International Journal of Engineering Applied Sciences and Technology, 2025
Vol. 10, Issue 02, ISSN No. 2455-2143, Pages 19-26
Published Online June 2025 in IJEAST (http://www.ijeast.com)

DEVELOPMENT OF GATED RECURRENT UNIT (GRU) MODELS FOR EFFICIENT ADAPTIVE PID PARAMETER PREDICTION IN REAL-TIME DC MOTOR SPEED REGULATION

Hatim O. ELhassan
PhD Student
Alsalama College of science & technology,
Khartoum-Sudan

M.I.Shukri
Professor
Alsalama College of science & technology,
Khartoum-Sudan

Abstract—**Achieving optimal real-time speed regulation in DC motors remains challenging with conventional fixed-gain PID controllers due to system nonlinearities and uncertainties. This paper proposes a novel approach employing a Gated Recurrent Unit (GRU) neural network for real-time, adaptive prediction of proportional (Kp), integral (Ki), and derivative (Kd) parameters. The GRU model is designed to process time-series dynamic features, including error, integral, and derivative terms. It is trained on a synthesized dataset, learning intricate relationships to optimal PID gains for continuous parameter adjustment. Simulation results demonstrate GRU's superior computational efficiency and real-time adaptability. It effectively captures temporal dependencies, offering quick responses and achieving rapid stabilization in dynamic environments. This GRU-based adaptive controller significantly advances intelligent and adaptive control systems for real-world DC motor speed regulation.**

Keywords—**DC Motor Control, Artificial Neural Network (ANN), Gated Recurrent Unit (GRU), Long Short-Term Memory (LSTM), PID Control, Parameter Prediction, Speed Regulation.**

I. INTRODUCTION

Direct current (DC) motors are fundamental components of electromechanical systems, having been integral to industrial applications such as robotics, automotive systems, aerospace, and automation since their invention in the 19th century [1]. Their widespread use derived from their ability to convert electrical energy into precise mechanical motion, enabling controlled speed, torque, and position in critical tasks [2]. Despite their reliability and simplicity, DC motors require advanced control strategies to maintain performance under real-world nonlinearities such as load variations, friction, and thermal effects [3]. Traditional Proportional-Integral-Derivative (PID) controllers, while widely used, assume linear system behavior and fixed parameters, limiting their adaptability in dynamic environments [4]. This necessitates the development of intelligent control techniques capable of responding to time-varying conditions. Adaptive control methods, particularly those integrating artificial neural networks (ANNs), offer promising solutions by continuously learning and adjusting to system dynamics [5]. This paper focuses on implementing Gated Recurrent Unit (GRU) networks—an ANN architecture specialized for time-series prediction—to enhance DC motor speed control precision and robustness. Despite GRUs' efficiency in modeling temporal dependencies, their application in motor control remains largely unexplored, especially in real-time scenarios [6]. The absence of systematic GRU-based evaluations in adaptive PID control highlights a significant research gap.
The Paper addresses the question: *How to make an artificial neural network model implementation of adaptive PID control of DC motor speed that is reliable and robust to maintain precise control under varied disturbances? * By developing GRU-based controllers optimized for computational efficiency and temporal adaptability, this paper advances motor control methodologies through Python-based simulations evaluated using rise time, settling time, and overshoot metrics [7]. It contributes a novel, efficient framework for real-time adaptive control in industrial applications.

II. DC MOTOR

A Direct Current (DC) motor is an electromechanical device that converts electrical energy into mechanical motion [8]. Its construction consists of two primary components: the stator and the rotor. The stator, the stationary part, contains poles excited by a direct current to produce a magnetic field. The rotor, the rotating component, features a laminated iron core with slots housing coils. These coils are connected in series and interact with the magnetic field to generate motion [9].

