IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Controlling a Buck Converter with a Neural Network

Nick Cardamone[1], Electrical Engineering Student
[1]University of Ottawa, Canada

ABSTRACT In this work, a buck converter was simulated using MATLAB and Simulink with a neural network control loop wrapped around it. The simulation was done to show that an intelligent control system using a neural network is possible. The results showed this is indeed the case using the NARMA L2 controller provided by the MATLAB deep learning toolbox, but with several caveats and short comings. The most important conclusions gathered were that more effort is required to get the NARMA L2 controller to a state where ease of design is possible. Where as the current controller version did not allow for flexibility and customization in the design phase. Second most import consideration is practical implementation of such a neural network controller. The typical feedforward architecture was not suited to control a dynamical system where as a modification on a recurrent neural network was shown to be effective. The last import insight is the possible power consumption of such a controller when implemented in hardware. For this to be an effective and feasible control strategy, significant effort must be put in to minimize the electrical power needed to run and compute this application.

INDEX TERMS Neural Network, Feedforward Neural Network, Recurrent Neural Network, Buck Converter, Switch Mode Power Supply, Microcontroller, ASIC, FPGA

I. INTRODUCTION

In modern electronics, it is often necessary to convert one DC voltage to another using a power converter. This power converter is a circuit that either relies on a transistor to act as a linear regulator or a complex switching topology built from energy storage elements and transistors among other components. In this article, a simulation of a DC to DC buck converter is carried out with an innovative control law. Instead of using a traditional PID control loop, the NARMA L2 neural network controller was simulated using MATLAB and Simulink. It is the hope that these neural network controllers will become suitable candidates to replace PID controllers in modern switch mode power supplies. The ideal implementation of such a controller would an ASIC or FPGA but using a traditional microcontroller would also work.

II. Literature Review

In [1] a buck dc to dc converter was simulated using a neural network controller. The particular controller simulated was the NARMA L2 controller provided in the MATLAB deep learning toolbox. The results showed that it was possible to control a buck converter using an intelligent control law. However, the simulations in this paper are limited to a single step response. This limitation does not fully prove that it was possible to control the output voltage of a buck converter at a fine enough resolution.

In [2] a buck converter was simulated using a neural network predictive controller. This control law was then compared against a traditional PID control loop. The results showed that the neural network predictive controller was effective and can produce better results than the PI counter part. That wasn't always the case. In some of the presented test cases, there was considerable ripple on their output voltage and some strange behavior of the inductor current. It was also worth mentioning that their model predictive controller was very computationally expensive to run.

In [3] the authors propose a new approach to controlling a switch mode power supply. The plant example they chose was the common buck cell. The method of control being evaluated was a switch-based clustering algorithm. The simulated results show a 2.7% possible improvement in settling time and 0.6% improvement in overshoot as compared to a more traditional control law.

In [4] a simulation of a dc motor control system was performed. The particular controller being studied was a variant of a neural network, named the deep belief network. The results showed that their particular implementation was effective. However, their does not appear to be a significant difference in results between the simulated intelligent controller and the traditional PID controller.

In [5] the authors present a revised approach to non linear dynamical system modeling using their improved novel SINDy algorithm. The advantage of this algorithm was that it produced a sparse linear state space model that was not overly computationally expensive to implement in simulations. They then went on to robustify their algorithm to improve the results from their test simulations. To conclude the authors validate their work by performing a simulation of a reverse pendulum on a cart, where they apply a control law to stabilize the pendulum based on their SINDy algorithm.

III. Buck Converters

Looking at a buck converter from a high level, there are two distinct sub sections. The first switches in the supply voltage. This is done by applying a PWM signal to the gate of the MOSFET. The MOSFET then allows current to flow when the PWM is high and the diode allows the current to flow when the PWM is off. The second subsection is an LC low pass filter. This takes the switched input voltage and smooths it into a lower DC voltage. The resistor just acts as a load in this case.

