ARL-TR-10216 ● SEP 2025

DEVCOM ARMY RESEARCH LABORATORY

Modeling and Optimal Adaptive Control of an Actuator Using Machine Learning Techniques with Long Short-Term Memory Integration

by Richard McKee, D. Johann Djanal-Mann, Sun Yi, and Muthuvel Murugan

NOTICES

Disclaimers

The views and conclusions contained in this report are those of the authors and should not be interpreted as representing the official policies or positions, either expressed or implied, of the participating U.S. Government DoD agencies or the U.S. Government. The U.S. Government is authorized to reproduce and distribute reprints for Government purposes not withstanding any copyright notation herein.

The findings in this report are not to be construed as an official Department of the Army position unless so designated by other authorized documents.

Citation of manufacturer's or trade names does not constitute an official endorsement or approval of the use thereof.

Destroy this report when it is no longer needed. Do not return it to the originator.

ARL-TR-10216 ● SEP 2025

Modeling and Optimal Adaptive Control of an Actuator Using Machine Learning Techniques with Long Short-Term Memory Integration

Richard McKee and Sun Yi
North Carolina Agricultural and Technical State University

D. Johann Djanal-Mann
Oak Ridge Associated Universities

Muthuvel Murugan
DEVCOM Army Research Laboratory

REPORT DOCUMENTATION PAGE

1. REPORT DATE	2. REPORT TYPE	3. DATES COVERED	
		START DATE	END DATE
September 2025	Technical Report	26 April 2022	31 December 2025

4. TITLE AND SUBTITLE
Modeling and Optimal Adaptive Control of an Actuator Using Machine Learning Techniques with Long Short-Term Memory Integration

5a. CONTRACT NUMBER	5b. GRANT NUMBER	5c. PROGRAM ELEMENT NUMBER
5d. PROJECT NUMBER	**5e. TASK NUMBER**	**5f. WORK UNIT NUMBER**

6. AUTHOR(S)
Richard McKee, D. Johann Djanal-Mann, Sun Yi, and Muthuvel Murugan

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
DEVCOM Army Research Laboratory
ATTN: FCDD-RLA-VA
Aberdeen Proving Ground, MD 21005

8. PERFORMING ORGANIZATION REPORT NUMBER
ARL-TR-10216

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)	10. SPONSOR/MONITOR'S ACRONYM(S)	11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT
DISTRIBUTION STATEMENT A. Approved for public release: distribution unlimited.

13. SUPPLEMENTARY NOTES
ORCIDs: Richard McKee, 0009-0003-5062-6503; D. Johann Djanal-Mann, 0009-0005-1374-7765; Sun Yi, 0000-0001-6375-3104; Muthuvel Murugan, 0000-0002-8314-3650

14. ABSTRACT
Adaptive control has been around for the past few years. Recent advancements in ML techniques have allowed these new techniques to be paired with traditional adaptive control to better track the controller output. A long short-term memory (LSTM) neural network has the unique ability to handle thousands of previous time steps, which makes it an ideal choice when using time-series data. There are several considerations for the design of a controller for an actuator, including nonlinearities such as hysteresis. The hysteresis is time dependent; therefore, the LSTM can capture the behaviors of the actuator with relative ease. By using experimental data, a neural network is trained and used as a controller to achieve robust control of an actuator.

15. SUBJECT TERMS
Mechanical Sciences, adaptive control, optimal control, AI/ML-based control, neural network-based control, model reference adaptive controller

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES
a. REPORT	b. ABSTRACT	C. THIS PAGE		
UNCLASSIFIED	UNCLASSIFIED	UNCLASSIFIED	UU	37

19a. NAME OF RESPONSIBLE PERSON	19b. PHONE NUMBER *(Include area code)*
Muthuvel Murugan	(520) 691-5892

STANDARD FORM 298 (REV. 5/2020)
Prescribed by ANSI Std. Z39.18

Contents

List of Figures

List of Tables

Acknowledgments

This research supports the Applied Research for the Advancement of Science and Technology Priorities program conducted by ARL in collaboration with the Naval Research Laboratory (NRL), Air Force Research Laboratory, and Missile Defense Agency under the sponsorship of the Office of the Under Secretary of Defense for Research and Engineering and Oak Ridge Associated Universities. We thank the following individuals: Jeff Fisher (NRL); Yan Borden and Dan Inman (University of Michigan); Byron Hall, Lowell Welburn (North Carolina Agricultural and Technical State University); and the support teams at MathWorks and Speedgoat GmbH and all project team members and their collaborators for support and valuable input on actuation concepts and control techniques suitable for the program.

1. Introduction

This research presents the design of a controller that attains robust control of an actuator using model reference adaptive control (MRAC) and a long short-term memory (LSTM) neural network (NN). The NN black-box model is trained using experimental data. It is then used to replace the existing plant model to achieve robust, real-time control of an actuator.

Traditional proportional–integral–derivative (PID) control methods have provided reliable control systems capabilities. A traditional PID control is susceptible to noise, delay, and disturbances; therefore, they struggle with highly nonlinear systems. One such nonlinearity that is inherent in piezoelectric actuators is hysteresis. Hysteresis is the phenomenon in which the value of a physical property output lags behind the input effect causing it (Adrianes et al. 2000).

The nonlinearities of the actuator create a unique challenge that has led to many attempts at developing advanced adaptive controls. The demand for advanced control systems with high-precision tracking accuracy has increased with progressing technology. Recent advancements with AI and ML techniques have paved the way for alternative designs for control systems other than a traditional PID controller (Chong et al. 2005). AI and ML techniques have become useful tools for obtaining these unique solutions.

There are many types of NNs. The characteristics of the dataset help decide which NN to use. Specifically, a bidirectional LSTM NN is used to train the model based on the experimental data. A traditional recurrent NN (RNN) struggles to handle more than 10 previous time steps (Patil et al. 2021). The complexity of the hysteresis nonlinearity requires many timesteps to be recalled for the NN to learn the behavior of the system. The LSTM is a RNN that can handle many previous time steps, making it an ideal choice for learning a time-dependent behavior such as hysteresis (Haughn et al. 2022).

Using a NN to replace a traditional PID controller is a unique approach because of the NN's unique ability to learn from experience in its environment (Li et al. 2020). The NN plant model predicts the system's dynamics, thus replacing complex mathematical models. The NN can approximate an unknown nonlinear function. It does so through regression analysis by fitting the relationship to a linear function seen in Equation 1. Weights and biases are assigned and then updated until the error converges to the minimum.

$$y = Wx + B \tag{1}$$

The outline of the work is as follows. In Section 2, we discuss the methodology for the different techniques used to analyze the performance of the NN controller. This includes data collection methods, the architecture of the NN, and the control system design. In Section 3, we discuss results for the Python and MATLAB approaches. Section 3 includes the plot of results, including the DC motor results, using both a traditional PID controller and an NN controller. These results will be quantified and compared using time domain analysis of risetime and overshoot.

2. Methodology

The aim of this project is to achieve robust control of an actuator using adaptive control and LSTM as a plant model. To achieve this, LSTM's ability to learn the behavior of an actuator must first be demonstrated. Next, using MATLAB and Simulink software, the model reference controller must have an ability to track the reference output accurately and precisely. Once the black box model has been demonstrated, the network parameters are adjusted to achieve an optimal response and disturbance is introduced to the system. The trained NN can then be used to achieve real-time control of the actuator.

2.1 Obtaining Training Data

The experimental data was first provided by the University of Michigan and contained several datasets for a piezoelectric actuator. The U.S. Army Combat Capabilities Development Command (DEVCOM) Army Research Laboratory (ARL) provided the datasets for the hydraulic actuator. These datasets contained various input profiles including sine wave, sawtooth wave, triangle wave, and square wave. Each profile was trained, tested, and compared by the accuracy of their predictions as well as their loss scores. The best datasets were selected to be used to train the NN. Both MATLAB and Python software were used to implement the NN.

The dataset provided included three columns: time in seconds, voltage applied in volts, and displacement in centimeters. A fourth column was added that converted the displacement in centimeters to millimeters. The Excel file was then saved as a .csv file so that it could be used in Python using the "readcsv" function. There was no missing data or not a number values; therefore, no further data processing was required other than the unit conversion and file type conversion.

2.2 Building NN

The basic architecture of an LSTM NN consists of three gates (Figure 1): input gate (It), output gate (Ot), and forget gate (Ft) (Yu et al. 2019). Each gate has a sigmoid activation function (σ). An additional candidate gate (Ct) decides the importance of past information. Information is either stored or discarded through the Ft based on how that information influenced the response during previous time steps. In this way, the network can learn the behavior of the system over time.

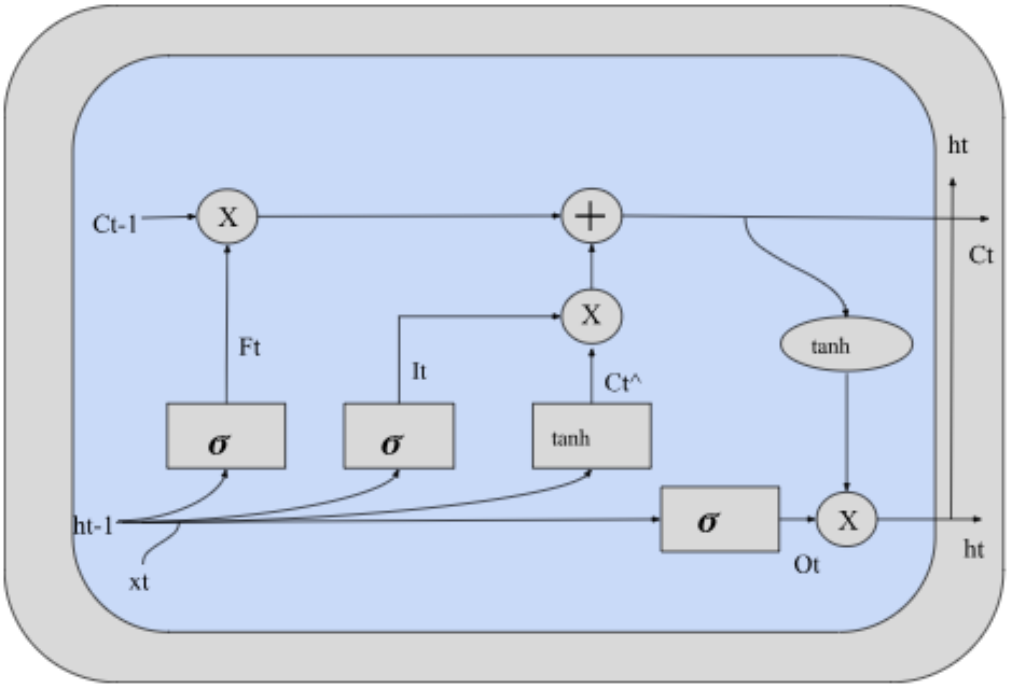

Figure 1. Architecture of LSTM.

The first LSTM NN created for this project was built in Python using the sequential function, which allows the layers of the network to be added manually and in sequential order. The Keras and Pytorch packages are used and provide many necessary built-in functions that make constructing an NN from scratch less difficult. A bidirectional LSTM allows for data to be shared forward and backward through the various layers of the network (Chen et al. 2007). The code is designed to be interactive with the user and prompts menus allowing the user to customize the parameters of the network and choose which datasets to use.