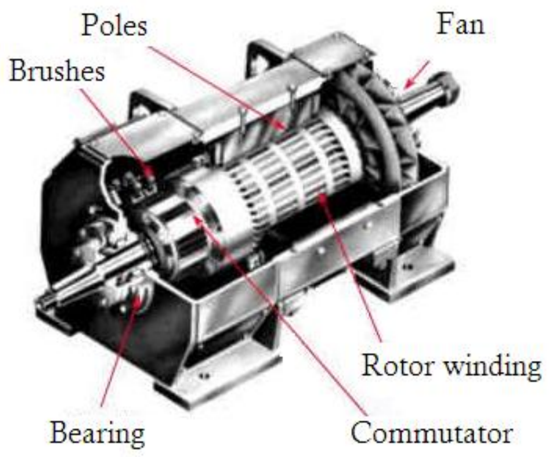

Fig. 1. DC motor rotor construction

The principle of operation of a DC motor is rooted in electromagnetic induction and torque generation. Electromagnetic induction occurs when the rotor rotates within the stator's magnetic field, causing the armature coils to cut through magnetic flux lines. This induces an electromotive force (EMF), known as back EMF:

$$E_b = K_e\omega\#(1)$$

WhereE_b is the back EMF, $K_e$is the motor's back EMF constant, and ω is the angular velocity of the rotor.

Torque generation arises from the interaction between the magnetic field and current flowing through the armature windings. The resulting torque drives the rotor and is expressed as:

$$T = K_t I_a\#(2)$$

where T is the torque, $K_t$is the torque constant, and I_a is the armature current.

The voltage equation for the armature circuit is given by:

$$V_a = E_b + I_a R_a + L_a \frac{dI_a}{dt}\#(3)$$

In steady-state conditions ($\frac{dI_a}{dt} = 0$), this simplifies to:

$$V_a = K_e\omega + I_a R_a\#(4)$$

These equations collectively describe how applied voltage and current influence motor speed and torque, underpinning its operational dynamics [10].

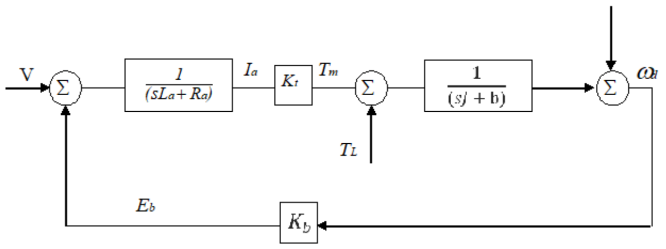

Fig. 2. DC Motor Block Diagram

III. PID CONTROLLER

The Proportional-Integral-Derivative (PID) controller is an important of control engineering, widely used for its simplicity and effectiveness in regulating dynamic systems such as DC motors. A PID controller minimizes the error between the desired setpoint (ω_{set})and the actual output $\left(\omega(t)\right)$ by applying a weighted sum of three terms: proportional, integral, and derivative. The control signal $\left(V(t)\right)$, which corresponds to the applied voltage in the case of a DC motor [11], is computed as:

$$V(t) = K_p \cdot e(t) + K_i \cdot \int e(t)\ dt + K_d \cdot \frac{de(t)}{dt}\#(5)$$

Where:

- $e(t) = \omega_{set} - \omega(t)$: Error between the set speed and the current speed.
- K_p: Proportional gain, which directly reduces the error.
- K_i: Integral gain, which eliminates steady-state error by accumulating past errors over time.

K_d: Derivative gain, which minimizes overshoot and oscillations by predicting future trends based on the rate of change of the error [11].

IV. GATED RECURRENT UNIT (GRU)

Gated Recurrent Units (GRUs) are a class of recurrent neural networks (RNNs) designed to efficiently process sequential data by addressing the vanishing gradient problem inherent in traditional RNNs [6]. Unlike standard architectures, GRUs employ gating mechanisms that regulate information flow, enabling selective memory retention and rapid adaptation. Compared to Long Short-Term Memory (LSTM) units, GRUs have a simplified structure with fewer parameters, making them computationally efficient while preserving temporal modeling capabilities. A GRU unit comprises an input layer that feeds sequential data into the network, a hidden layer responsible for recurrent computation, and an output layer that produces the final result. The core components include two adaptive gates: the reset gate rt and the update gate zt. These gates control how much past information is retained or updated at each time step:

$$rt = \sigma(Wr \cdot [ht-1, xt] + br)\#(6)$$
$$zt = \sigma(Wz \cdot [ht-1, xt] + bz)\#(7)$$

The reset gate determines which portion of the previous hidden state h_{t-1} to forget, while the update gate regulates how much of the candidate activation vector $\widetilde{h_t}$ influences the new hidden state ht:

$$\widetilde{h_t} = tanh(Wh \cdot [rt \odot ht - 1, xt] + bh)\#(8)$$
$$ht = (1 - zt) \odot ht - 1 + zt \odot h{\sim}t\#(9)$$