For this analysis and simulation, some simplifying assumptions are made. The first is that the MOSFET acts as an ideal switch, such that the voltage drop across it is negligible. The second is the (Schottky) diode is ideal and has zero forward voltage. It is also accurate to say this buck cell is being modeled as an ideal synchronous buck cell, with the diode replaced with differently clocked MOSFET, and hence just apply assumption number one. The last simplification is switching/ circuit losses have been ignored

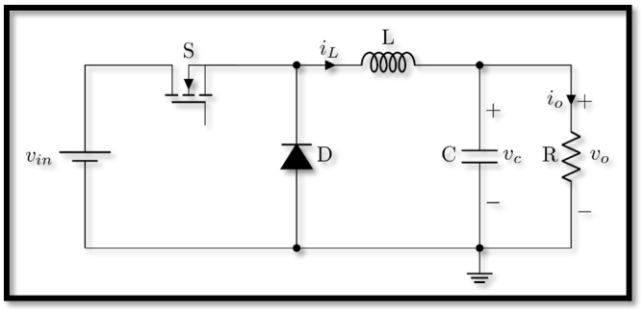

Figure 1: Buck Converter Schematic

The following are the mathematical models used in the simulations. To derive them, KCL, KVL and Ohm's Law are used to create a system of ODEs based on first principles.

$$\boldsymbol{x} = \begin{bmatrix} v_c \\ i_L \end{bmatrix} = \begin{bmatrix} v_{out} \\ i_L \end{bmatrix}, \quad \boldsymbol{A} = \begin{bmatrix} -\frac{1}{RC} & \frac{1}{C} \\ -\frac{1}{L} & 0 \end{bmatrix}, \quad \boldsymbol{B} = \begin{bmatrix} 0 \\ \frac{v_{in}}{L} \end{bmatrix}, \quad \boldsymbol{C} = [1 \quad 0], \quad \boldsymbol{D} = 0$$

$$\dot{\boldsymbol{x}} = \boldsymbol{A}\boldsymbol{x} + \boldsymbol{B} * pwm(t)$$

$$y = \boldsymbol{C}\boldsymbol{x} + \boldsymbol{D} * pwm(t)$$

V. Neural Network Control

In essence, a neural network is a structure that approximates a function. Where this structure takes an input state vector and uses a learned, nonlinear mapping to produce an output state vector. In the context of control, the neural network takes in a desired state as well as measurements of the system and produces the optimal control signal to drive our output to where we want it to be.

VI. Neural Networks

A. Basics

As the name implies, a neural network is a network of neurons, where a neuron is really a variable stored in memory. Traditionally, the value a neuron can take ranges between zero and one. We call the specific numerical value a neuron takes to be its activation. The network part of the name comes from how we connect neurons together.

IV. PID Control Systems

In traditional control systems, one of the most common controller architectures is the PID controller. In essence, the PID controller takes a weighted sum of a signal, the integral of that signal and the derivative of that signal. The following is the equation that describes a PID controller

$$y(t) = K_P * x(t) + K_I * \int_{-\infty}^{t} x(\tau)d\tau + K_D * \frac{dx(t)}{dt}$$

The following is a diagram depicting the generation of a single activation.

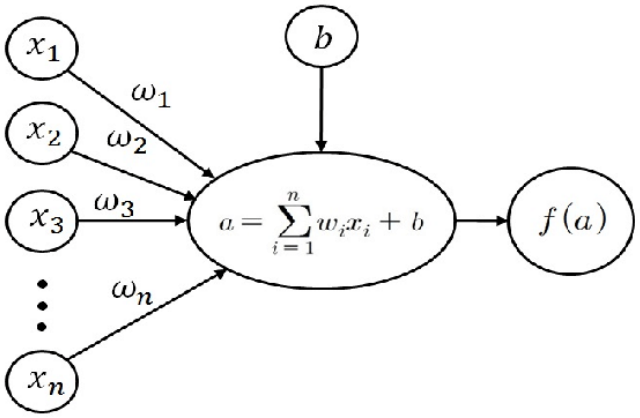

Figure 2: Single Neuron Activation Calculation [6]

From the diagram, we can see that the activation of a single neuron is calculated from a weighted sum of all the previous neurons activations plus some number to bias the neuron to usually on or usually off. The w's in the diagram are the weights of the network and the b's are the biases, as seen through out the literature on neural networks. After this summing step, the resultant activation is compressed down by an activation function to be in the interval between zero and one. Two examples of activation functions are the sigmoid and the ReLu.