Experimental data provided by the University of Michigan team is used as the training data for the LSTM. The University of Michigan used a piezoelectric actuator. Each input profile had its own unique output and hysteresis curve (Figure 2). There are several factors that influence the shape of the hysteresis curve, and the most influential is the input frequency. The various amplitudes and

frequencies were used to increase the richness of the datasets, which is an important step in obtaining a useful training dataset.

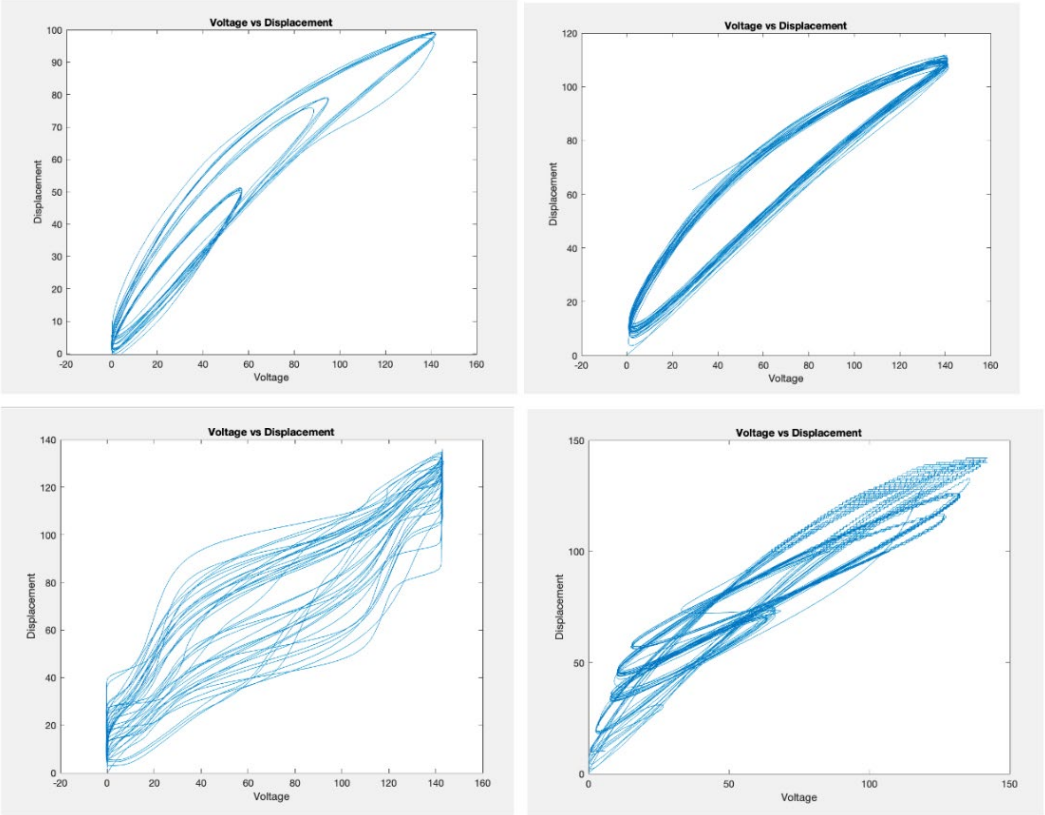

Figure 2. Hysteresis curves of different input profiles.

Once the approach has been demonstrated using Python, the model is replicated using MATLAB and Simulink. This was done to save computational time and for ease of implementation to actuators' hardware. There were several additional datasets provided by ARL (hydraulic actuator) used within Simulink, each with a different profile (Borden et al. 2023). The richness of the dataset (how much variation) can contribute to a better training result and is encouraged. The input profiles provided included sine, sawtooth, square, and triangle inputs as the desired reference input.

A significant contribution to this project that led to better training results was combining all the different input profiles into one large dataset. This data was used to train the plant model because it provided a rich dataset that covered several input profiles and their responses. Essentially, the network was better prepared for random data because it experienced more variation in training than the single profile was able to provide. The combined training dataset (shown in Figures 3 and 4) led to better tracking and more accurate results than each of the individual input profiles.

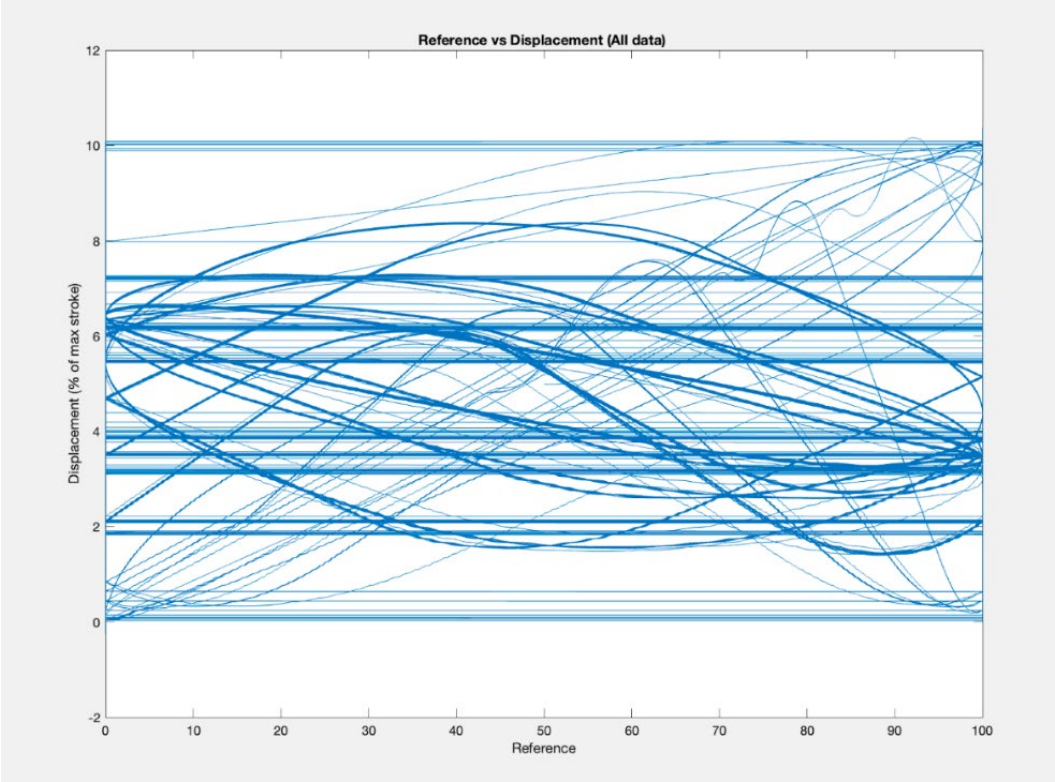

Figure 3. Input profile of the combined datasets.

The complexity of the NN has a profound effect on the results of the LSTM. Increasing the number of hidden layers and changing the number of epochs adds complexity to the network. Increasing the number of sequences requires more computational cost but allows the behavior of the system to be understood over longer periods of time. When comparing Python with MATLAB, we found that MATLAB handles larger datasets better than Python and generally has less computational time.

Overfitting can be a problem when working with an LSTM NN. One way to handle this is by adding a dropout layer (Cong 2009). The dropout layer removes a small percentage of the input and hidden layer nodes to prevent overfitting. A dropout rate of 0.2 is used throughout this project.

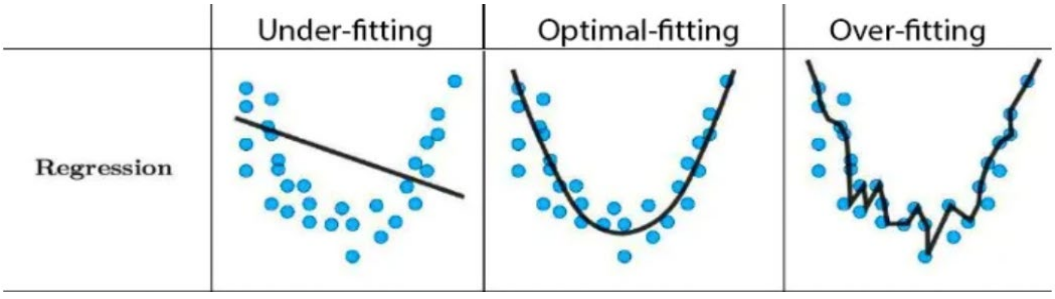

Figure 4. Different fits for regression.

2.3 Controller Design

2.3.1 Mathematical Model

To complete the MRAC, a reference model is required. An exact plant model is not available, and a simple model usually leads to a simple controller that is easy to use (Ioannou et al. 2013). This leads to more practical uses for the controller. For this project, a mass spring damper model is used for the reference model (Equation 2). The Simulink model representing this system is shown in Figure 5. The voltage input, u, has a displacement output, y. The original parameters for mass, spring constant, and damper are chosen by their ability to stabilize the system (Equation 3). Later, they are tweaked to tune the response.

$$m\frac{d^2x}{dt^2}(t) + c\frac{dx}{dt}(t) + kx(t) = f(x) \tag{2}$$

$$T.F. = \frac{output}{\text{input}} = \frac{9}{s^2+6s+9} \tag{3}$$

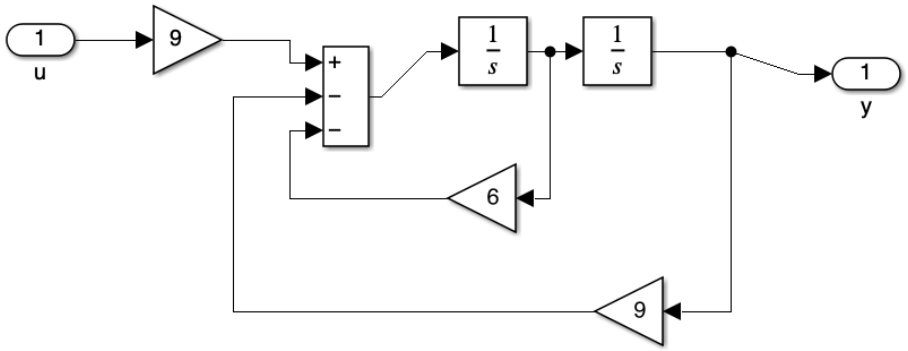

Figure 5. Simulink model used for the reference model.

2.3.2 MRAC Simulink

The MRAC is used in Simulink (Figure 6), and the default structure is modified to accommodate this project.

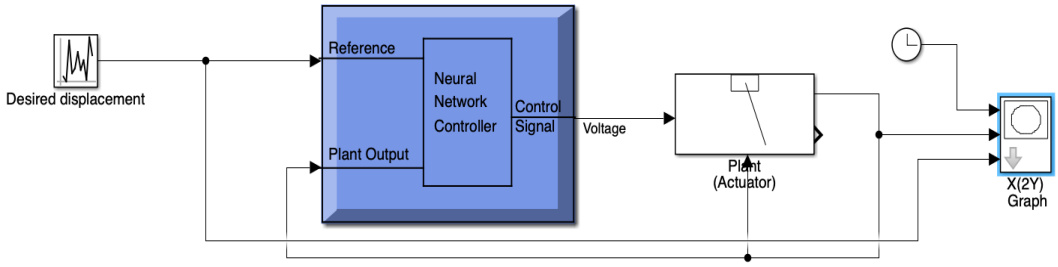

Figure 6. MRAC.

