment to show how the PID controller behaves when applied over the non-linear Van der Pol oscillator. The results as seen in Figures 2 and 3 show that a simple PID controller is able to make the system converge to the SP.

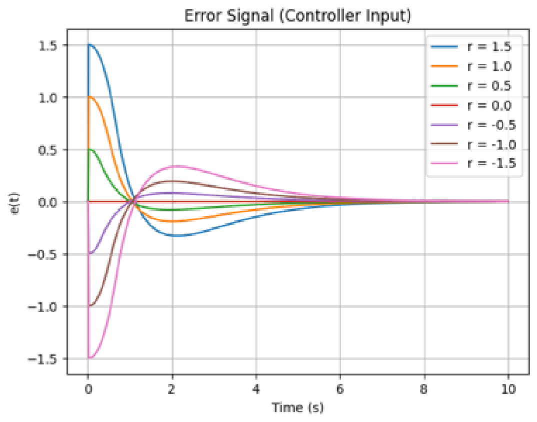

Fig. 2. PID - Error Signal $e(t)$

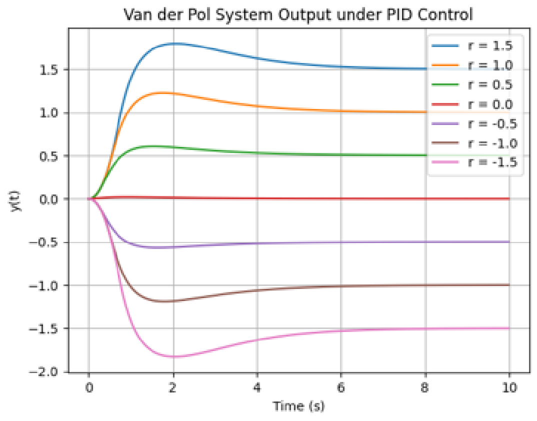

Fig. 3. PID - System Output Signal $y(t)$

In order to train the controller we need the error signal $e(t)$ as the input and the control signal $u(t)$ of the controller as the output, however we do not know $u(t)$ beforehand. The solution is to train another RNN for the system then train jointly both the system RNN with the weights frozen and the controller RNN. Once the controller is trained, we can extract it and test it on the original system.

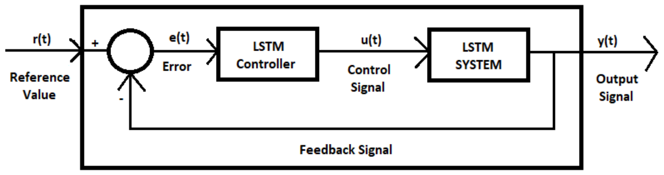

Fig. 4. CLF with an LSTM Controller and Dynamic Systems

III. SYSTEM IDENTIFICATION

We modeled the Van der Pol system with the second order differential equation and from a generated input signal $s(t)$ we obtained the respective output signal from the system $y(t)$ as seen in Figure 5. We trained on 2000 input and output sequences a first model with a Vanilla RNN shifting later to Long Short Memory Unit (LSTM) due to underfitting.

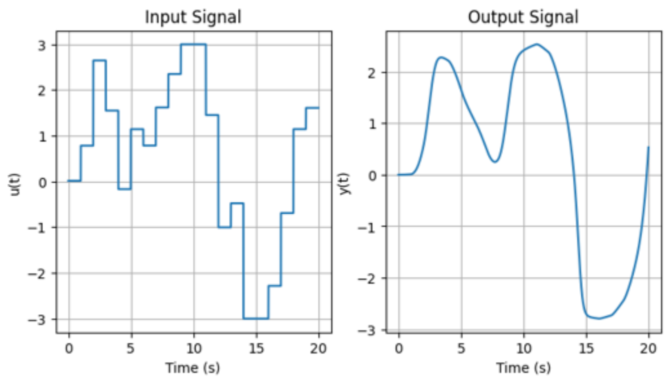

Fig. 5. Van der Pol Input and Output Signals

The characteristics of both RNN and LSTM networks and the training procedure are the following, and the results can be seen in Figures 6 and 7 for both the Vanilla RNN and LSTM.

