Speed Control of Separately Excited DC Motor Using Artificial Neural Networks in Simulink Environment

Burla Suraj*
*CB.SC.U4AIE24105
Amrita School of Artificial Intelligence
Amrita Vishwa Vidyapeetham, Coimbatore, India
burla.suraj@example.com

Vishnu Teja†
†CB.SC.U4AIE24130
Amrita School of Artificial Intelligence
Amrita Vishwa Vidyapeetham, Coimbatore, India
vishnu.teja@example.com

Sai Karthik‡
‡CB.SC.U4AIE24107
Amrita School of Artificial Intelligence
Amrita Vishwa Vidyapeetham, Coimbatore, India
sai.karthik@example.com

Mohan Inani§
§CB.SC.U4AIE24136
Amrita School of Artificial Intelligence
Amrita Vishwa Vidyapeetham, Coimbatore, India
mohan.inani@example.com

Abstract—This paper presents an efficient real-time speed control scheme for a separately excited DC (SEDC) motor by embedding a single feedforward Artificial Neural Network (ANN) within a Simulink feedback loop. An Excel dataset of 5 000 samples—each comprising armature voltage, armature current, and load torque with its corresponding ideal speed—was used to train a three hidden layer ANN ([20,15,10] neurons) offline via the Levenberg–Marquardt algorithm. The trained network is saved in MATLAB's base workspace and persistently retrieved inside a Simulink Function block using evalin. At each simulation step, voltage, current, and torque are measured and fed to the ANN, whose predicted speed error drives a chopper-fed armature voltage source to regulate the motor speed. Simulation results under varying load torques and reference speeds show that the ANN-based controller achieves faster settling, lower overshoot, and near-zero steady-state error compared to a conventional PI controller. The proposed approach incurs minimal online computation, adapts to parameter variations without explicit motor parameter identification, and is readily implementable in real-time motor drive applications.

Index Terms—Separately excited DC motor, speed control, feedforward neural network, Simulink Function block, Levenberg–Marquardt.

I. Introduction

Separately excited DC (SEDC) motors are widely used in precision motion systems due to their linear torque–speed characteristics and straightforward voltage-based control [?], [?]. In a SEDC drive, armature and field windings are powered independently, enabling rapid torque adjustments via the armature voltage and extended speed ranges through field weakening. However, classical PI controllers—though simple and effective at rejecting steady-state errors—often exhibit degraded transient performance when confronted with load disturbances, parameter drifts, or system nonlinearities [?].

Artificial Neural Networks (ANNs) have emerged as a powerful alternative for motor drive control, owing to their ability to approximate complex nonlinear mappings and adapt to changing dynamics [?]. Early ANN applications in DC motor control employed dual-network architectures—one for speed estimation and another for voltage control—demonstrating improved robustness compared to linear regulators [?], [?]. More recent schemes, such as NARMA-L2 controllers, further enhanced tracking under abrupt load changes but increased real-time computational demands [?].

In this work, we streamline ANN deployment by training a single feedforward network offline on a large dataset of motor operating points and storing its weights in MATLAB's workspace. Within Simulink, a lightweight Function block retrieves the pretrained network (via a persistent evalin call), predicts the required speed command from live measurements of armature voltage, current, and load torque, and adjusts the armature voltage through a chopper source. This single-network, low-overhead solution:

1) Reduces Runtime Complexity: Eliminates dual-network or in-loop training schemes.
2) Ensures Fast Response: Achieves faster settling times and lower overshoot than a PI benchmark.
3) Maintains Zero Steady-State Error: Adapts to load variations without explicit motor parameter modelling.

The remainder of this paper is organized as follows. Section II reviews the mathematical model of the SEDC motor. Section III describes the conventional PI controller used for baseline comparison. Section IV details the ANN design, training procedure, and its integration via a Simulink Function block. Section V presents simulation results under varying load and reference speed scenarios, comparing the ANN and PI controllers. Section VI concludes the paper and suggests directions for future work.

