Mitigation of Applied Load Using Machine Learning for Adaptive Motor Control

Kourosh Rahnamai, Jacob Rollins, Luke Moisan, Ryan Kayfus
ECE Department
Western New England University
Springfield, USA

***Abstract*—This paper investigates the use of machine learning to estimate the duty cycle of a pulse-width modulated (PWM) signal for motor control applications. Emerging industry trends show a greater focus on the use of machine learning algorithms in control systems over traditional proportional-integral-derivative (PID) controllers. Results demonstrate the machine learning model's accuracy below a targeted 3% average, highlighting the importance of optimizing model design for practical implementation.**

I. Introduction

This study applies machine learning to a DC motor control system. Proportional Integral Derivative controllers are an industry standard for control systems and require precise tuning. This study aims to provide an alternative control method based on a machine learning algorithm. The PWM duty cycle for a DC motor was controlled over a 30-second simulation, ensuring RPM adherence to a target curve despite load variations. The motivation stems from the need for a motorized warehouse cart that transports packages of varying weights along a straight track within a fixed time frame. The duty cycle updates every second to maintain the desired RPM.

A Simulink model of the DC motor system, with counteractive torque simulating load, generates training data for a function-fitting machine learning algorithm. The model maps input loads to required duty cycles over the travel period.

This approach has significant implications for warehouse automation by demonstrating how machine learning can be used in control systems. The machine learning algorithm ensured consistent timing, enhanced system efficiency, reliability, and accuracy in automated logistics as shown by the high system accuracy. Typical industry standards target an error of less than five percent.

II. Design

A. Simulink Simulation

A Simulink-based DC motor simulation was developed to collect motor system data without the high costs and complexities of physical testing. The simulation included an adjustable load, precise to one decimal place, enabling controlled data collection. The motor was driven by a pulse-width modulated (PWM) signal, which regulated power delivery by switching the supply voltage on and off.

The PWM signal was generated using a 100 Hz triangle wave, which increased linearly from 0 to 100, defining duty cycles from 0% to 100%, as illustrated in Fig. 1 and Fig. 2. This configuration ensured smooth motor control with 1% duty resolution [2]. The PWM frequency was set to 100 Hz to balance simulation accuracy and the motor's low-pass filter behavior. As shown in Fig. 3, higher frequencies exceeded solver constraints, while lower frequencies caused erratic motor response.

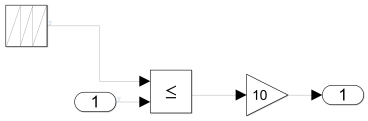

Fig. 1. Simulink PWM module.

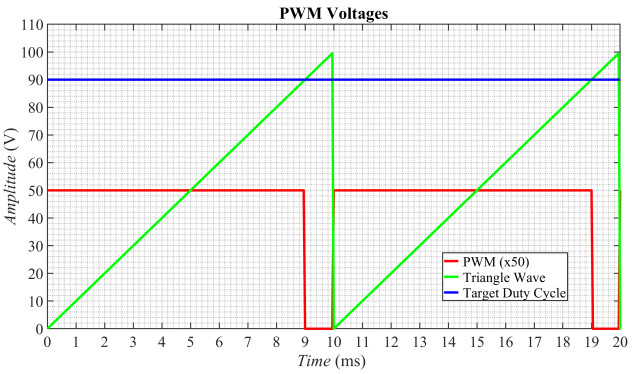

Fig. 2. Simulink PWM signal generation.

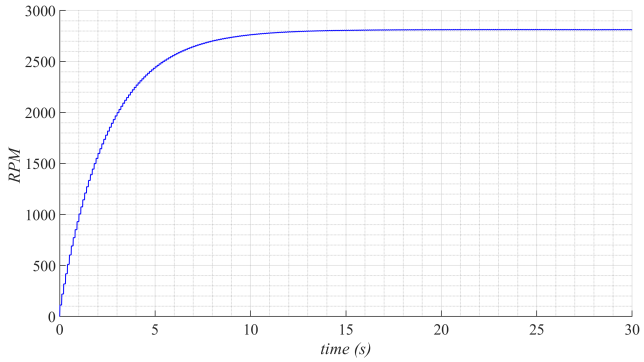

Fig. 3. Jagged motor response.