- Train, Validation, Test Size: [0.6, 0.2, 0.2]
- $N_{epochs} = 300 :: Batch_{size} = 120 :: L_r = 0.001$
- $N_{layers} = 1 :: I_n = 1 :: O_n = 1 :: H_n = 200$
- Activation Function: tanh Hyperbolic Tangent
- Optimizer: Adam

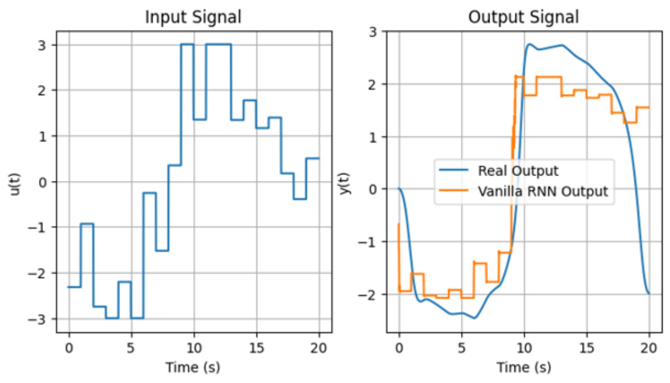

Fig. 6. Vanilla RNN System - Test Loss: 1.319

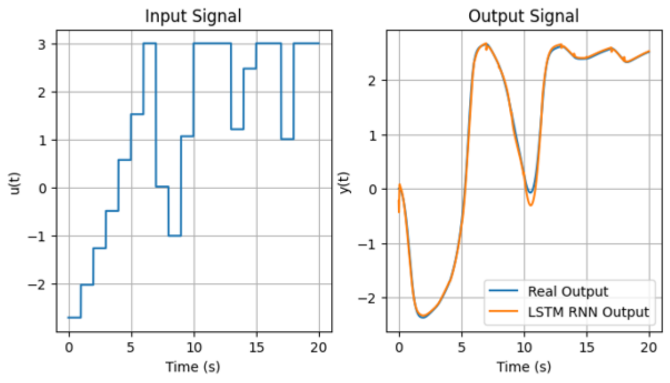

Fig. 7. LSTM System - Test Loss: 0.003

The LSTM architecture seems to outperform the vanilla RNN due to the way it tackles with the vanishing gradient problem and how it is able to retain long range dependencies. For this reason, we have stuck with the LSTM architecture in the following experiments.

IV. CLOSED LOOP RNN

Once we have the system implemented as an RNN, we need to train the controller in a joint network as explained before. The input and output values as seen in Figure 4 are expected to be the same, for what we generated a set of random signals in the amplitude range of $[-3, 3]$ and considered them for both input and output.

We trained a model with two connected LSTMs, one defining the controller and another defining the system. The system was already trained and its weights were frozen.

The controller, referred to as the PID LSTM hereafter, shares the same structure as the System LSTM and underwent fine-tuning on the CLF network. With its implementation we simulated the controlled system and obtained the error (input) and control (output) signals as seen in Figure 8.

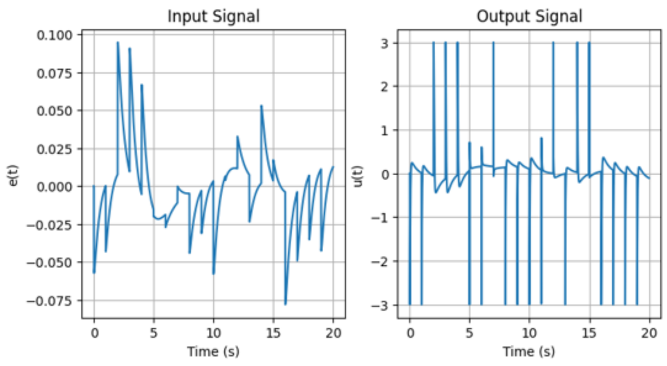

Fig. 8. PID Controller Input and Output Signals

The characteristics of the PID LSTM network and the training procedure were:

- Train, Validation, Test Size: [0.6, 0.2, 0.2]
- $N_{epochs} = 500$ :: $Batch_{size} = 120$ :: $L_r = 0.00001$
- $N_{layers} = 1$ :: $I_n = 1$ :: $O_n = 1$ :: $H_n = 200$
- Activation Function: tanh Hyperbolic Tangent
- Optimizer: Adam