The MRAC has an interactive menu that makes customizing this model simple and efficient. The number of inputs, outputs, and hidden layers can be selected along with the number of samples, epochs, and training segments. The "use cumulative training" box is checked so that the network can learn from previous training segments. This feature is very useful for finding the optimal controls for the network weights quickly. If an initial result was bad, the "use current weights" box is unchecked, so that those weights are not selected. The model generates new weights based on avoiding weights that do not work and using cumulative training to find weights that work (Figure 7).

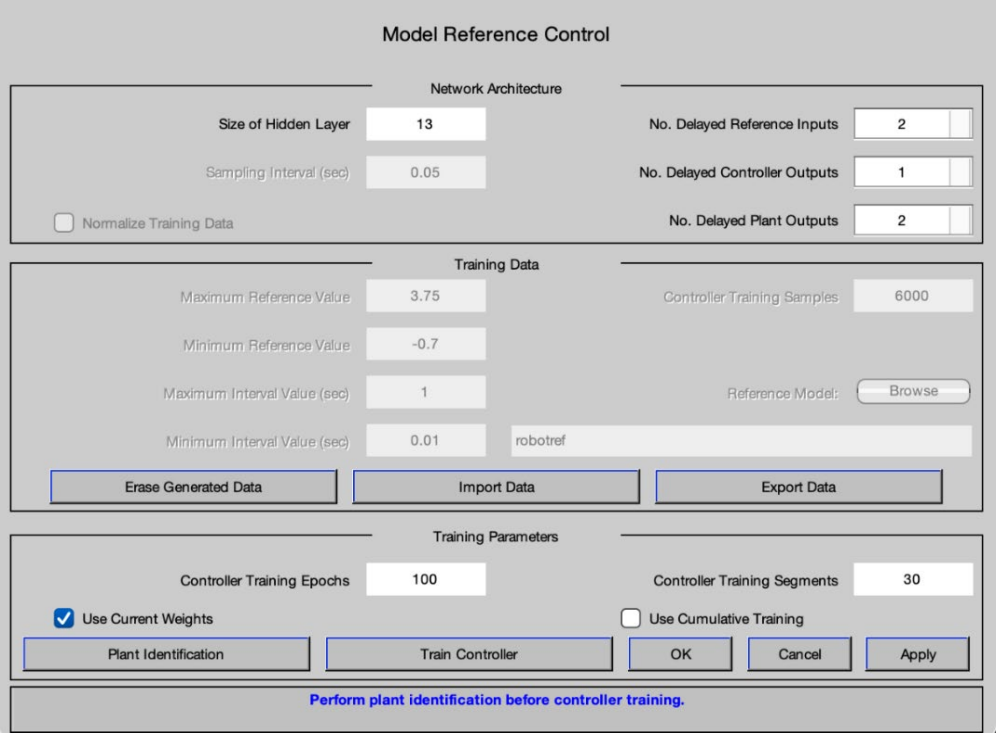

Figure 7. MRAC user interface.

2.3.3 Disturbance Rejection

Once a model has been well-trained, disturbance is introduced to the model. A sine wave block is used in Simulink to simulate the disturbance. The maximum amplitude of the sine wave is between 5% and 10% of the maximum actuator stroke value. The MRAC adapts to the disturbance signal via a feedback loop. Disturbance can cause issues with control systems, and they occur in every real-world scenario. Handling the disturbances within a signal is essential for obtaining a robust control signal.

3. Results

3.1 Piezoelectric Actuator Experimental Data

Table 1 shows that the best accuracy achieved was only 83% given a dataset limited in sample size. Obtaining useful training data requires a large sample size. More experiments were run and provided much larger datasets to use for training. Each of these datasets were used in Python to train the bidirectional LSTM model.

Table 1. Initial Python results using dataset provided by University of Michigan.

Neural Network	Input	Output	Size	Accuracy
NN1	PK4GA7Px data (Voltage)	PK4GA7Px data (Increasing and Decreasing Displacement)	31 x 1	83.141%
NN2	PK4GA7Px interpolated (Voltage)	PK4GA7Px interpolated data (Increasing and Decreasing Displacement)	162 x 1	82.343%
NN3	Randomly generated array	Data from Bouc Wen model output using randomly generated array as input	280 x 1	80.785%

3.1.1 Python Initial Results

Although the predicted displacement did not achieve the correct amplitude, the tracking behavior was accurate. Once the network displayed the correct predicted profile, the parameters were adjusted to achieve even more accurate results (Figures 8–11). Specifically, the sequence length, number of epochs, and number of hidden layers were all increased.

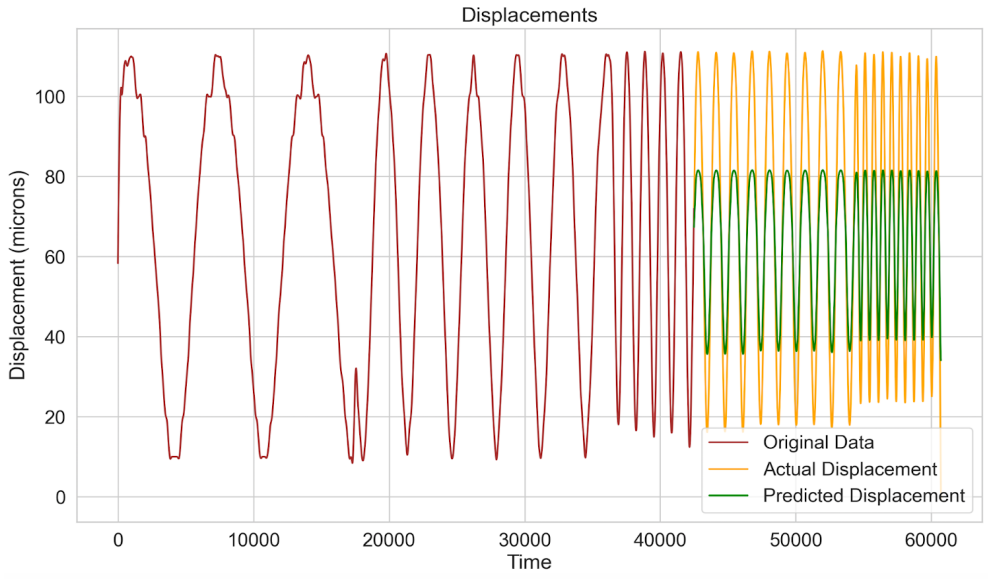

Figure 8. Initial results of the Python sine wave.

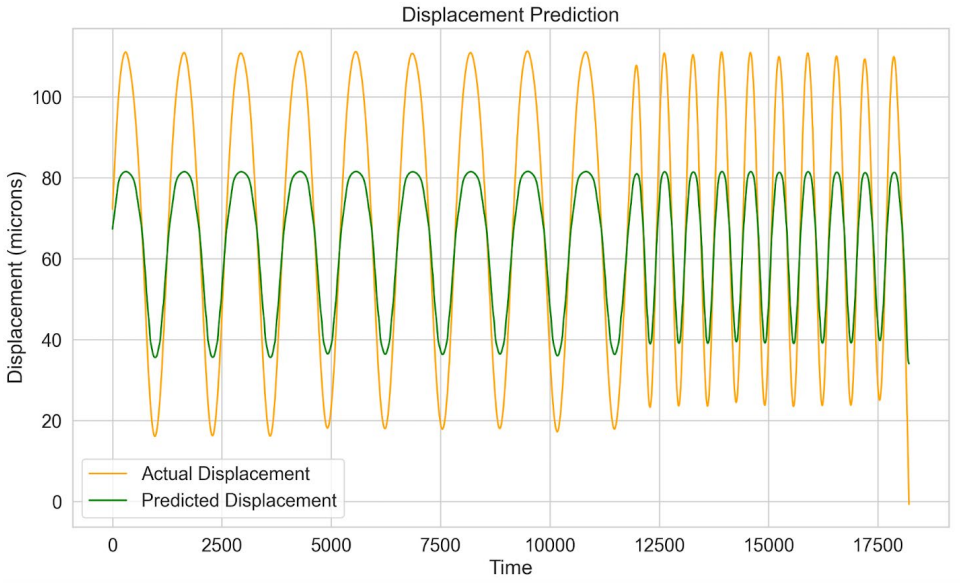

Figure 9. Initial results of the Python predictions' sine wave.

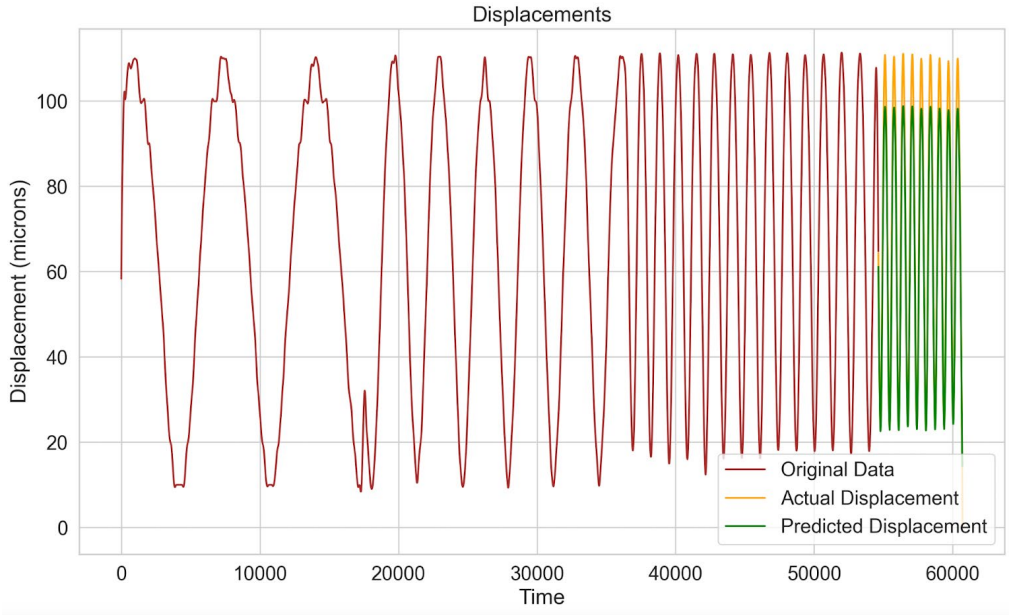

Figure 10. Improved results from updating parameters in Python.

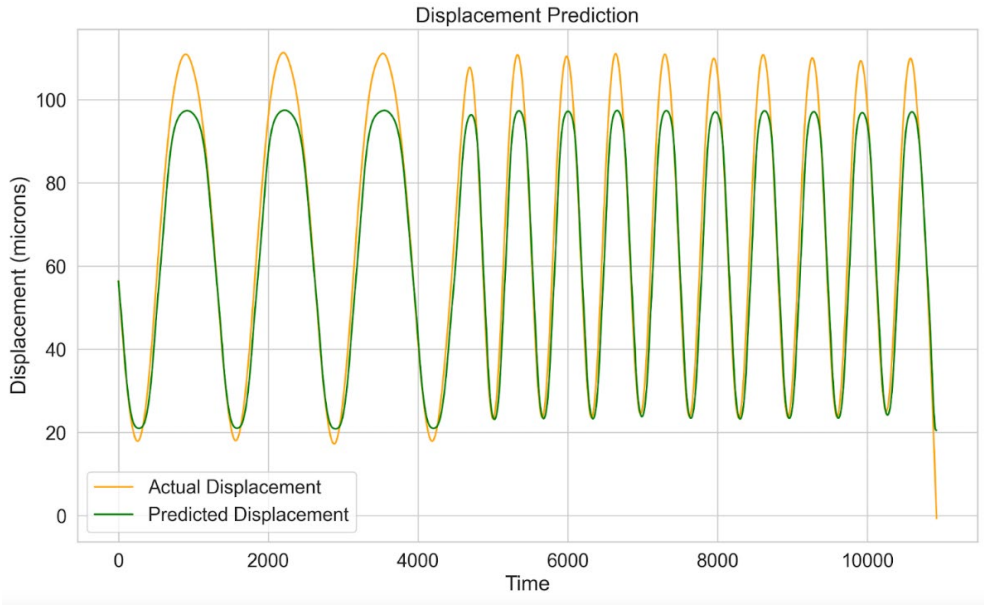

Figure 11. Improved results of Python predictions.