The equations above represent the Gated Recurrent Unit (GRU) operations, r_t is the reset gate, z_t is the update gate, $\widetilde{h_t}$is the candidate activation vector, and h_t is the hidden state at time t. σ denotes the sigmoid function, and $tanh$ is the hyperbolic tangent function. W_r, W_z, W_h are weight matrices, b_r, b_z, b_h are bias terms, h_{t-1} is the previous hidden state, and x_t is the current input. The gates control information flow, while $\widetilde{h_t}$ computes a candidate hidden state influenced by the reset gate. Finally, h_t combines previous and new states using the update gate.
This mechanism allows GRUs to maintain relevant historical context while adapting quickly to new inputs, making them ideal for real-time applications such as DC motor speed control [6]. Their advantages include computational efficiency, fast inference, simplified hyperparameter tuning, and robustness with limited datasets. In control systems, GRUs offer superior adaptability under dynamic conditions due to their ability to capture temporal dependencies with reduced complexity. This makes them a practical choice for resource-constrained environments where responsiveness and efficiency are critical.

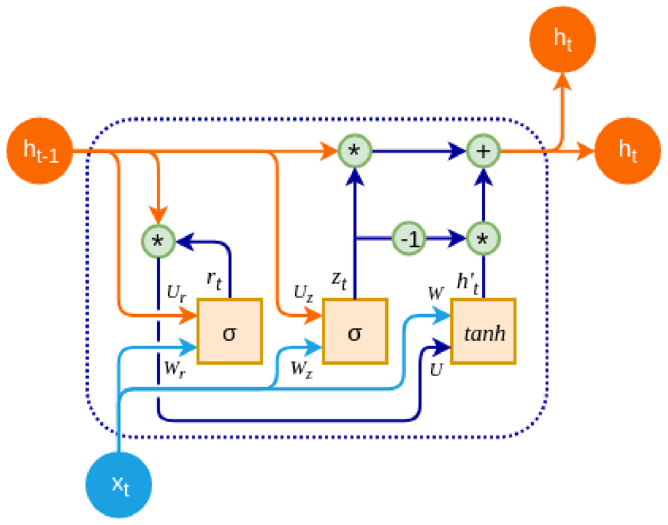

Fig. 3. GRU Cell

V. METHODOLOGY

A. DC Motor Model assumptions

While the DC motor model presented here is comprehensive, certain assumptions are made to simplify the analysis:

- Linearity: The relationships between voltage, current, torque, and speed are assumed to be linear. Nonlinear effects such as magnetic saturation are neglected.
- Constant Parameters: Parameters such as $(R_a), (L_a), (K_e), (K_t), (J), and (b)$are assumed to remain constant. Variations due to temperature, wear, or other factors are ignored.
- Ideal Conditions: Effects such as eddy currents, hysteresis, and bearing friction are not explicitly modeled.

These assumptions ensure that the model remains tractable for simulation and control design while still capturing the essential dynamics of the motor.

B. DC Motor Model Parameters

To ensure the simulation reflects real-world conditions, we define the parameters of a 0.5 HP, 6000 RPM DC motor. These parameters are chosen based on typical values for small industrial motors, which are commonly used in robotics, automation, and precision control applications. The selected parameters are as follows:

- Armature Resistance (Ra): 0.027 Ω
- Armature Inductance (La): 0.002 H
- Back EMF Constant (Ke): 0.0382 V·s/rad
- Torque Constant (Kt): 0.0382 N·m/A
- Moment of Inertia (J): 5×10^{-5} kg·m²
- Viscous Friction Coefficient (b): 5×10^{-3} N·m·s/rad

C. Simulating PID-Controlled DC Motor

To simulate the behavior of a PID-controlled DC motor, the mathematical model of the motor described in Section III is integrated with the PID control law. The simulation involves numerically solving the coupled electrical and mechanical equations while dynamically updating the control signal based on the error, integral term, and derivative term. Below is step by step instruction on the simulation is achieved using python programming language.

D. Key Steps in DC Motor Model Simulation

1. Initialization:
- Define initial conditions, including initial speed ω_0, set speed ω_{set}, load torque T_L, and PID gains K_p K_i K_d.
- Specify simulation parameters time step = $1*10^{-4}$ seconds and maximum simulation time of 1.5 seconds.

2. Dynamic Calculation of Error Terms:
- At each time step, compute the error e(t), integral term$\int e(t)$, dt, and derivative term $\frac{de(t)}{dt}$ using numerical methods.