B. Feedforward Neural Network

In order to use neurons in a design, they are connected together in a network. The most simple form of this network is called the feedforward neural network, named this way because the propagation of information only flows in one direction.

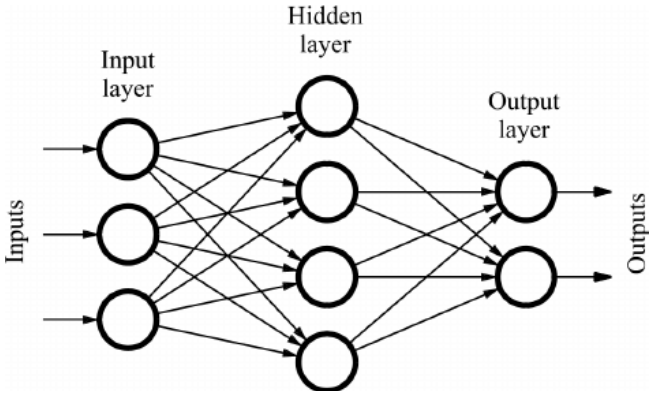

Figure 3: Feedforward Neural Network [7]

As seen in the above figure, a neural network is organized into layers. The first of those layers or the input layer, as the name suggests, collects the inputs to our network and converts them to an activation. After the input layer, there are the hidden layers consisting of regular neurons connected together. In the above figure, there is only one hidden layer as that is a simplification for this explanation. The final layer or output layer is where the network's decision takes place. The output layer collects the activations of the previous layers and is supposed to produce an output that is correlated to the network's decision.

C. Recurrent Neural Network

The feedforward network architecture is great for static, time independent data, but is sub optimal for dynamical systems which control is desired. For the systems of interest, a better class of architectures is the recurrent neural network. In this architecture, we start with the base feedforward configuration and feedback neurons to one another. This feedback gives the network a form of memory which allows for a better performance with time varying data or systems.

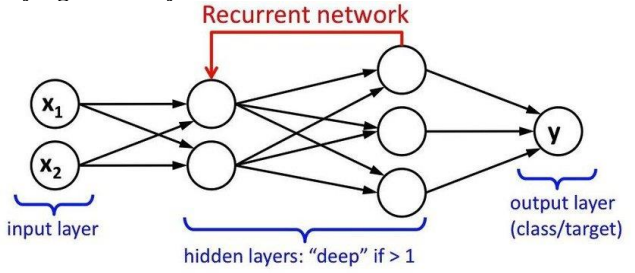

Figure 4: Recurrent Neural Network [8]

D. Network Training

In order to discover or learn what the optimal weights and biases are, the network must be trained using data gathered about the system. To do the training, the weights and biases are initialized randomly at first. Then the network is fed a training example, or subset of the total data available to train the network. After the network has produced a result, a comparison is made between what the network gave and what it should have been. This comparison is called a cost function. This cost function is then used to adjust the weights and biases of the network to produce a better result for the next training iteration. This process seeks to minimize the cost function using the algorithms of gradient decent and backpropagation. As this iterative training it taking place, the progress is assed with a separate database of system data to validate the network performance and to decide when to stop training.

E. Plant Modelling

In order to use a MATLAB neural network controller in a control system, the network needs to learn a model of the plant. This was done using a series of step inputs. The size and duration of the steps were random to maximize the accuracy of the model. As seen in the figure below, this training data was generated automatically by Simulink and fed into the training routine of the controller.