There is some overshoot on the lower half of each wave pattern, but the maximum amplitude is not reached. To help fix this discrepancy, more richness was needed with the dataset profiles. To add richness to the dataset, experiments were run and produced training data that had various amplitudes and frequencies within the same dataset. The network was then trained using a rich dataset, leading to the most accurate results using Python.

3.1.2 Python Results

Table 2 highlights two of the datasets that were considered to have the best results. Although the "Plate pulse wave 1 Hz" and "Sine wave var amp" datasets had a lower loss score, they did not test as well, and the predictions were not as good as the "Plate pulse 1 Hz diff amps" and "Newdata2" datasets. This is because the data used for testing is different from the data used for training, even though they originated from the same dataset. The robustness of both "Plate pulse 1 Hz diff amps" and "Newdata2" datasets were the reason these datasets were considered to have performed the best. Both datasets had varying amplitudes and frequencies to improve the richness of the datasets.

Table 2. Selecting the best datasets based on richness, not only loss score.

Dataset	Test/Train split	Number of Sequences	Epochs	Best Loss Score
Plate Pulse wave 1 Hz	80/20	25	50	0.00358
Plate Pulse wave 1 Hz diff amps	80/20	25	50	0.00353
Sinewave var amp	80/20	25	50	0.00507
Newdata1	70/30	25	50	0.006700
Newdata2	70/30	25	50	0.000607
Newdata2	90/10	25	50	0.000696

Once the best datasets had been selected, the best parameters were chosen, and the complexity of the network was increased by adding 16 hidden layers. The predictions were more accurate than the original results, and the LSTM has proven to be a successful approach to training based on the time-series data (Figures 12 and 13).

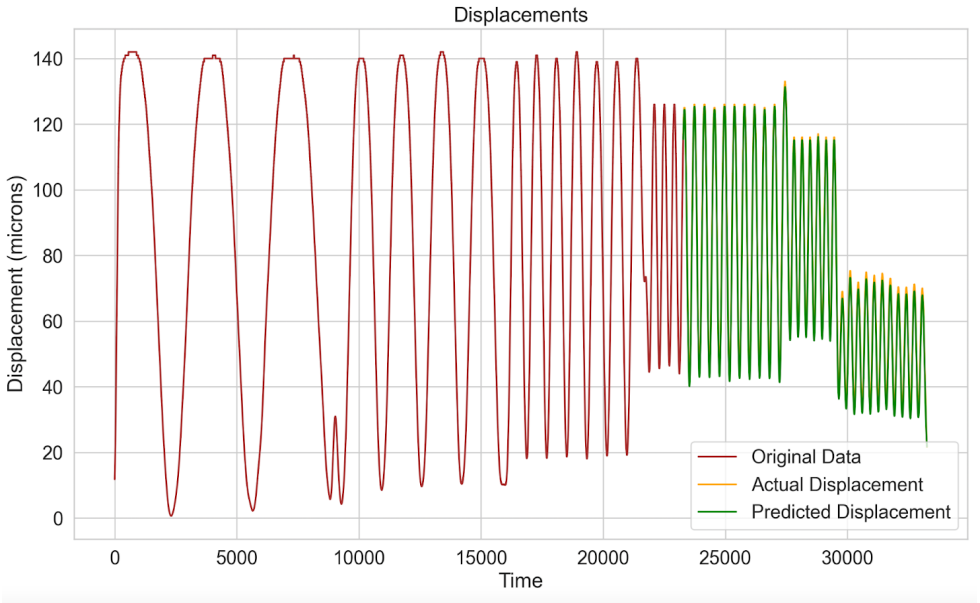

Figure 12. Final results in Python.

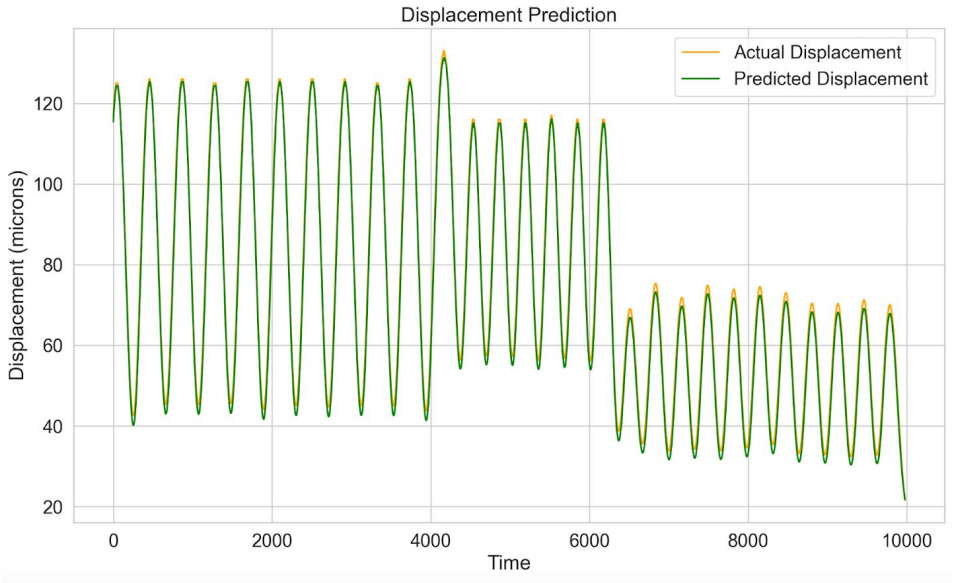

Figure 13. Final results from Python predictions.

The optimization algorithm chosen for the Python LSTM is the Adam (Adaptive Moment Estimation) optimizer. The Adam optimizer allows a faster, smoother convergence (Bock et al. 2019). In addition, the Adam optimizer combines techniques from the root mean squared propagation (RMSProp) and the momentum, which accelerates the gradient descent. The update rule with momentum is displayed in Equations 4 and 5.

$$w_{t+1} = w_t - \alpha m_t \tag{4}$$

$$m_t = \beta_1 m_{t-1} + (1 - \beta_1)\frac{\delta L}{\delta w_t} \tag{5}$$

Figure 14 demonstrates how the training loss converged towards the minimum in less than 20 epochs, and the testing loss continued to decrease in the same number of epochs. This demonstrates the network learning the behavior of the system based on the data to which it is introduced. A dropout layer prevents overfitting, ensuring the validity of the results.

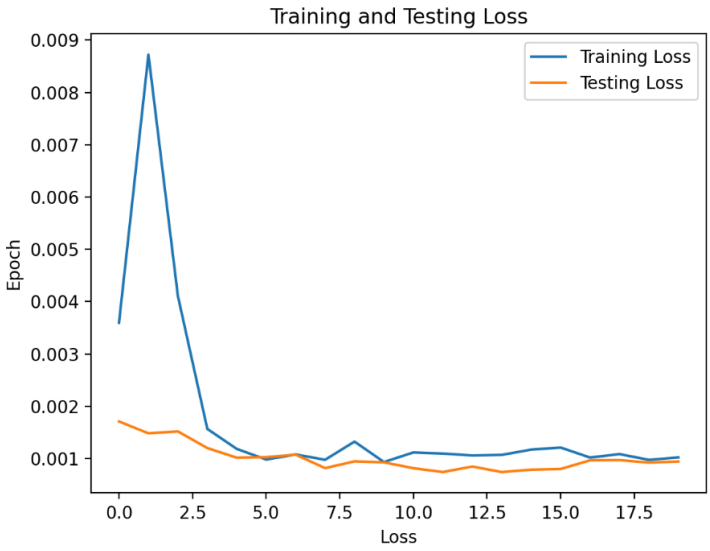

Figure 14. Trainng and testing loss in Python.

3.2 Hydraulic Actuator Experimental Data

The Python approach worked well for the initial datasets but struggled with large datasets of over 1 million samples, so MATLAB's Deep Network Designer was used to recreate the LSTM in MATLAB. The Deep Network Designer excels in processing large datasets quickly and efficiently. After obtaining several new datasets provided by ARL, MATLAB was used to train the plant model and build the model reference controller. The datasets provided contained a sine, sawtooth, triangle, and square wave input profile as well as their corresponding output displacements. The reference signal was a value between 0 and 100 and represented the percentage of maximum stroke that the actuator displaced.

3.2.1 MATLAB Deep Learning Network Designer Simulink

MATLAB's Deep Learning Toolbox has a deep network designer that is user friendly and allows the layers of a network to be constructed simply by clicking and dragging corresponding blocks into the desired location within the network. A prebuilt LSTM model was selected and then modified for this project. The prebuilt version was designed for classification, whereas this project needs an LSTM for regression purposes. To change this, the "SoftMax" and "classification output" layers are removed, and a "regression output" block replaces the output block. The LSTM is no longer a classification LSTM but a regression LSTM (Figure 15).

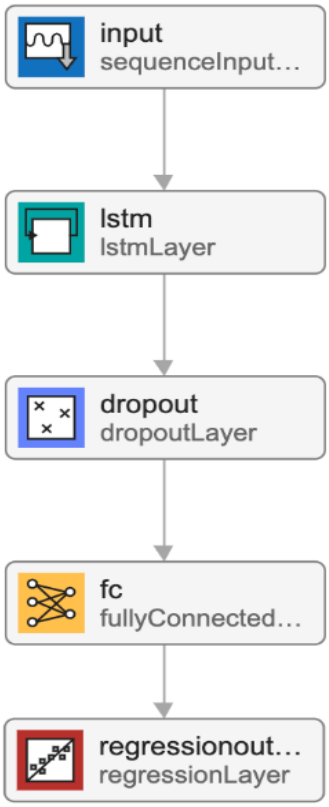

Figure 15. Layers of LSTM using MATLAB's deep network designer.

Once the network structure was built and parameters customized, the network was trained using experimental data. The Deep Learning Network Designer gives the user the choice to output the trained network as a script or a Simulink model. The Simulink model is selected so that it can be directly implemented with the MRAC model.

3.2.2 MATLAB Deep Network Design Training Results

The experimental data is split into training, testing, and validation datasets. The results show excellent tracking behavior. The NN output can recognize subtle differences in amplitudes (Figure 16).