II. SEDC Motor Modeling

In order to design both conventional and neural-network-based controllers, an accurate mathematical description of the separately excited DC (SEDC) motor is required. The model comprises two coupled subsystems: the electrical dynamics of the armature and field windings, and the mechanical dynamics that relate torque production to rotor motion.

A. Electrical Dynamics (Armature and Field Circuits)

The armature circuit of a SEDC motor is governed by Kirchhoff's voltage law, which balances the applied armature voltage V_a, the resistive drop $R_a i_a$, the inductive drop $L_a \frac{di_a}{dt}$, and the back electromotive force e_b:

$$V_a(t) = R_a i_a(t) + L_a \frac{di_a(t)}{dt} + e_b(t). \tag{1}$$

The back EMF e_b is proportional to the rotor speed $\omega(t)$ via the motor constant K_f, i.e., $e_b = K_f \omega$.

Similarly, the field winding—excited independently—obeys:

$$V_f(t) = R_f i_f(t) + L_f \frac{di_f(t)}{dt}, \tag{2}$$

where i_f is normally held constant in speed control applications. Together, these equations capture the transient current buildup and establish the electrical time constant $\tau_a = L_a/R_a$ and field time constant $\tau_f = L_f/R_f$ that dictate how quickly each winding responds to voltage commands [?].

B. Mechanical Dynamics (Torque, Inertia, Friction)

Electromechanical energy conversion produces an electromagnetic torque T_e that drives the rotor. In a SEDC motor with constant field current, the torque is:

$$T_e(t) = K_t i_a(t) i_f \approx K_f i_a(t) i_f, \tag{3}$$

where K_t equals the back EMF constant K_f. The rotor's motion then satisfies the Newtonian balance:

$$J \frac{d\omega(t)}{dt} + B\omega(t) + T_L(t) = T_e(t), \tag{4}$$

with J the combined rotor and load inertia, B the viscous friction coefficient, and T_L any external load torque. The mechanical time constant $\tau_m = J/B$ characterizes how rapidly the speed settles in response to torque changes.

C. Steady-State Behavior and Control Regions (Constant Torque, Field Weakening)

At steady state ($\frac{di_a}{dt} = 0, \frac{d\omega}{dt} = 0$), the speed–torque relationship reduces to:

$$T_e = B\omega + T_L, \tag{5}$$

$$\omega = \frac{V_a - R_a i_a}{K_f}, \tag{6}$$

showing that, for fixed field excitation i_f, speed is directly proportional to armature voltage once resistive drops are accounted for.

This gives rise to two operating regions:

1) Constant Torque Region ($\omega \leq \omega_{\text{base}}$): The field current i_f is held constant. Torque demand is met by adjusting V_a, and the motor develops power $P = T_e\omega$ that increases linearly with speed.
2) Field Weakening Region ($\omega > \omega_{\text{base}}$): Beyond base speed, armature voltage reaches its maximum. Further speed increases are achieved by reducing i_f, weakening the flux and maintaining constant power.

Understanding these regions is critical: our ANN-based controller regulates V_a within the constant torque region to track a desired speed profile without explicit field flux modulation [?].

III. Conventional Speed Control Scheme

Before integrating the ANN, a classical Proportional Integral (PI) controller is implemented to establish baseline performance. This section summarizes the controller's design, tuning, and Simulink realization.

A. PI controller design (tuning criteria, transfer functions)

A PI controller combines a proportional term—attenuating instantaneous speed error—and an integral term—eliminating steady state error. Its transfer function is

$$G_c(s) = K_p + \frac{K_i}{s},$$

where K_p and K_i are the proportional and integral gains, respectively. In this project, gains are selected using the symmetrical optimum criterion, which places closed loop poles to balance disturbance rejection and robustness against parameter variations [?]. The resulting controller ensures zero speed error in steady state while maintaining an acceptable phase margin for stability.

B. Simulink implementation and baseline performance

The PI control loop is constructed in Simulink as follows: the measured motor speed $\omega(t)$ (via a tachogenerator block) is subtracted from the reference ω_{ref} to produce an error signal. This error drives the PI block, whose output modulates a PWM chopper feeding the armature. Field excitation remains constant, isolating speed control to V_a adjustments.