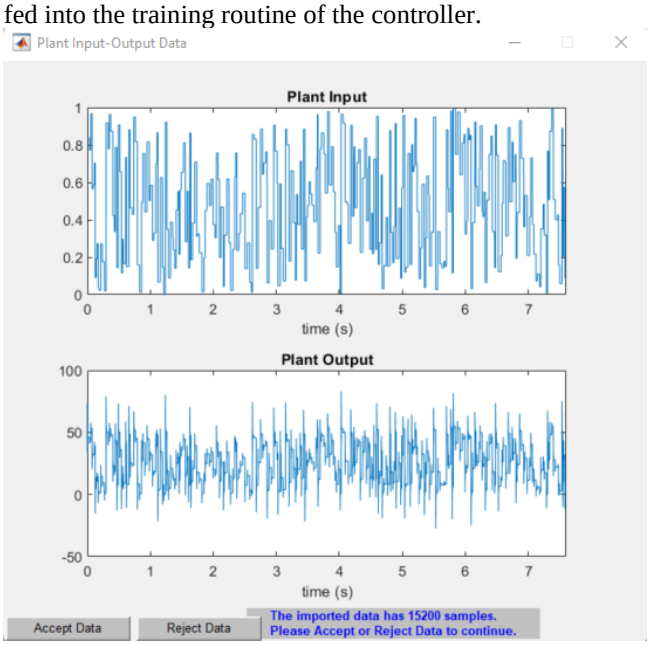

Figure 5: Autogenerated Network Training Data

VII. NARMA L2 Controller

To simulate a neural network controller, the NARMA L2 controller, provided in the MATLAB deep learning toolbox, was used. The internal architecture of the NARMA L2 controller is a variant of a recurrent neural network. The basic principle of the NARMA L2 controller is the controller tries to make a non linear system linear using a neural network, same kind of thing seen in Koopman operator theory, but using two separate function approximations to cancel out the non linearities. With this linearized model, an optimization is done.
The following is the simulation diagram.

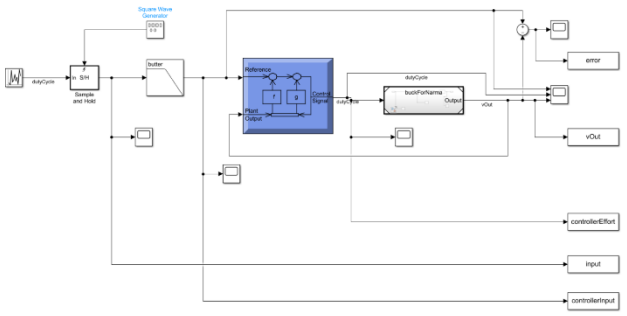

Figure 6: Buck Converter with NARMA L2 Controller

When the controller icon is clicked, the following menu pops up. The specific network parameters were chosen in an effort to minimize the complexity of the network while still providing a sufficient level of fidelity and performance.