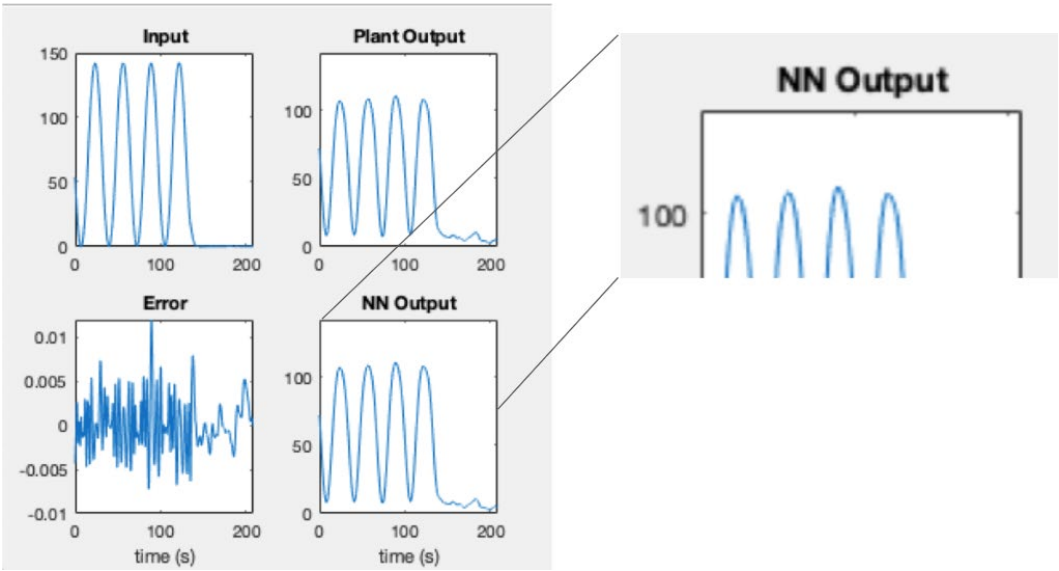

Figure 16. NN's ability to notice subtle differences in amplitude.

After each epoch, the parameters are updated, and the mean squared error loss function is minimized during convergence. Figure 17 illustrates how the error decreases over time, eventually settling at the minimum. There are several different algorithms that can be used, as listed in Table 3. Each one uses a slightly different approach for finding the minimum. For this project, a faster convergence is desired; therefore, an algorithm (traingdx) with an adaptive learning rate and momentum is used. The adaptive learning rate allows the speed of convergence to accelerate if it is headed towards the minimum (Figures 18–20).

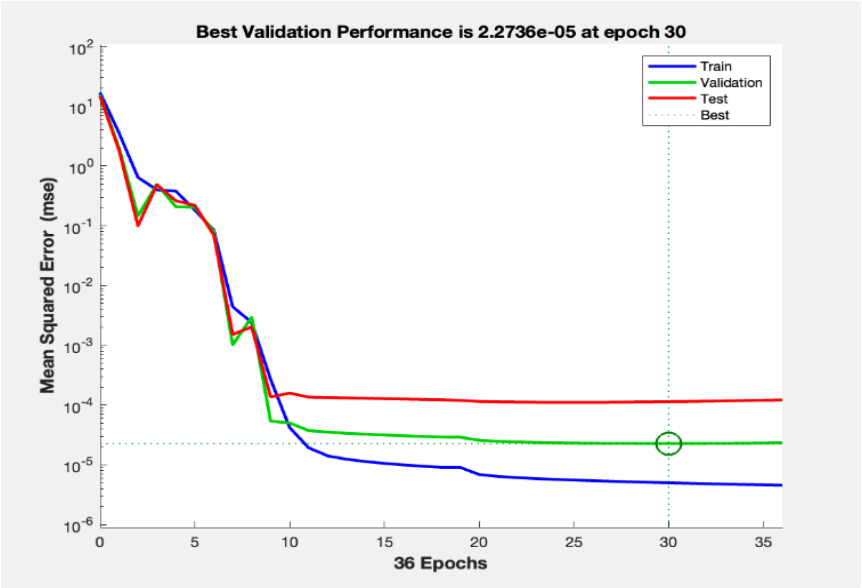

Figure 17. Convergence of MATLAB LSTM.

Table 3. Optimization algorithm options for MRAC.

Algorithm	Brief Description	Additional information
trainbfg	2nd order, Quasi-Newton	Numerical optimization
trainbr	Uses Levenberg-Marquardt to updates weights and biases	Minimizes squared error and weights
trainlm	Fastest backpropagation in toolbox	Can sacrifice accuracy for speed
traincgb	Updates weights and biases with Powell-Beale restraints	Large scale unconstrained optimization
traincgf	Conjugate gradient descent with Fletcher-Reeves updates	Functions must have derivatives
traingdm	Gradient Descent with momentum	Pushes the cost further around a saddle point
traingda	Gradient Descent with Adaptive learning rate	1st order, local minimum may not be global minimum
traingdx	Gradient Descent with momentum and adaptive learning rate	Used for this project
trainoss	One-step secant backpropagation	Bridges gap between conjugate and quasi-Newton
trainrp	Resilient backpropagation	1st order, feedforward artificial neural network
trainsg	Scaled Conjugate gradient	Computationally expensive, no line search at each iteration

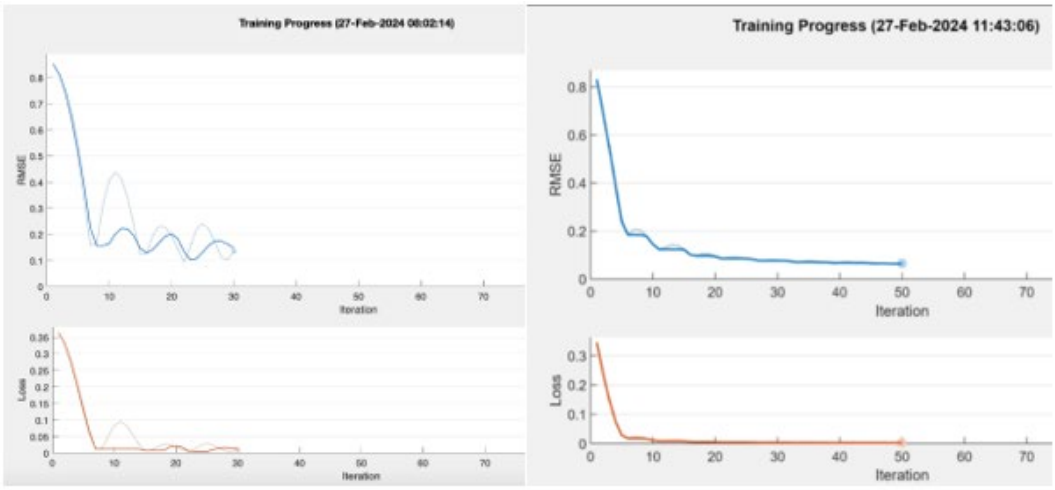

Figure 18. Comparing the convergence of the standard gradient descent (left panel) with the Adam optimizer (right panel).

By combining elements from both the momentum and RMSProp techniques, the Adam optimizer has a few advantages over other algorithms. By using bias correction, it can prevent early age instability. The learning rate adapts based on weights and this prevents oscillations (Figure 18). By adjusting fewer hyperparameters, the Adam optimizer allows a more cost-efficient, accurate result.

3.2.3 MATLAB MRAC Results

A randomly generated input profile that represents the desired displacement is created and used as the reference model input (Figure 19). The NN controller output (green) is compared with the reference model output (blue) (Figure 20).

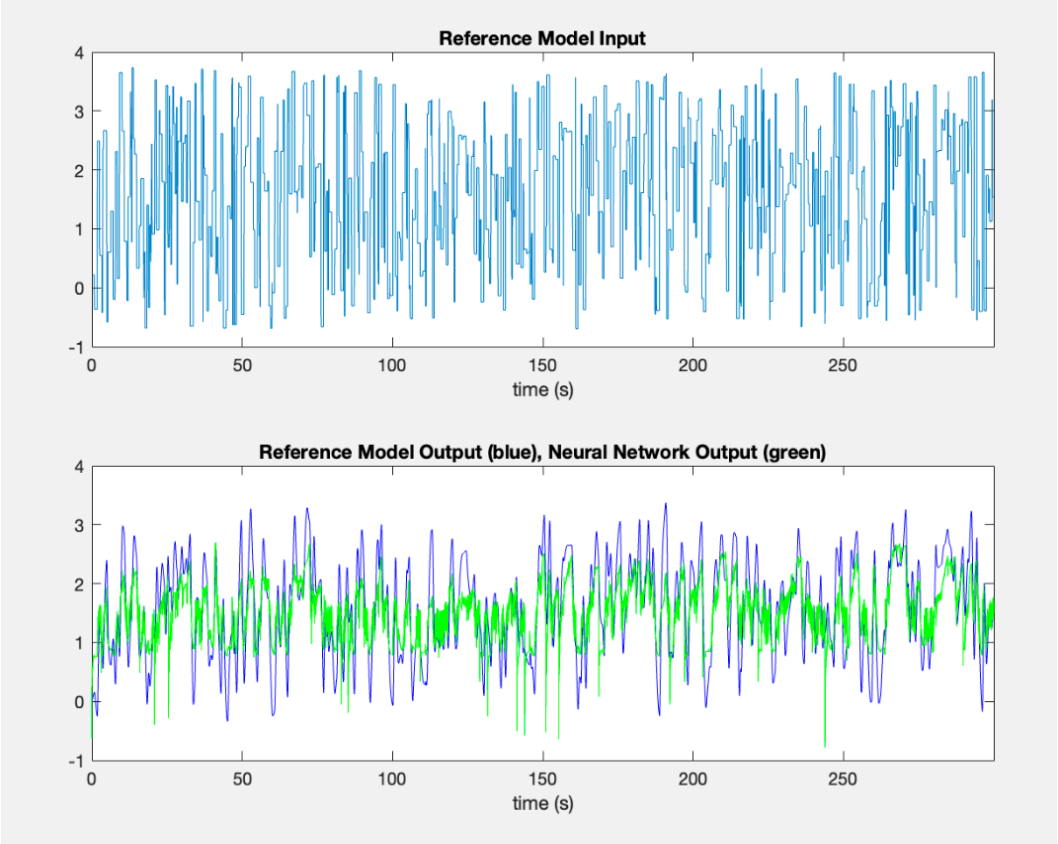

Figure 19. Initial MATLAB results showing small undershoots.

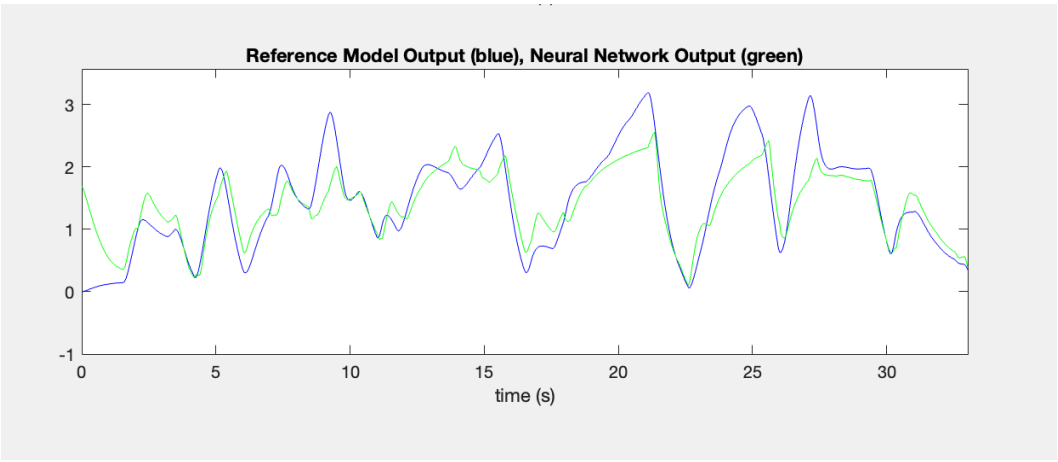

Figure 20. Initial MATLAB results showing good tracking behavior.