A relational operator converted the triangle wave into a PWM signal by comparing its instantaneous value to the target duty cycle. Since this Boolean signal alone could not drive the motor, a voltage amplifier module, consisting of a transistor and a 12 V power source, was implemented. The transistor, controlled by the PWM signal, switched the motor between 0 V and 12 V, achieving precise duty cycle modulation.

A 12 V, 3.6 W brushed DC motor was modeled with a maximum RPM of 2500 under a counteractive torque of 9.5986E−3 Nm. Though not based on a specific commercial motor, its parameters could be adapted for real-world applications.

A dynamic load module was incorporated using an ideal torque source. As illustrated in Fig. 4, load values ranging from 0 to 10 were scaled by 9.6E−4, applying torque between 0 and 9.5986E−3 Nm. This module was essential for simulating real-world load variations without physical replication [3].

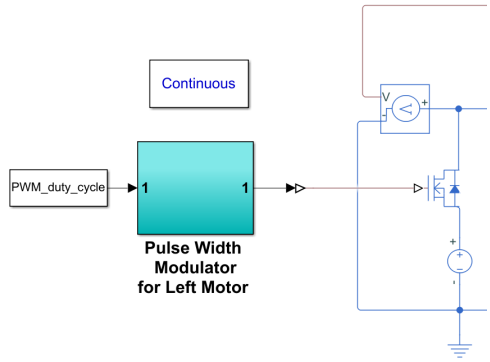

Fig. 4. Simulink PWM signal generation system.

After configuring the motor and load modules, the Simulink simulation was completed, as shown in Fig. 5.

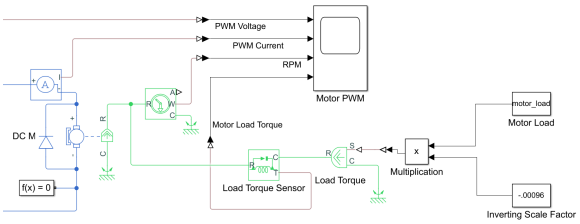

Fig. 5. Simulink DC motor system.

B. Machine Learning Model

The machine learning model used in this study was adapted from MATLAB's "Body Fat Estimation" example, which predicts body fat percentage based on anatomical measurements using a function-fitting machine learning algorithm [4]. The algorithm was modified to accept applied loads as inputs and generate duty cycles as outputs for motor control. The function-fitting algorithm are well suited to approximate complex functions by learning correlations from data. Accurate training data collection was essential for a precise model response.

The training dataset consisted of motor loads ranging from 0 to 10, incremented in 0.5-step intervals, corresponding to a maximum counteractive torque of −3 ft-lb. These values were selected as scale factors based on the motor specifications in the Simulink simulation. The simulated system represented a motorized warehouse cart transporting packages of varying weights along a straight track within a fixed 30-second period.

As illustrated in Fig. 6, the machine learning algorithm was implemented as a two-layer feedforward neural network with one hidden layer utilizing five neurons, one input, and 30 outputs. It functioned as a function-fitting model, mapping applied loads to corresponding duty cycle values, while generalizing for previously unseen inputs.

Network creation was performed using MATLAB's "fitnet" function, with the number of hidden-layer neurons as the primary parameter. The network, along with input loads and target duty cycles, was trained using MATLAB's "train" function.

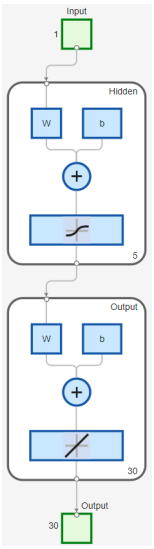

Fig. 6. Machine learning algorithm architecture.

III. Procedure

A. Training Data Collection

The target duty cycles were determined using a reference RPM curve generated from a 30-second simulation with zero applied load and a constant duty cycle of 20%, as illustrated in Fig. 7. Under these conditions, the motor's RPM increased for approximately 20 seconds before stabilizing at around 2600 RPM. A 20% duty cycle was selected to provide sufficient headroom for load compensation. If the target curve had been generated with a 100% duty cycle, any additional load would have prevented the controller from matching the target RPM due to the lack of available power. This reference curve served as the baseline trajectory for the motor under different load conditions and was used to derive the target duty cycles required to maintain the desired performance.