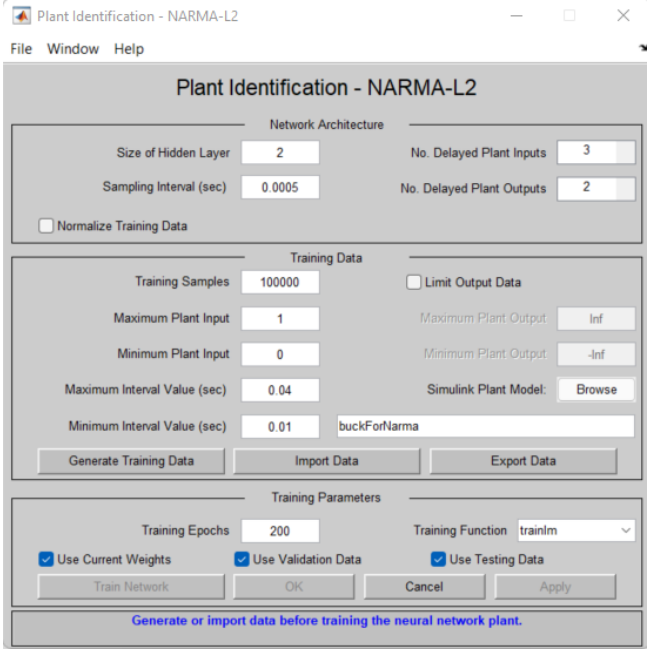

Figure 7: NARMA L2 Controller Setup

VIII. Simulation Results

After implementing and simulating the buck converter using the NARMA L2 block as the controller, the following results were obtained.

After providing the system with multiple, equally spaced (in time), step inputs, the following accumulated step response was observed.

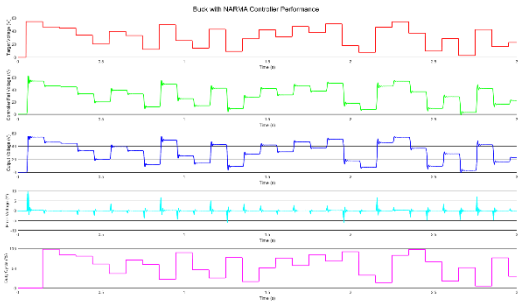

Figure 8: Closed Loop Step Responses, System Characterization

Zooming in on one particular step input, the following was observed.

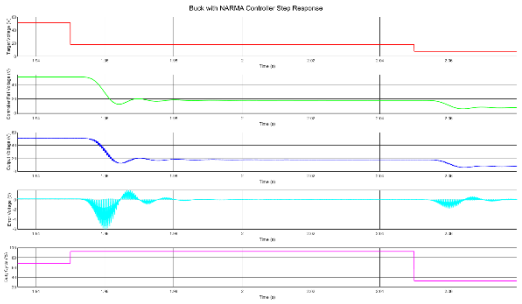

Figure 9: Closed Loop Step Response

Zooming in to the output voltage only, the following step response is observed.

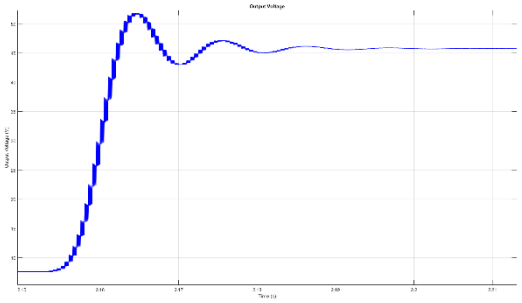

Figure 10: Output Voltage Step Response

As seen in the above plot, the following parameters are obtained

Table 1: Voltage Step Response Observations

Natural Frequency	100 Hz
Overshoot Voltage	55 V
Steady State Voltage	46 V
Percent Overshoot	20 %
Rise Time	5 ms
Settling Time	40 ms

After training the NARMA L2 controller, the following performance (mean squared error) plot was generated as follows along with a summary of the training metrics