These results show the tracking behavior of the NN controller. However, there are small undershoots that need to be corrected (Figure 21). After updating the parameters, the accuracy of the response improved (Figure 22).

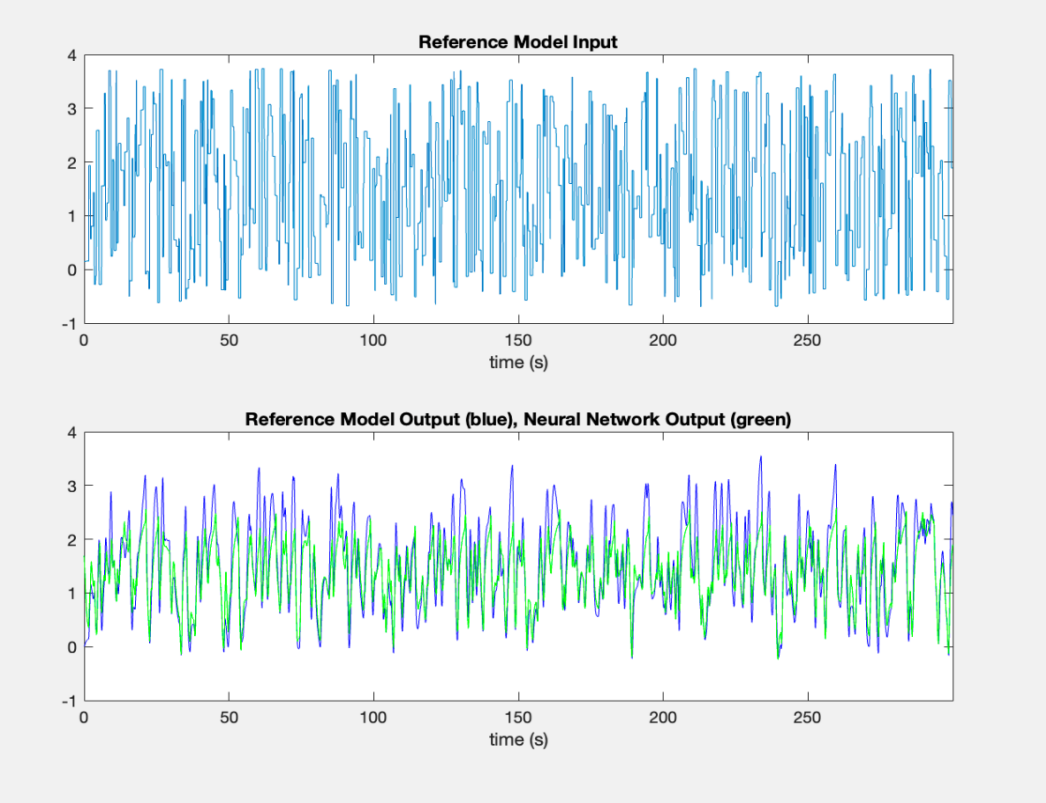

Figure 21. Updated MATLAB results after parameter update to correct undershoots.

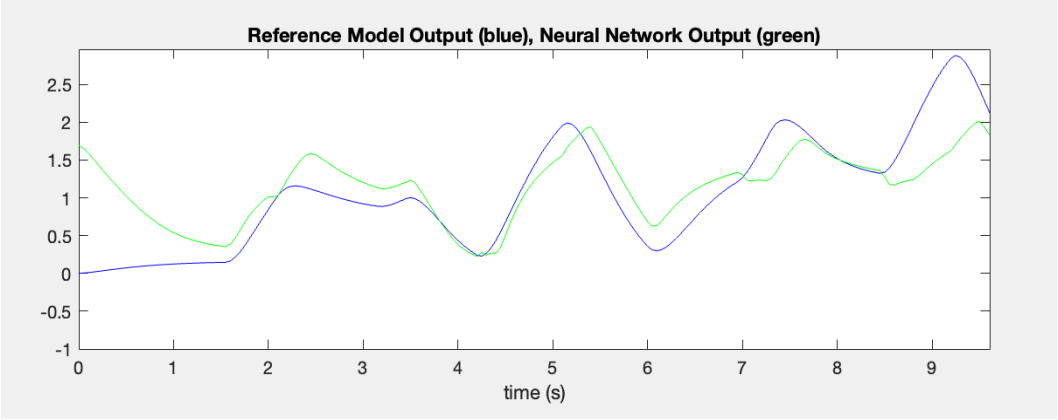

Figure 22. Results after correcting undershoots with better tracking behavior.

A sine wave block is added to introduce disturbance. The "cumulative training" box is checked within the GUI of the MRAC. This allows the network to build upon previous training to learn the behavior of the system. By doing so, the controller can accurately predict the output. Figure 23 shows that the reference model

produced a sine wave output. The NN output normally attempts to follow the reference model output; however, in this instance, it predicted that the input profile was a square wave, which proved to be the case. The NN controller outperformed the reference model in this situation, and, in fact, it led to the most accurate results with nearly perfect tracking (Figures 24 and 25).

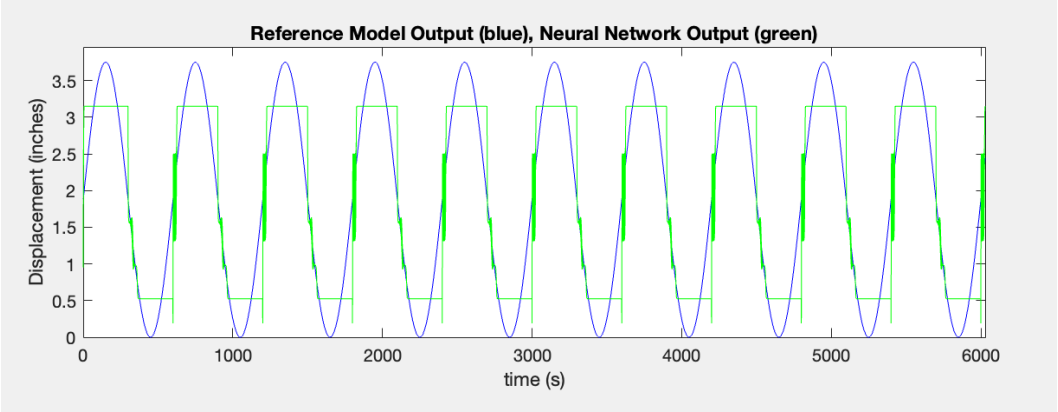

Figure 23. Given a square wave input, the reference model response is a sine wave, but the NN controller was able to learn that the input profile was a square wave.

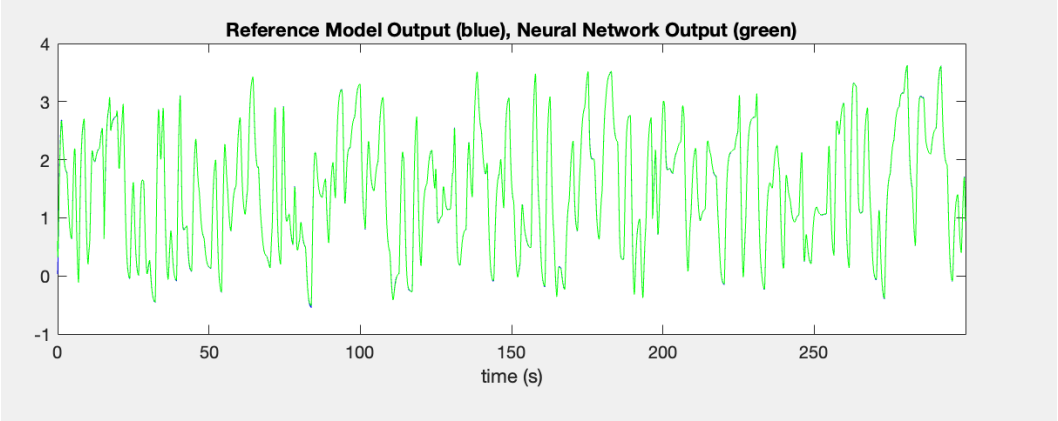

Figure 24. Perfect tracking ability after training the MRAC.

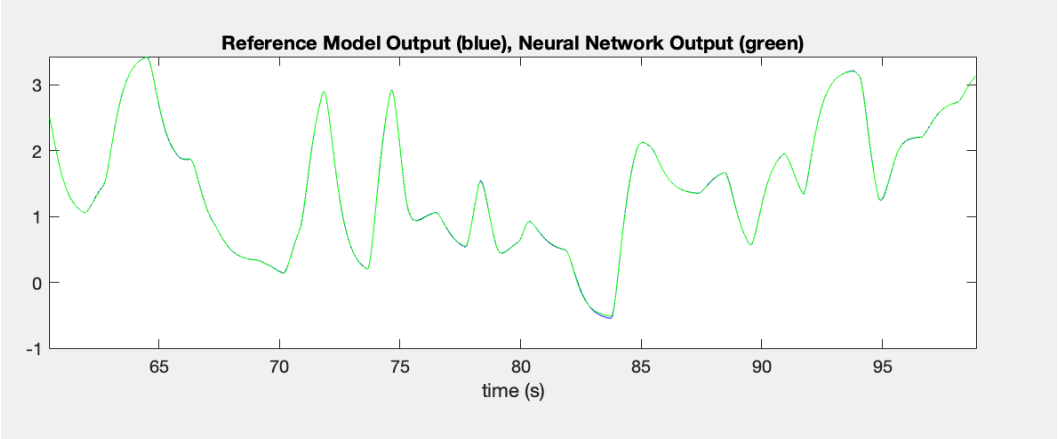

Figure 25. Magnified view of the perfect tracking ability of the NN controller.

3.2.4 Implementation on DC Motor

The Quanser SRV-02 DC motor is used to test the implementation of the NN-based controller. The original Simulink model was used for the baseline, and the original PID controller is replaced with the trained NN model. The SRV02 DC motor used a first-order transfer function, which led to a unique challenge. The large spikes in voltage shown in Figure 26 have infinite slopes, which led to a poor result. Using a second-order transfer function (Figure 27), this issue is mitigated.

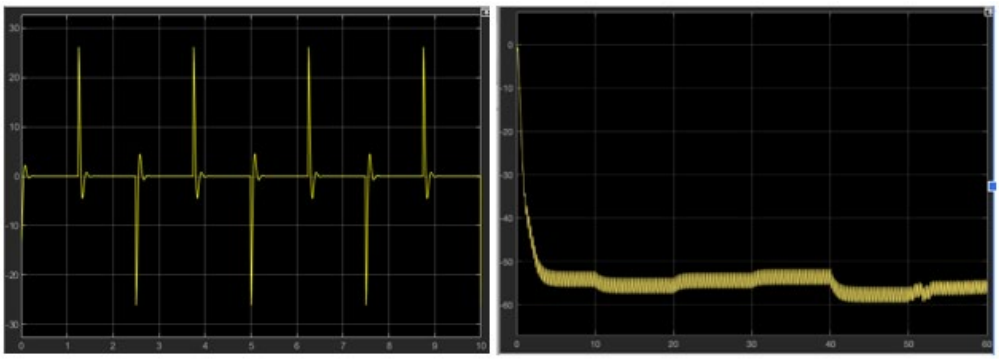

Figure 26. Large spikes in voltage (left panel) have an infinite slope, resulting in problems when differentiating (right panel).