3. Control Signal Computation:
- Apply the PID control law to calculate the applied voltage V(t).

4. Motor Dynamics Simulation:
- Update the motor's speed and current using the electrical and mechanical equations:

$$V(t) = R_a \cdot I(t) + L_a \cdot \frac{dI(t)}{dt} + E_b(t)\#(10)$$

$$J \cdot \frac{d\omega(t)}{dt} = T_m(t) - T_L(t) - b \cdot \omega(t)\#(11)$$

Where Eq. (10) and Eq. (11) are electrical and mechanical equations consecutively.

5. Performance Evaluation:

- Analyze the motor's response by calculating key performance metrics, as described in the next section.

E. Performance Metrics

The performance of the PID-controlled DC motor is evaluated using several key metrics derived from the step response of the system. These metrics provide quantitative insights into the controller's ability to regulate the motor's speed effectively. The equations for each metric are as follows:

1. Rise Time (tr): The time taken for the motor speed to reach 90% of the set speed.
2. Settling Time (ts):The time required for the motor speed to stabilize within ±2% of the set speed.
3. Overshoot (Mp): The maximum percentage deviation above the set speed during the transient phase.

$$M_p = \frac{max(\omega(t)) - \omega_{\text{set}}}{\omega_{\text{set}}} \cdot 100\backslash\%\#(12)$$

4. Steady-State Error (ess): The difference between the final speed and the set speed after the transient phase.

$$e_{ss} = \lim_{t\to\infty} |\omega(t) - \omega_{\text{set}}|\ \#(13)$$

5. Peak Time (tp): The time at which the motor speed reaches its first peak.
6. Damping Ratio (ζ): A measure of how oscillatory the system is. It is derived from the second-order system approximation.

$$\zeta = \frac{-\ln(M_p/100)}{\sqrt{\pi^2 + [\ln(M_p/100)]^2}}\#(14)$$

7. Natural Frequency (ωn): The frequency at which the system oscillates in the absence of damping.

$$\omega_n = \frac{\pi}{t_p\sqrt{1-\zeta^2}}\#(15)$$

8. Down Time: The duration during which the motor speed falls below the set speed before stabilizing.

$$\text{Down Time} = \int_{t_1}^{t_2} 1\, dt \quad \text{where } \omega(t) < \omega_{\text{set}}\#(16)$$

These metrics are calculated from the simulated data and used to assess the effectiveness of the PID controller under different operating conditions and gain settings.

F. Synthesis of Training Dataset

The synthesis of a high-quality training dataset is a cornerstone in developing machine learning models for controlling DC motors. This dataset serves as the foundation for training GRU model, enabling it to learn the intricate relationships between dynamic features (e.g., error, integral term, derivative term) and optimal PID parameters $(K_p), (K_i), (K_d)$.

A well-synthesized dataset ensures that the trained models generalize effectively to unseen scenarios, making them robust to variations in operating conditions. For a DC motor control system, the dataset must capture the relationship between:

- Input Features: Error e(t), integral term $\int e(t)\,dt$, and derivative term $\frac{de(t)}{dt}$.
- Output Targets: Optimal PID parameters$(K_p), (K_i), (K_d)$.

By synthesizing the dataset, we can simulate a wide range of realistic operating conditions, ensuring that the models are exposed to diverse scenarios during training. This approach enables the development of adaptive controllers capable of handling dynamic and uncertain environments and to ensure the dataset is representative of real-world applications, operating conditions are randomly sampled within predefined ranges. These conditions include:

1. Initial Speed ω_0:
- Randomly sampled within a range (500 – 3000 rpm).
- Represents the motor's starting speed before control is applied.
2. Set speed ω_{set}:
- Randomly sampled within a range (0 – 3000 rpm).
- Represents the desired speed that the controller aims to achieve.
3. Load Torque T_L:
- Randomly sampled within a range (0– 0.5 N·m).
- Represents external forces opposing the motor's motion, such as friction or mechanical loads.

These random samples ensure that the dataset captures a variety of scenarios, including low-speed, high-speed, no-load, and loaded conditions. The diversity of these operating conditions enhances the generalization capability of the trained models.