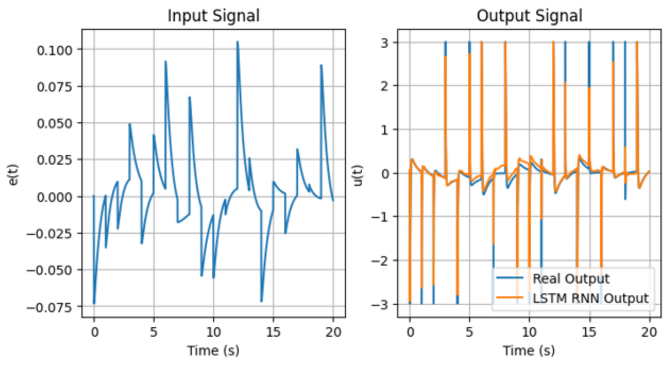

Fig. 9. LSTM PID Controller - Test Loss: 0.031

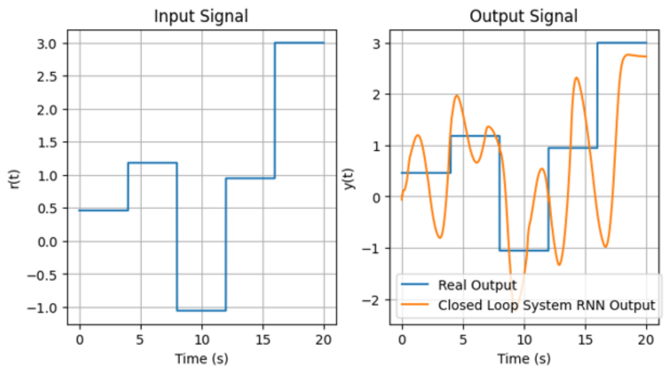

Fig. 10. Fine-Tuned PID LSTM Closed Loop - Test Loss: 1.063

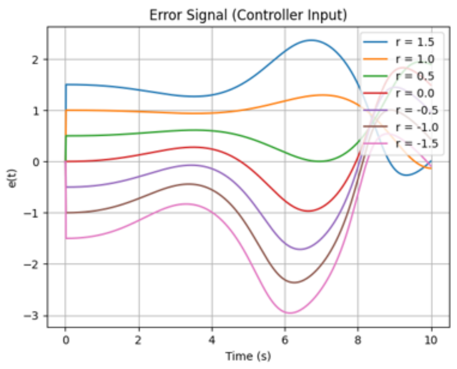

Fig. 11. PID LSTM - Error Signal $e(t)$

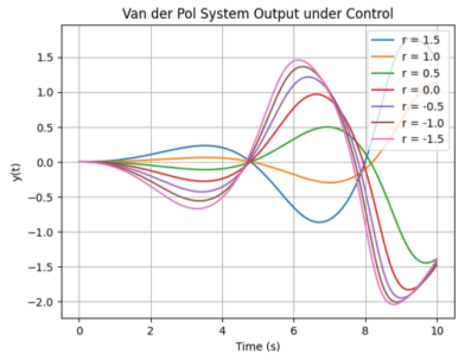

Fig. 12. PID LSTM - System Output Signal $y(t)$

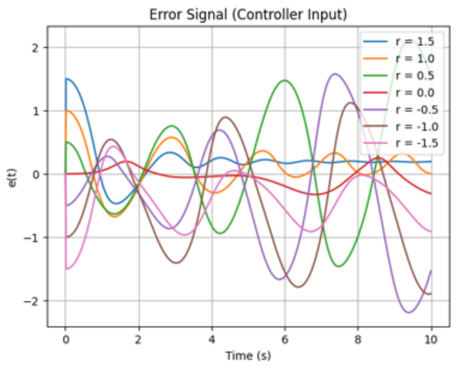

Fig. 13. Fine-Tuned PID LSTM - Error Signal $e(t)$

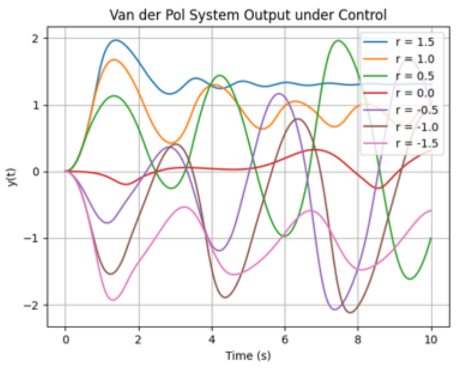

Fig. 14. Fine-Tuned PID LSTM - System Output Signal $y(t)$

As we can see in Figure 9 the results of the trained LSTM for PID identification are very accurate, but when applied on the simulation (Figures 11 and 12) the error does not converge to 0 and the Van der Pol system output does not converge to the reference input values.