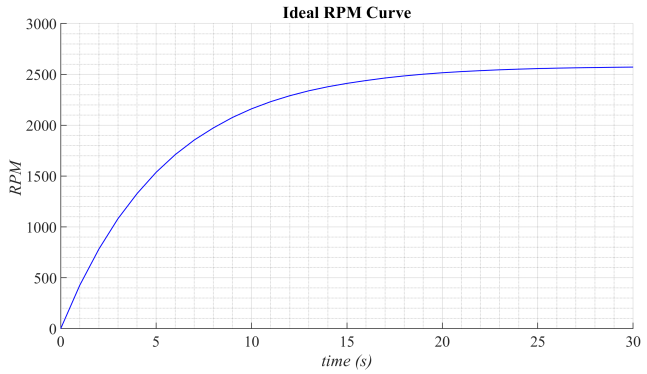

Fig. 7. Target RPM curve.

To automate data collection, the Simulink simulation parameters were initialized using a MATLAB script, enabling sequential simulations under varying conditions. The motor's RPM was sampled once per second from startup, producing 30 RPM values per simulation. These values were stored in a data file and later processed to determine the corresponding duty cycles.

The training dataset was generated by comparing the actual RPM of the motor under load to the target RPM curve at discrete time intervals. The simulation began with the rotor speed initialized to zero and a starting duty cycle of 20%. Load values ranged from 0.5 to 10 in increments of 0.5, with 10 corresponding to the maximum applied torque. These values were used as reference points in the plots to illustrate the increasing load in a simple unit format. Although the actual loads were relatively small for the motor's capabilities, they were sufficient to produce a measurable effect on its response.

For each load step, the simulation was executed for one second, after which the actual RPM was compared to the target RPM at that timestamp. The algorithm then evaluated whether the actual RPM was within ±1% of the target, exceeded it, or fell below it. If the actual RPM was within tolerance or exceeded the target, the duty cycle was recorded, and the simulation proceeded to the next time step. If the actual RPM was below the target, the duty cycle was incremented by 1%, and the simulation was re-run until the target RPM was achieved.

To improve computational efficiency, previously calculated values were leveraged to refine the starting conditions for each subsequent time step. At the first time step (0 to 1 second) for each applied load, the rotor speed was initialized to 0 RPM, as the motor was not yet in motion. The duty cycle was increased in increments until the actual RPM matched the target RPM curve, at which point the corresponding duty cycle was recorded. Instead of resetting the duty cycle to 20% at each new time step, the initial duty cycle for the next step was set to 1% below the previously recorded duty cycle. Additionally, the initial rotor speed for each new time step was set to the RPM recorded at the end of the previous time step. This approach accounted for the motor's existing momentum, eliminating unnecessary recalculations and reducing simulation time. The process continued for each time step up to 30 seconds, ensuring that the model efficiently determined the correct duty cycle at every stage of motor operation.

Once all target duty cycles were determined for a given load, the load was incremented by 0.5, and the matching process was re-run starting from the 0 to 1-second timestamp. This iterative approach was applied to all loads up to 10, with each new load case following the same duty cycle matching process while leveraging previously recorded RPM values to optimize computational efficiency.

B. Training

The model was trained using duty cycles that matched the target RPM curve. These duty cycles, recorded at one-second intervals over 30 seconds for each load, ensured the motor response aligned with the target. The training inputs consisted of applied loads, while the target outputs were the corresponding sets of 30 duty cycle values. By processing these associations, the model learned to reproduce the required duty cycles for a given load. Once it successfully generated the expected outputs, its ability to generalize to unseen inputs was evaluated.

C. Testing

After initial training, the model was tested to assess its ability to generalize. The model was tested with load values absent from the training dataset, including non-integer loads such as 5.7, outside the original 0.5-step increments up to ten. In early testing, it showed moderate success in tracking the target RPM curve. Based on performance, adjustments were made to improve accuracy by modifying algorithm parameters. As shown in Fig. 8 and 9, reducing the number of neurons improved tracking precision [5]. Testing intermediate values assessed the model's ability to interpolate and produce accurate duty cycles for untrained loads [6].