G. Optimization of PID Parameters Using Genetic Algorithm (GA)

For each set of operating conditions, the optimal PID parameters (K_p, K_I, K_D) are determined using a Genetic Algorithm (GA). GA is a heuristic optimization technique inspired by the process of natural selection. It mimics biological evolution by iteratively evolving a population of candidate solutions through processes such as selection, crossover[12], and mutation. In this context, each individual represents a set of PID parameters (K_p, K_I, K_D), and the fitness function evaluates the control performance based on metrics such as rise time, overshoot, and settling time. The process involves the following steps[13]:

1. Encoding of Candidate Solutions:
Each candidate solution represents a set of PID parameters (K_p, K_I, K_D), encoded as a chromosome. For example, a chromosome might be represented as:

$$\text{Chromosome} = [K_p, K_i, K_d] \#(17)$$

2. Initialization of Population:
A population of candidate solutions is initialized randomly within predefined bounds:

$$K_p \in [K_{p,\min}, K_{p,\max}], \quad K_i \in [K_{i,\min}, K_{i,\max}], \quad K_d \in [K_{d,\min}, K_{d,\max}] \#(18)$$

In this context the bound are chosen between (0, 10) for (K_p, K_I, K_D).

3. Fitness Function:
The fitness of each candidate solution is evaluated using an objective function that minimizes a weighted combination of performance metrics. Common metrics include rise time, settling time, overshoot, and steady-state error. For example:

$$\text{Objective Function} = w_1 \cdot \text{Rise Time} + w_2 \cdot \text{Settling Time} + w_3 \cdot |\text{Overshoot}| + w_4 \cdot |\text{Steady-State Error}| \#(19)$$

Where w_1, w_2, w_3, w_4 are weighting factors that prioritize specific metrics.

4. Selection:
Individuals with higher fitness scores (better performance) are selected for reproduction. Techniques such as roulette-wheel selection or tournament selection are commonly used.

5. Crossover:
Pairs of selected individuals exchange genetic material to produce offspring. For example, single-point or two-point crossover can be applied to combine portions of two parent chromosomes.

6. Mutation:
Random changes are introduced into the offspring to maintain genetic diversity and prevent premature convergence. Mutation ensures that the algorithm explores new regions of the search space.

7. Termination Criteria:
The algorithm terminates when a predefined number of generations is reached or when the fitness score converges to an acceptable level.
The output of the GA is the set of optimal PID parameters (K_p, K_I, K_D), for each set of operating conditions. These parameters are then paired with the corresponding dynamic features to form input-output pairs.

H. Construction of the Dataset

The synthesized dataset is constructed by pairing the extracted dynamic features (error, integral term, derivative term) with the optimized PID parameters (K_p, K_I, K_D),Since these features are time-series data, they are repeated for each time step during the simulation. The dataset is structured as follows:

1. Inputs (Features):
- Error e(t)
- Integral Term($\int e(t)\ dt$)
- Derivative Term($\frac{de(t)}{dt}$)
2. Outputs (Targets):
- Optimal PID parameters(K_p, K_I, K_D).

The dataset is normalized to improve the performance of machine learning models using standardization technique that scales features to have zero mean and unit variance as follows:

$$x_{\text{normalized}} = \frac{x - \mu}{\sigma} \#(20)$$

Where(μ) *and* $(\sigma)$are the mean and standard deviation of the feature.
Finally, the dataset is split into training and testing sets to evaluate the model's generalization performance. A common split ratio is 80% for training and 20% for testing.

I. GRU Model Design

The GRU architecture is designed to process time-series data, where each input sequence corresponds to a series of dynamic features. The configuration includes:

1. Input Layer: Accepts sequences of error, integral term, and derivative term.
2. Hidden Layer: Two stacked GRU layers with 64 and 32 units, respectively.
3. Dense Layer: A fully connected layer with three neurons and softplus activation for outputting (K_p, K_I, K_D).

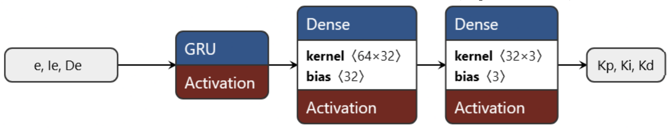

Fig. 4. GRU Architecture

J. Training Process

The GRU model is trained using the synthesized dataset described in Section (H). Key aspects of the training process include using Mean Squared Error (MSE) as a Loss function to measure the difference between predicted and actual PID parameters [14], as for the optimizer, Adam optimizer is employed for efficient gradient-based updates [15].
A batch size of 32 is selected to balance computational efficiency and convergence speed and the model is trained for 200 epochs, with early stopping implemented to prevent overfitting.
However, GRUs typically converge faster due to their simpler architecture, unlike LSTM.