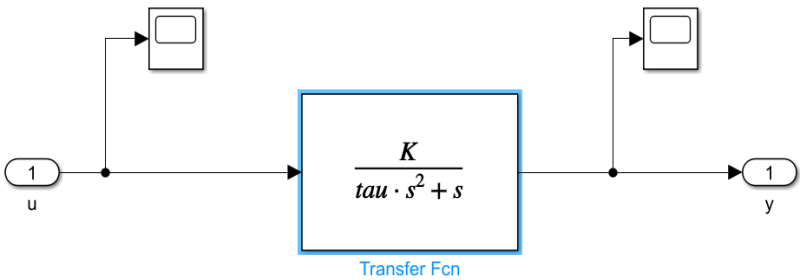

Figure 27. Second-order transfer function used for the SRV02 DC motor.

The original PID controller is replaced with the NN controller (Figure 28). The initial results were not good due to a sampling rate discrepancy between the trained model and the SRV02 model. The network had to be retrained at the correct sample rate (0.01 s) used in the SRV02 model. Once this issue was resolved, the NN controller was implemented with the DC motor. The results with and without disturbance are shown in Figures 29 and 30, respectively.

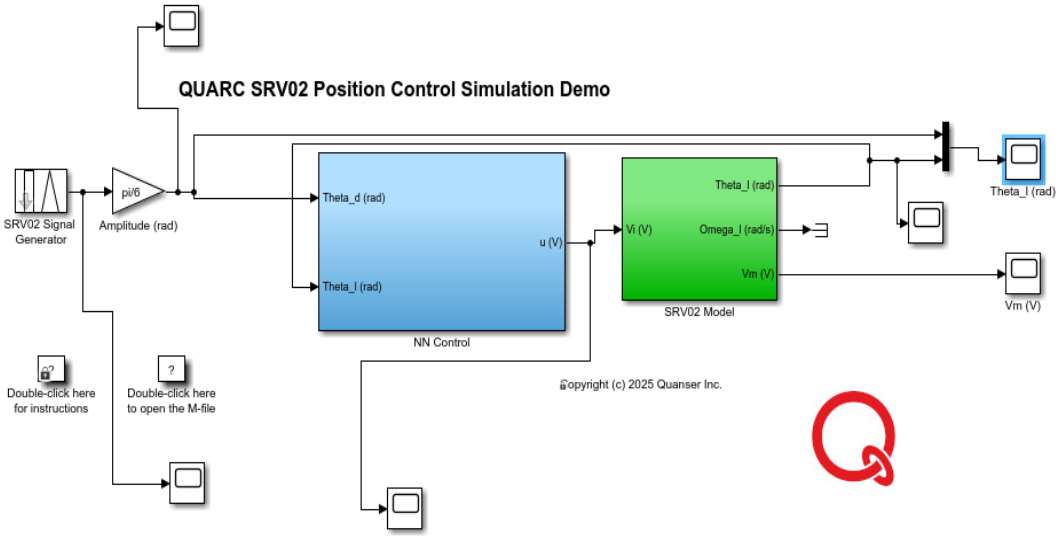

Figure 28. Simulink/QUARC model for Quanser SRV02 DC motor: the PID controller has been replaced with the NN controller.

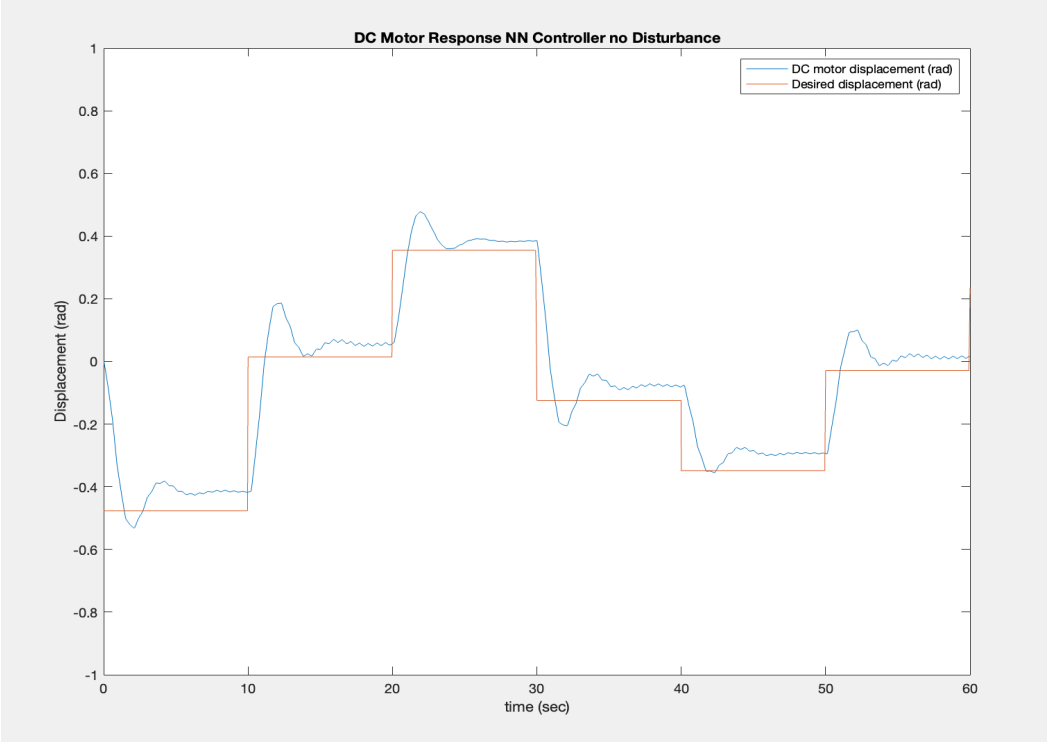

Figure 29. DC motor response using the NN controller without added disturbance.

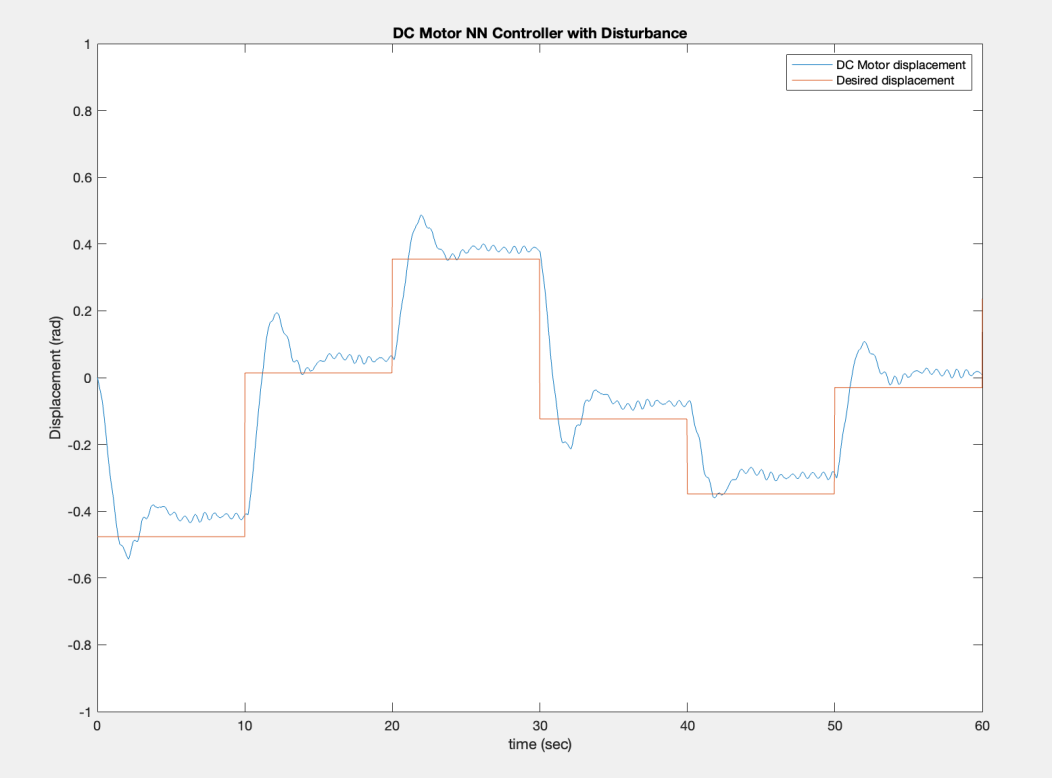

Figure 30. DC motor response using the NN controller with added disturbance.

The reference model parameters were then adjusted, and a limiter was added to the QUARC model (Figure 31). The limiter prevented the overshoot error.

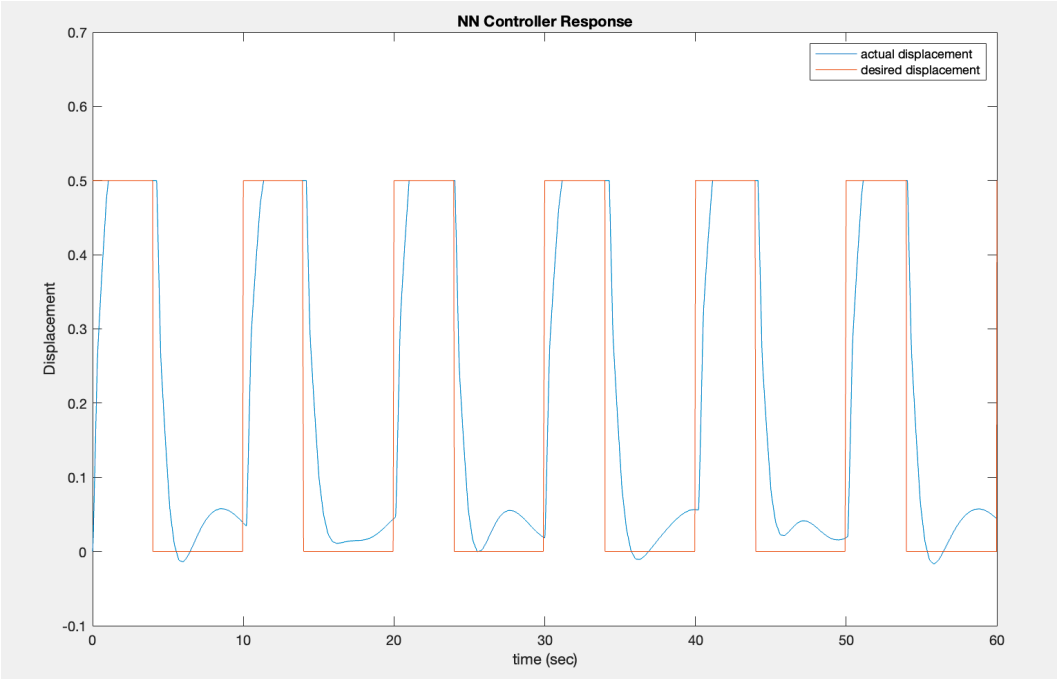

Figure 31. DC motor response using the NN controller with the added limiter.

Next, the results are compared with those of a PID controller. The PID controller is highly dependent on selecting the correct gains (Figure 32). The NN controller showed excellent ability to adapt and respond to external forces, and it was more consistent due to its independence to gains (Figure 33). The NN controller is dependent on the frequency of the input (Figure 34).