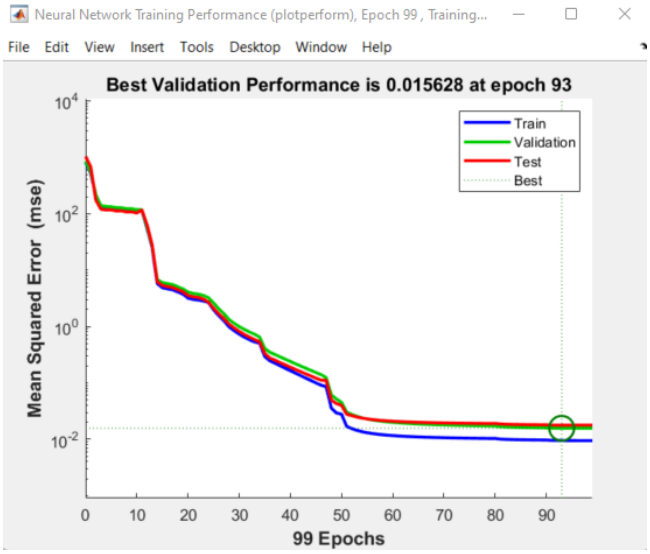

Figure 11: Network Training Performance

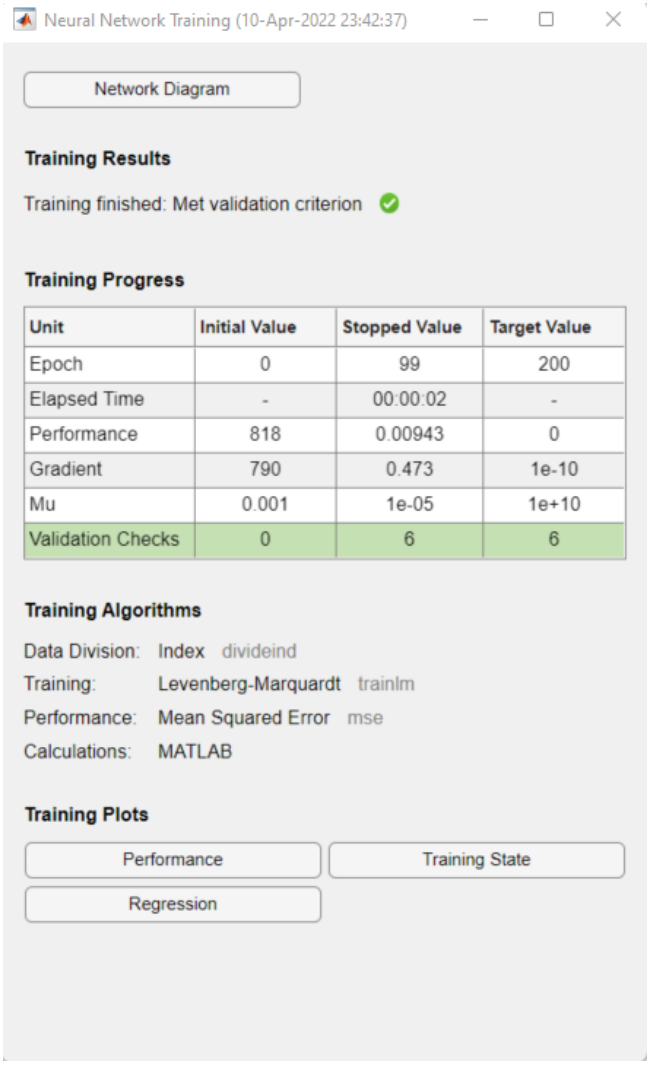

Figure 12: Network Training Summary

IX. Discussion

F. *Instability and Network Tuning*

As seen in the initial simulation results, the neural network produced an unstable system. After doing several iterations of the simulation, two root causes were discovered. The first is that the sampling rate of the controller must be slower than the natural frequency of the open loop oscillations. When this is not the case, the neural network controllers built into deep learning toolbox (MATLAB) are unable to work correctly. The second root cause has to do with bugs with in the neural network controller its self. The most significant bug has to do with the training routine producing an erroneous plant model.