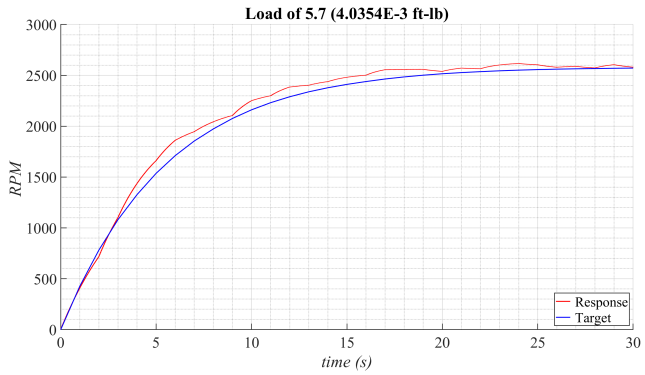

Fig. 8. Load case trained using 15 neurons.

IV. Results

A. Generalization to Unseen Load Values

To evaluate generalization, the model was tested with unseen load values outside the original 0.5-step increments up to ten [7]. The model successfully interpolated and maintained

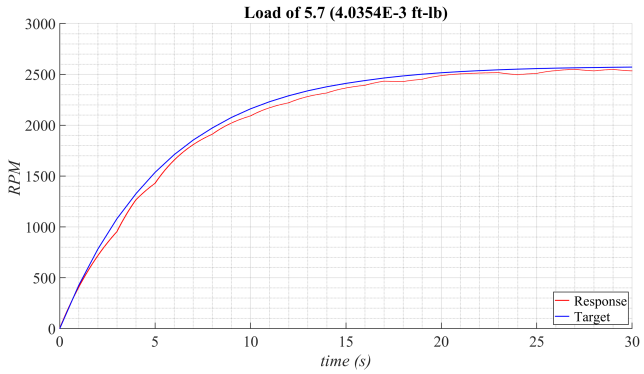

Fig. 9. Load case trained using 10 neurons.

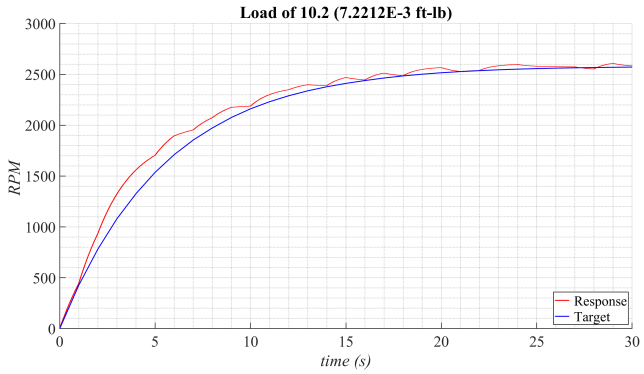

Fig. 11. Load of 10.2.

accurate RPM tracking using five neurons, as illustrated in Fig. 10. These results suggest applicability to real-world adaptive disturbance rejection. Further improvements could be achieved by reducing sampling time, enabling faster response. This study confirms that machine learning can be used to dynamically adjust motor duty cycles to maintain target RPMs under varying loads.

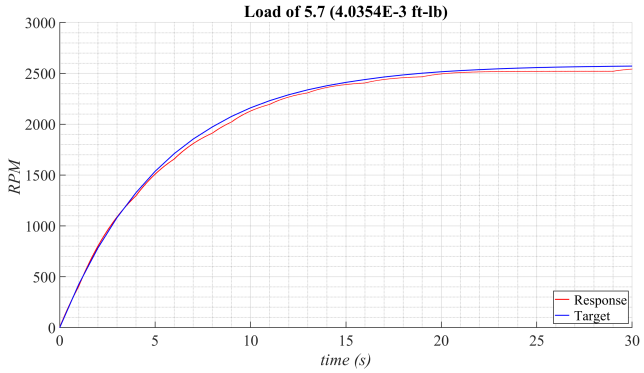

Fig. 10. Load case trained using 5 neurons.

B. Limitations with Loads Exceeding Training Range

When tested with loads exceeding 10, the model failed to track the target RPM curve, as shown in Fig. 11. Performance degraded as load values exceeded the training range, a common limitation in machine learning models. This highlights the importance of comprehensive training data in enhancing generalization and robustness.