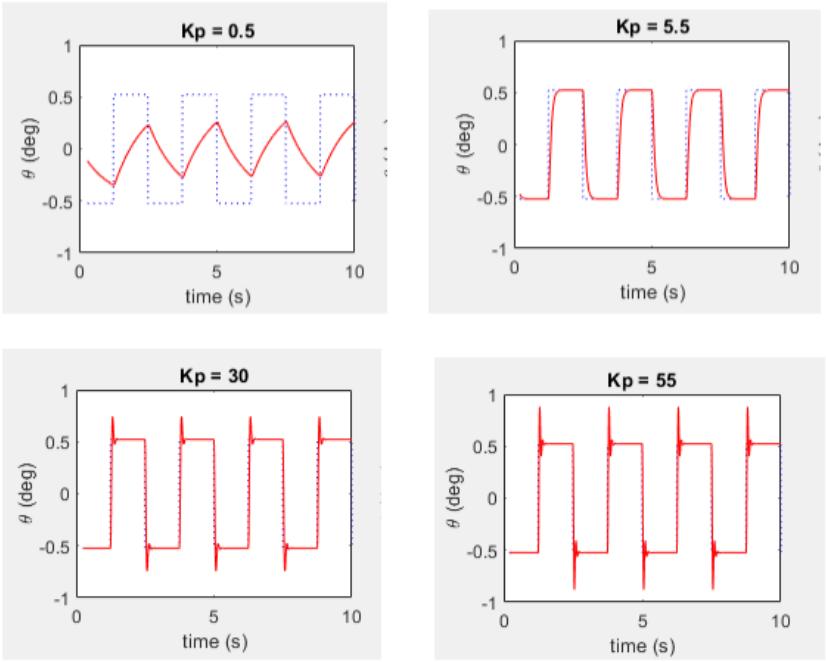

Figure 32. PID controller response showing dependence on selecting correct gains.

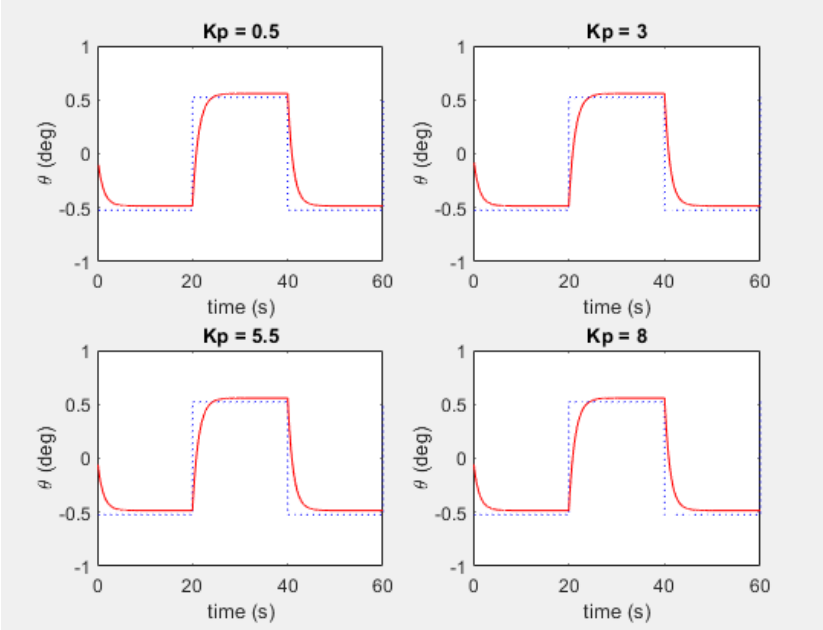

Figure 33. NN controller response showing the ability to prevent overshoot and respond quickly.

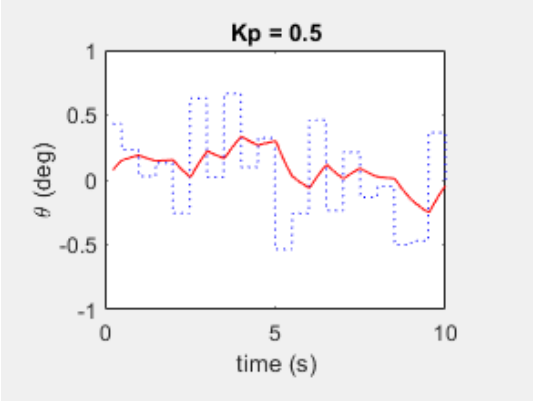

Figure 34. NN controller response to high-frequency, random data.

The risetime is the time it takes for the controller to respond and reach a certain percentage of its maximum rotation. Both the 63% rule and the 10%–90% risetime are considered. The maximum displacement was ±0.5 radians, or a rotation of approximately ±28°. The time it takes for the response to go from 0% to 63% maximum rotation is considered the risetime for the 63% approach. The values for 63% maximum rotation are 0.315 radians, or 18°. For the 10%–90% approach, the time it takes to go from 0.05 to 0.45 radians or from 2.86° to 25.78° is considered the risetime. Table 4 shows the different risetimes for different frequencies. The risetime for the 63% approach is higher because it starts from a 0% rotational state (initial "rest" position).

Table 4. Risetime for various frequencies for the NN controller.

Frequency in Hz	0%–63% risetime in seconds	10%–90% risetime in seconds
1.155	0.865645	0.505198
1.181	0.845774	0.409384
1.320	0.757439	0.304861
1.383	0.722816	0.252599
1.431	0.698828	0.165496
1.693	0.590622	0.121944

4. Conclusions and Future Work

The LSTM demonstrated a unique ability to train a controller to achieve a fast and accurate response. The LSTM was effective in both Python and MATLAB. As the size of the dataset and the complexity of the network increased, the computational time greatly increased in Python, making MATLAB a more time efficient option to use.

The MRAC within Simulink provided a useful tool for training the network quickly, but careful consideration to detail was necessary to effectively train the model.

There are many parameters, variables, and hyperparameters that need to be adjusted, and each one must be done correctly to achieve an accurate training result that can be considered reliable. By using the "cumulative training" option, the MRAC was able to learn how to achieve better results over multiple training cycles.

The richness of the datasets had the largest influence on the accuracy of the results. The number of samples clearly played a role in determining the effectiveness of the training. The experiment for generating training data was run for 60 s. A dataset containing hundreds of thousands of samples and a sample rate of 0.01 s were sufficient for training the model accurately. Varying the amplitude of the signal during training provided the information needed for the network to handle a random reference signal.

Allowing the use of an adaptive learning rate by use of the Adam optimizer algorithm allowed a faster convergence than a standard gradient descent algorithm approach. Sometimes a faster learning rate results in some instability, but this was not the case for the Adam optimizer due to its biased correction ability. There are many other optimization algorithms available, but the Adam optimizer proves to be a reliable option for the fastest, most accurate response when using a bidirectional LSTM for a time-series-based dataset.

The complexity of the NN was the final factor that led to a nearly perfect training result. The complexity of the training data was increased by combining each input profile into one large dataset. Hidden layers were added simply by increasing the number of hidden layers in the MRAC's interactive menu. Achieving good training results was a result of careful selection and tuning of parameters, choosing a large, rich dataset, and adding the correct amount of complexity.

The PID controller results were dependent on selecting the optimal control gains. Large overshoots were often present and would not be desirable in a real-world scenario, potentially leading not only to a loss of accuracy but also to damaged equipment. The NN controller prevented overshoot and had a consistent response. The NN was dependent on selecting optimal training data and parameters. The frequency used for training and the frequency of the input used in testing was a deciding factor in the response of the NN controller.

Now that the effectiveness of the LSTM NN controller is demonstrated, there are more opportunities to enhance and improve on this idea. Each application will have its own unique dataset. This requires that each network to be designed to complement the uniqueness of the data. By changing the number of hidden layers, choosing different optimization algorithms, and adjusting the parameters of the network, optimal results can be found and applied to a particular system.

As additional experiments are conducted, more data will be available, creating more opportunities to make improvements to training results. One issue that came up during the implementation of the controller on an actual motor was the sample rate. The sample rate for training must match the sample rate used for implementation. When working with many digital devices, this can potentially be a difficult problem to manage.

Other areas that may require more research include computational cost. The large datasets and high sample rates needed for training and implementation require a healthy amount of CPU power. Parallel computing as well as quantum computing offer potential solutions to the computational cost problems. Recent advancements in these fields have allowed these approaches to be more easily implemented.

This work has demonstrated the accuracy that a trained NN can offer. Through careful selection and tuning of parameters and hyperparameters of the network, a high level of accuracy was achieved. For the NN-based techniques to be readily implemented in place of traditional PID controllers, the speed and accuracy of response must be consistently better than traditional controllers. In addition, these complex approaches must be simplified for the user to encourage their use. Finally, a simple design of a controller leads to a simple-to-use controller.

5. References

Adrianes H, De Koning WL, Banning R. Modeling piezoelectric actuators. IEEE/ASME Transactions on Mechatronics. 2000; 5(4):331–341. https://doi.org/10.1109/3516.891044

Bock SA et al. Proof of local convergence for the Adam optimizer. International Joint Conference on Neural Networks (IJCNN); 2019 July 14–19; Budapest, Hungary. IEEE; c2019 p. 4–6.

Borden Y et al. Dynamics and adaptive control of a hybrid piezoelectric-hydraulic actuator for hypersonic morphing control surface. Joint Army-Navy-NASA-Air Force (JANNAF) Meeting; 2023 May 22–26; Pittsburgh, PA. p. 8–12.

Chen X et al. Adaptive control for the systems with hysteresis and uncertainties. IFAC Proceedings Volumes. 2007;40(13):275–280.

Chong A et al. PID control system analysis, design, and technology. IEEE Transactions. 2005;13(13):559–576.

Cong S. PID-like neural network nonlinear adaptive control for uncertain multivariable motion control systems. IEEE Transactions on Industrial Electronics. 2009;56(10):3872–3879.

Haughn K et al. Deep reinforcement learning achieves multifunctional morphing airfoil control. Journal of Composite Materials. 2022;57(4):721–736.

Ioannou P et al. Robust adaptive control. Dover Publications; 2013.

Li W et al. Neural network self-tuning control for a piezoelectric actuator. Sensors. 2020;20(12):3342.

Patil M et al. Long short-term memory (LSTM) neural network-based system identification and augmented predictive control of piezoelectric actuators. ScienceDirect. IFAC PapersOnLine. 2021;54(20):38–45.

Yu Y et al. A review of recuurent neural networks: LSTM cells and network architectures. Neural Computation. 2019;31(7):1235–1270.

List of Symbols, Abbreviations, and Acronyms

Adam	Adaptive Moment Estimation
AI	artificial intelligence
ARL	Army Research Laboratory
CPU	central processing unit
Ct	candidate gate
DC	direct current
DEVCOM	U.S. Army Combat Capabilities Development Command
DoD	Department of Defense
Ft	forget gate
GUI	graphical user interface
Ht	hidden gate
It	input gate
LSTM	long short-term memory
ML	machine learning
MRAC	model reference adaptive control
NN	neural network
NRL	Naval Research Laboratory
Ot	output gate
PID	proportional–integral–derivative
RMSprop	root mean squared propagation
RNN	recurrent NN

1 (PDF)	DEFENSE TECHNICAL INFORMATION CTR DTIC OCA
1 (PDF)	DEVCOM ARL FCDD RLB CI TECH LIB