VI. SIMULATION RESULTS

The simulation is based on a high-speed DC motor with parameters carefully selected to represent a small industrial motor suitable for dynamic applications. The key specifications of the motor include an armature resistance$R_a\ of\ 0.027\,\Omega$, armature inductance$\mathrm{L_a}$ of $0.002\ \mathrm{H}$, back EMF constant$\mathrm{K_e}$ and torque constant$(\mathrm{K_t})$of $0.0382\ \mathrm{V\cdot s/rad}$, and $0.0382\ \mathrm{N\cdot m/A}$ respectively, a moment of inertiaJ of $5\times10^{-5}\ \mathrm{kg\cdot m^2}$, and a viscous friction coefficient (b) of $5\times10^{-3}\ \mathrm{N\cdot m\cdot s/rad}$. These parameters define the motor's electrical and mechanical behavior during operation, The simulation time stepdt was set to $1\times10^{-4}\,\mathrm{s}$, with a maximum simulation duration of1.5sPID parameters were updated at regular intervals0.01sto ensure real-time adaptability.
assess the robustness and adaptability of the control strategies, three distinct trailer scenarios were simulated, each with varying target speeds, initial speeds, and load torques. The results are presented in tabular form for each trial, followed by step response figures that illustrate the performance of GRUmodel.

1. Trail 1 (Low Speed Operation): initial speed is set to 500 RPM, Target Speed of 1500 RPM and Load Torque of 0.2 N.m.

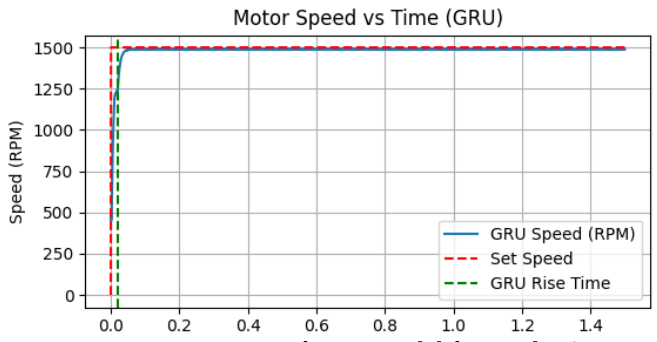

Fig. 5. Step Response of GRU model for Trial 1 (Low-Speed Operation)

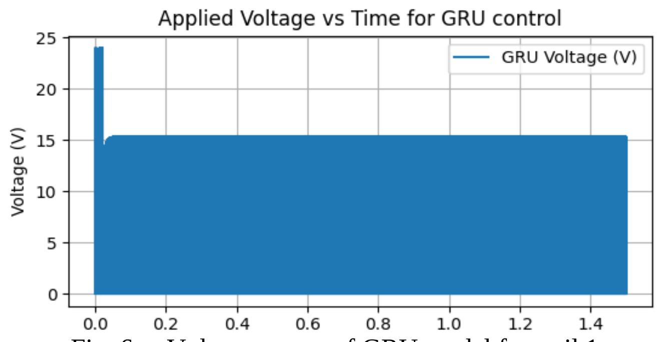

Fig. 6. Voltage output of GRU model for trail 1

2. Trail 2 (High Speed Operation): initial speed is set to 1000 RPM, Target Speed of 2000 RPM and Load Torque of 0.05 N.m.

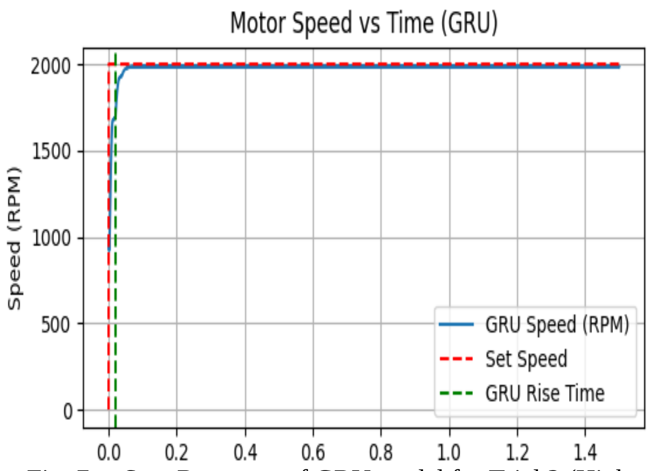

Fig. 7. Step Response of GRU model for Trial 2 (High-Speed Operation)