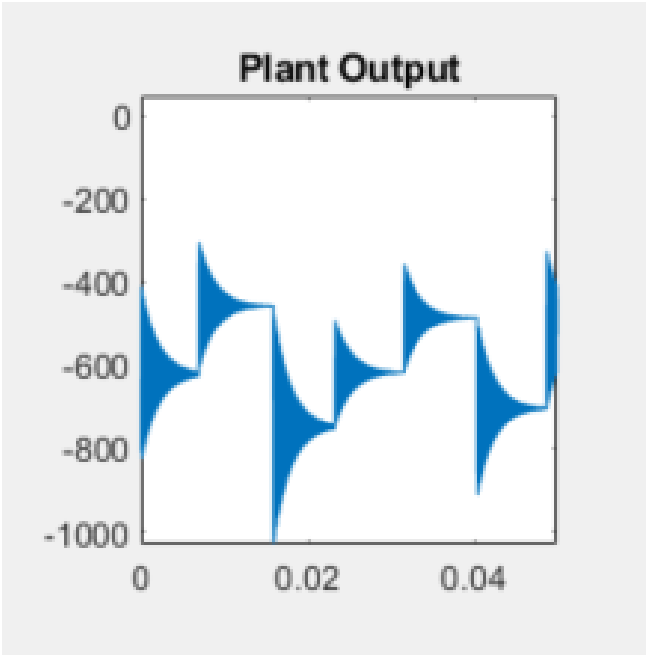

Figure 13: Erroneous Plant Output from Network Training

As seen in the figure above, the learned plant model of a buck converter produces a negative output from a positive supply voltage. When the training data was reloaded, this issue went away.

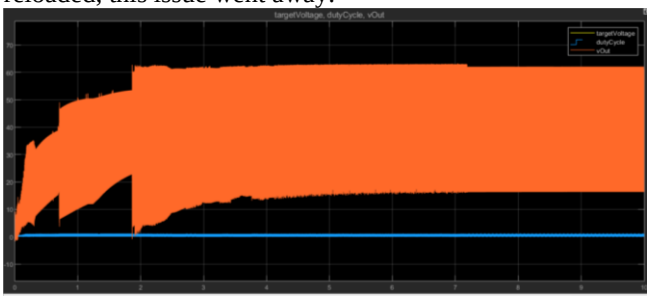

Figure 14: Initial (Unstable) Closed Loop Step Response

G. Error Detection and Correction

As is common knowledge with in the machine learning community, neural networks are not 100% accurate all the time. With in the field of machine learning controls, this can pose a problem in some circumstances. For example, the results of the simulations done in this work show an overshoot of 7 V. In cases where the output voltage of a switch mode power supply are critical, this overshoot might damage the components placed after the SMPS. To compensate for this, extra care must be taken to characterize these in accuracies, predict when they will occur and compensate for them.

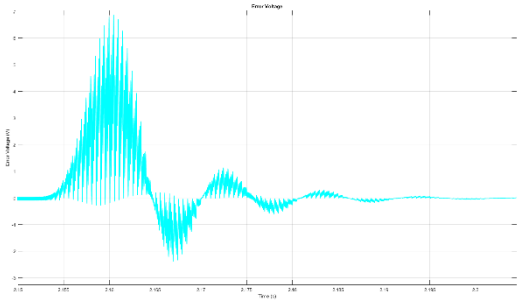

Figure 15: Step Response Error Voltage

H. Sustainability and Power Consumption

As the world moves to a more sustainable future, it is worth considering the power consumption of a neural network implementation in a microcontroller. When comparing a neural network to a more traditional controller, the machine instructions required for the neural network typically exceed the number needed for an older control strategy. This is due to the repeated multiplication and addition needed when traversing the neurons with in the network. For the neural network to be selected as an optimal control strategy, the operating power must be less than or equal to the power required to implement a traditional controller, such as a PID loop. Alternatively, specialized hardware could be designed, such as an ASIC, to deploy the neural network on to increase the power efficiency.

I. Loop Shaping

Within the controllers provided in the MATLAB deep learning toolbox, there are no parameters to tune to get the desired transient response. Instead, the controller assumes the desired rise time is infinite/ the maximum possible but limited by the sample rate of the simulation. When designing control systems, this is rarely the case, as a fast rise time usually leads to more overshoot (when the system is linear). In some cases, the desired rise time is on the order of milliseconds to seconds depending on the application.