However, even if our results might not be the best on the fine tuned PID (Figures 13 and 14) we can see that the original system diverges less compared to Figure 12.

V. OPEN LOOP RNN

After working with the CLF, we considered a different approach. In this case, we thought of having an open loop control system were the controller would be the inverse of the system receiving as input the original output and generating as output the original input. After training, we could then test by inputting the desired output (i.e. the SP) to the inversed system and expecting the original system to reproduce that output as seen in Figure 15.

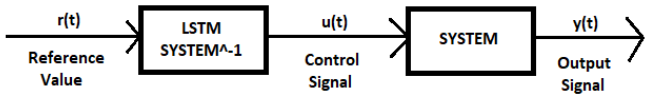

Fig. 15. Open Loop with an Inverse LSTM System

The characteristics of the Inversed System LSTM network and the training procedure were:

- Train, Validation, Test Size: [0.6, 0.2, 0.2]
- $N_{epochs} = 200 :: Batch_{size} = 120 :: L_r = 0.01$
- $N_{layers} = 1 :: I_n = 1 :: O_n = 1 :: H_n = 200$
- Activation Function: tanh Hyperbolic Tangent
- Optimizer: Adam

The results of the inversed system, as seen in Figure 16, show that the system is capable of closely resembling the original input from the original output (considering that the original input has non derivable points).

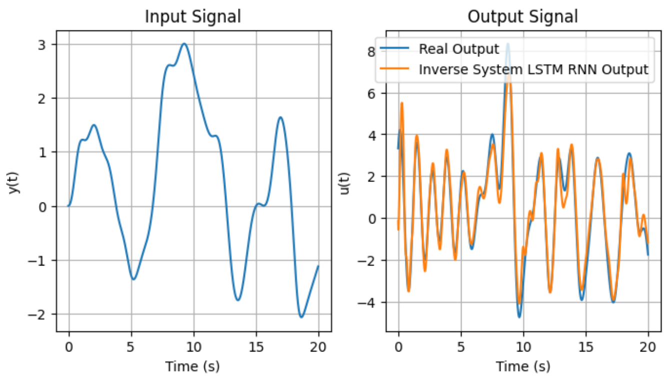

Fig. 16. LSTM Inversed System - Test Loss: 0.387

Once again, we are capable of seeing the power of LSTMs at modeling sequential data. The inverse system was used as a controller on our experimental scenario but the results did not show improvements on the oscillations compared with Figures 12 and 14.

VI. CONCLUSIONS

A set of key ideas can be concluded from this project work, firstly the LSTM shows improved performance when compared to the Vanilla RNN, due to the way it solves the Vanishing Gradient problem. We have seen how it is possible to model the behavior of a non-linear system such as the Van der Pol system and the PID controller through the use of LSTMs.

Secondly, our supervised learning method of training did not succeed in achieving the desired accuracy on RNNs for controlling non-linear systems and it does seem like Reinforcement Learning might be the better training procedure that would achieve this.

As a final point on this project, LSTMs seem capable of capturing long-term dependencies and modeling complex temporal patterns. However they are not able to learn the behavior governing non-linear systems, due to unrepresentative data or due to the nature of the problem.

REFERENCES

[1] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. Illustrated edition. Cambridge, Massachusetts: The MIT Press, Nov. 18, 2016. 800 pp. ISBN: 978-0-262-03561-3.

[2] Katsuhiko Ogata. *Modern control engineering*. 5th ed. Prentice-Hall electrical engineering series. Instrumentation and controls series. Boston: Prentice-Hall, 2010. 894 pp. ISBN: 978-0-13-615673-4.

[3] John Guckenheimer and Philip Holmes. *Nonlinear Oscillations, Dynamical Systems, and Bifurcations of Vector Fields*. 1st ed. 1983. Corr. 6th printing 2002 edition. New York: Springer, Aug. 1, 1983. 478 pp. ISBN: 978-0-387-90819-9.

[4] Karl Johan Åström and Tore Hägglund. *PID Controllers: Theory, Design, and Tuning*. Research Triangle Park, North Carolina: ISA - The Instrumentation, Systems and Automation Society, 1995. ISBN: 978-1-55617-516-9.