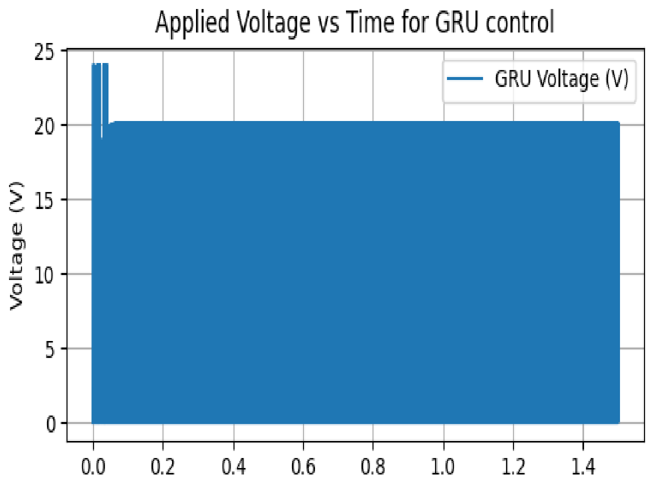

Fig. 8. Voltage output of GRU model for trail 2

3. Trail 3 (Variable Speed Operation): initial speed is set to 500 RPM, Target Speed start from 1500 RPM with periodic increment of 500 RPM until the set speed reaches 2500 RPM for every 1 second, then decreases speed by 400 RPM for every 1 second also, until it reaches a set speedof 1700 RPM, the entire simulation duration is 5 seconds and finally a variable load torque which is randomly set with every change in set speed between 0.05 and 0.5 N.m.

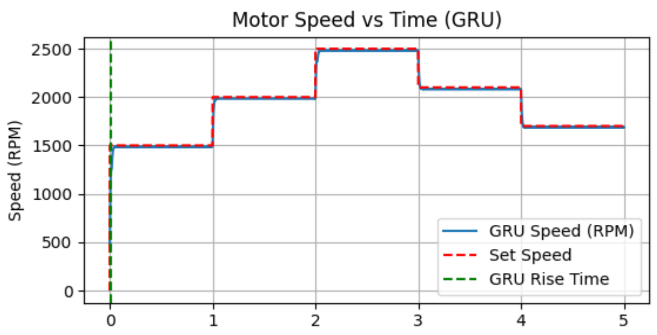

Fig. 9. Step Response of GRU model for Trial 3 (Variable-Speed Variable-Torque Operation)

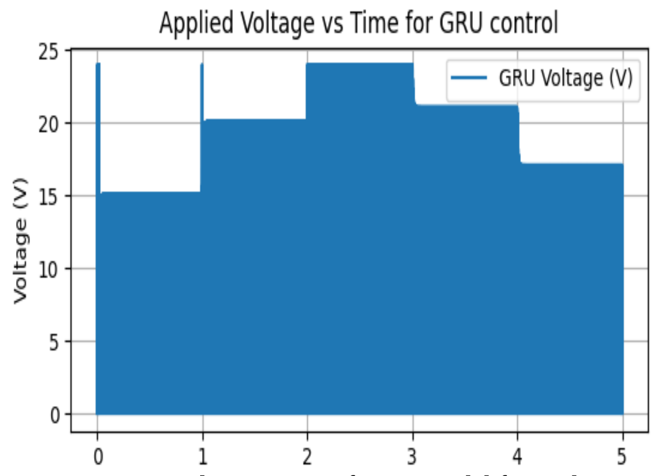

Fig. 10. Voltage output of GRU model for trail 3

Table -1 Experiment Result

GRU Metric	Trail 1	Trail 2	Trail 1
Rise Time (s):	0.0203	0.0191	0.0099
Settling Time (s):	0.0376	0.0444	0.0231
Down Time (s):	inf	inf	0.023
Overshoot (%):	-0.7775	-0.7642	-0.7831
Steady-State Error:	11.6631	16.0441	15.3626
Peak Time (s):	0.2506	0.2923	0.2494
Damping Ratio:	0.8397	0.8405	0.8393
Natural Frequency (rad/s):	23.0818	19.8386	23.2137