In this simulation with the NARMA L2 controller, this was achieved by adding a prefilter to the reference input. The goal of the prefilter was to reduce rise time of the reference signal, thus slowing the system output and achieving that slower rise time with a reduced overshoot. In practice however, this prefilter defeats the purpose of the neural network controller, which is to have a single controller to produce the desired system properties. To achieve this only using a neural network, a cost function must be defined that penalizes the appropriate parameters when the network is being trained. This custom cost function does not appear to be part of the GUI for the neural network controller that comes with the deep learning toolbox. To have this option implemented in a GUI would be a valuable product improvement.

X. Conclusion

The neural network controller approach to a buck system is viable and feasible, as demonstrated in this work. However, there are still several hurdles to overcome before widespread implementation. The more important hurdles are reliability and accuracy, ease of design and controller power consumption when implemented in hardware, whether that be in a microcontroller, FPGA or ASIC. The main findings were that the NARMA L2 controller can be used in a simulation to control the output voltage of a buck converter, but a costly prefilter had to be added to get desirable system properties. This prefilter is undesirable in a practical implementation but the overall architecture of the NARMA L2 controller is still promising. More effort is required to allow further customization of the controller to allow for custom cost functions and error detection and correction to take place.

XI. References

[1] G. Vacheva, N. Hinov, H. Kanchev, and B. Gilev, "Application of neural network for smart control of a buck DC/DC converter," 2019 42nd International Spring Seminar on Electronics Technology (ISSE), 2019.

[2] S. Saadatmand, P. Shamsi, and M. Ferdowsi, "The voltage regulation of a buck converter using a neural network predictive controller," 2020 IEEE Texas Power and Energy Conference (TPEC), 2020.

[3] B. W. Abegaz and M. Cmiel, "Smart control of Buck converters using a switching-based clustering algorithm," 2019 14th Annual Conference System of Systems Engineering (SoSE), 2019.

[4] K. Cheon, J. Kim, M. Hamadache, and D. Lee, "On replacing PID controller with Deep Learning Controller for DC motor system," Journal of Automation and Control Engineering, vol. 3, no. 6, pp. 452–456, 2015.

[5] K. Kaheman, J. N. Kutz, and S. L. Brunton, "Sindy-pi: A robust algorithm for parallel implicit sparse identification of nonlinear dynamics," Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences, vol. 476, no. 2242, p. 20200279, 2020.

[6] “Application of intelligent switching ... - researchgate.net,” Structure-and-mathematical-modeling-of-the-node-in-neural-network.png. [Online]. Available: https://www.researchgate.net/profile/Kyoung-Kwan-Ahn/publication/242586404_APPLICATION_OF_INTELLIGENT_SWITCHING_CONTROL_OF_PNEUMATIC_ARTIFICIAL_MUSCLE_MANIPULATORS_WITH_MAGNETO-RHEOLOGICAL_BRAKE/links/5414fcbf0cf2fa878ad3ed98/APPLICATION-OF-INTELLIGENT-SWITCHING-CONTROL-OF-PNEUMATIC-ARTIFICIAL-MUSCLE-MANIPULATORS-WITH-MAGNETO-RHEOLOGICAL-BRAKE.pdf?_sg%5B0%5D=started_experiment_milestone&origin=journalDetail. [Accessed: 13-Apr-2022].
[7] “2nd place solution in google AI open images object detection track 2019,” DeepAI, 17-Nov-2019. [Online]. Available: https://deepai.org/publication/2nd-place-solution-in-google-ai-open-images-object-detection-track-2019. [Accessed: 12-Apr-2022].
[8] 5. “Comprehensive and comparative analysis of neural network,” Recurrent-neural-networkRNN-or-Long-Short-Term-MemoryLSTM-5616.png. [Online]. Available: https://www.researchgate.net/profile/Vidushi-Mishra/publication/324883736_COMPREHENSIVE_AND_COMPARATIVE_ANALYSIS_OF_NEURAL_NETWORK/links/5eff046e458515505087a3e7/COMPREHENSIVE-AND-COMPARATIVE-ANALYSIS-OF-NEURAL-NETWORK.pdf. [Accessed: 13-Apr-2022].