C. Exhaustive Testing with Incremental Load Values

The final model underwent further testing to assess accuracy. It was evaluated on 84 test loads from 0.1 to 9.9 in 0.1 increments, excluding training values. Percent error was calculated by comparing the actual motor response to the target RPM curve, yielding an average error of 1.87% across all tests, below the targeted 3% average.

D. Error Analysis and System Response

Test results confirmed the model's ability to compensate for load disturbances. As shown in Fig. 12, for a test load of 5.7, the percent error initially peaked at 5%, then rapidly stabilized between 1% and 2% as the model adjusted. This initial error spike was due to a delay between receiving motor data and updating the duty cycle. A higher sampling rate could reduce response lag, improving startup accuracy. Overall, within the trained load range, the model consistently tracked the target RPM curve with the desired precision.

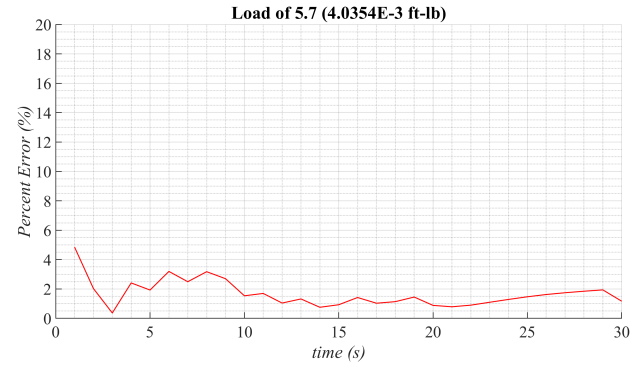

Fig. 12. Percent error over time for load of 5.7.

V. Conclusion

A machine learning algorithm was developed to estimate the duty cycle required to maintain a DC motor's RPM along a predefined target curve, independent of applied load. The model was trained using expected motor loads as inputs and corresponding duty cycles as outputs, with Simulink simulating the motorized warehouse cart system. While the study focused on a simplified motor response for analysis, this approach is applicable to more complex motor control scenarios.

After training, the model was tested with unseen load values, resulting in an average percent error of 1.87%, below the targeted 3% average. The model successfully controlled the motor's RPM, tracking the target curve throughout a 30-second simulation. This demonstrates the potential of machine learning to optimize motor performance in adaptive load conditions, with implications for advancing industrial automation and robotic systems.

REFERENCES

[1] J. David Rojas , O. Arrieta , R. Vilanova,, "Industrial PID controller tuning," 2021, pp. 1–147, doi: https://scholar.google.com/scholar?hl=en&as_sdt=0%2C30&q=what+are+the+pros+and+cons+of+a+tuned+pi+controller&btnG=#d=gs_qabs&t=1751496519598&u=%23p%3DOSUsiwrA0mcJ.

[2] D. C. Rus, N. S. Preda, I. I. Incze, M. Imecs, and Cs. Szabo, "Comparative analysis of PWM techniques: Simulation and DSP implementation," 2010, pp. 1–6, doi: https://doi.org/10.1109/aqtr.2010.5520764.

[3] K. J.P., N. Withana, K. Gallage, J. Wijayarathna, and A. Wijethilake, "Development of a Programmable Mechanical Motor Loading Unit using a DC Motor," 2019, pp. 211–215, doi: https://doi.org/10.1109/mercon.2019.8818829.

[4] "Body Fat Estimation," Mathworks.com. [Online]. Available: https://www.mathworks.com/help/deeplearning/ug/body-fat-estimation.html.

[5] Hush, "Classification with neural networks: a performance analysis," 1989, pp. 277-280, doi: https://doi.org/10.1109/icsyse.1989.48672.

[6] I. Bilbao and J. Bilbao, "Overfitting problem and the over-training in the era of data: Particularly for Artificial Neural Networks," 2017, pp. 173-177, doi: https://doi.org/10.1109/intelcis.2017.8260032.

[7] N. Feng, F. Wang, and Y. Qiu, "Novel Approach for Promoting the Generalization Ability of Neural Networks," 2006, pp. 131-135, Available: https://citeseerx.ist.psu.edu/document?repid=rep1&type=pdf&doi=08c04e908eecbcc870531b84e3d2ece17e370362.