VII. CONCLUSION

The Paper demonstrated the Gated Recurrent Unit (GRU) model's efficacy as an advanced control strategy for DC motor speed regulation. GRU exhibited rapid rise and settling times, effectively dampening oscillations while displaying minimal overshoot. Although its steady-state error was not as low as FLC, GRU proved highly adaptable and robust, excelling at capturing temporal dependencies crucial for complex, high-speed dynamics and predicting optimal PID parameters. Its computational efficiency, balancing accuracy and reduced parameter count, positioned GRU as a viable alternative to LSTM for real-time applications. This paper contributes by highlighting GRU's potential in intelligent control systems and offering practical insights into its dynamic performance. Given its strengths, GRU is well-suited for dynamic industrial environments, including autonomous vehicles and industrial automation, balancing adaptability with computational cost. Future work should investigate hybrid11 GRU methodologies, real-time hardware implementation, advanced optimization techniques, robustness under uncertainty, generalization to various motor types, and broader applications, alongside further customization, scalability for multi-input/multi-output (MIMO) systems, and thorough validation for safety-critical scenarios.

VIII. REFERENCE

[1]. K. J. Åström and B. Wittenmark, 1995, Adaptive Control, 2nd ed. Reading, MA: Addison-Wesley, pp. 1-20.

[2]. N. Mohan, T. M. Undeland, and W. P. Robbins,2003, Power Electronics: Converters, Applications, and Design, 3rd ed. Hoboken, NJ: Wiley, ch. 8.

[3]. J. J. E. Slotine and W. Li, 1991, Applied Nonlinear Control. Englewood Cliffs, NJ: Prentice Hall, pp. 123-145.

[4]. R. C. Dorf and R. H. Bishop, 2017, Modern Control Systems, 13th ed. Upper Saddle River, NJ: Pearson, ch. 5.

[5]. Y. LeCun, Y. Bengio, and G. Hinton, May 2015, "Deep learning," Nature, vol. 521, no. 7553, pp. 436-444.

[6]. W. Fan et al.,Jul. 2024, "Bi-LSTM/GRU-based anomaly diagnosis for virtual network function instance," Comput. Netw., vol. 249, p. 110515, doi: 10.1016/j.comnet.2024.110515.

[7]. B. K. Bose, Feb. 2007, "Neural Network Applications in Power Electronics and Motor Drives—An Introduction and Perspective," IEEE Trans. Ind. Electron., vol. 54, no. 1, pp. 14–33, doi: 10.1109/TIE.2006.888683.

[8]. M. Azri and B. A. Mutalib,Mar. 15, 2025, "Speed Control of Dc Motor Using Pi Controller Mohd Azri bin Abd Mutalib,"2008. Accessed:[Online].Available:https://www.semanticscholar.org/paper/Speed-Control-of-Dc-Motor-Using-Pi-Controller-Mohd-Azri-Mutalib/223931aa76bdd865a54ae28ccc8f54af28c5ce8c

[9]. Tan Kiong Howe, May 2003, "Evaluation of the Transient Response of a DC motor using MATLAB/SIMULINK,".

[10]. Muhammed Nasiruddin Bin Mahyddin, Nov. 2005, "Direct model reference adaptive control of coupled tank liquid level control system,".

[11]. K. J. Åström, T. Hägglund, and K. J. Åström, 1995, PID controllers, 2nd ed. Research Triangle Park, N.C: International Society for Measurement and Control.

[12]. T. Ahmmed, I. Akhter, S. M. Rezaul Karim, and F. A. Sabbir Ahamed, Dec. 2020, "Genetic Algorithm Based PID Parameter Optimization," Am. J. Intell. Syst., vol. 10, no. 1, pp. 8–13, doi: 10.5923/j.ajis.20201001.02.

[13]. T. A. Faris Ku Yusoff, M. F. Atan, N. Abdul Rahman, S. F. Salleh, and N. A. Wahab, Jan. 1970, "Optimization of PID Tuning Using Genetic Algorithm," J. Appl. Sci. Process Eng., vol. 2, no. 2, doi: 10.33736/jaspe.168.2015.

[14]. I.Goodfellow, Y. Bengio, and A., 2016, Courville, Deep learning. in Adaptive computation and machine learning. Cambridge, Massachusetts: The MIT Press.

[15]. D. P. Kingma and J. Ba,2014, "Adam: A Method for StochasticOptimization,", arXiv. doi: 10.48550/ARXIV.1412.6980.