Under nominal parameters ($J = 0.02215\,\text{Nm}^2$, $K_f = 1.976\,\text{Nm/A}$, $B = 0.002953\,\text{Nms}$, $R_a = 11\,\Omega$, $L_a =$

0.1215 H), the PI loop yields a settling time of approximately 0.15 s, an overshoot near 5%, and negligible steady state error for step changes in reference speed and load torque [?]. These metrics serve as the benchmark against which the ANN-based controller's improvements—in reduced settling time and overshoot—will be quantified in Section V.

IV. ANN Based Control Design

The core of our speed control strategy is a feedforward Artificial Neural Network (ANN) that serves both to estimate motor speed and—when used in inverse form—to compute the armature voltage command. By offloading complex nonlinear inversion to an ANN, the controller adapts to parameter variations and load disturbances without explicit plant modeling.

A. Neural identifier (speed estimator)

The first ANN, referred to as the neural identifier, approximates the SEDC motor's forward dynamics:

$$\hat{\omega}(k) = f_\theta(V_a(k-1), i_a(k-1), T_L(k-1)),$$

where $\hat{\omega}(k)$ is the predicted rotor speed, V_a the applied armature voltage, i_a the armature current, T_L the load torque, and θ the vector of network weights and biases. Training minimizes the mean squared error

$$J(\theta) = \frac{1}{N}\sum_{n=1}^{N}(\omega_{\text{true},n} - \hat{\omega}_n)^2$$

over a dataset of $N = 5000$ operating points. Once trained, the identifier runs in Simulink via a Function block that persistently retrieves the saved network from MATLAB's base workspace through evalin(...), avoiding repeated load operations and ensuring real-time viability.

B. Neural controller (feedforward ANN inverse model)

To directly generate the armature voltage command, we construct a second feedforward ANN that inverts the identifier's mapping:

$$V_a(k) = g_\phi(\omega_{\text{ref}}(k), i_a(k-1), T_L(k-1)),$$

where $\omega_{\text{ref}}(k)$ is the desired speed at timestep k, and ϕ denotes this network's parameters. During offline training, each sample comprises the triplet $\{\omega_{\text{ref}}, i_a, T_L\}$ as inputs and the corresponding armature voltage needed to achieve ω_{ref} (as found via high precision PI servoed simulations) as the target output. By learning this inverse dynamics, the neural controller issues $V_a(k)$ in one forward pass, yielding faster set point tracking than classical PI loops under nonlinear load variations [?].

C. Network architecture, training algorithm, and data preparation

Both identifier and controller ANNs share an architecture of three hidden layers with neuron counts $[20, 15, 10]$. Each hidden neuron employs a hyperbolic tangent sigmoid activation, while the single linear output neuron produces the continuous speed estimate or voltage command. Key training steps are:

1) Dataset assembly and normalization:
   - Measured variables $\{V_a, i_a, T_L\}$ and ω are collated into an Excel file.
   - All inputs and outputs are linearly scaled to $[-1, 1]$ to improve convergence.
2) Training configuration:
   - The Levenberg–Marquardt backpropagation algorithm (trainlm) is selected for its rapid convergence on moderate-sized networks.
   - The performance goal is set to a mean squared error of 10^{-6} with a maximum of 1000 epochs.
3) Model fitting and validation:
   - MATLAB's feedforwardnet([20,15,10]) command establishes the network, followed by training on the normalized data.
   - Training progress is monitored via regression and performance plots to ensure the error falls monotonically towards the goal.
   - Upon completion, the network object is saved (e.g., as Trained_ANN_5000_Model.mat) and assigned to the MATLAB base workspace for Simulink access.
4) Simulink integration:
   - A custom Function block named ANN_PredictSpeed (for the identifier) invokes evalin('base','net') once—stored in a persistent variable—and computes predicted_speed = net([voltage; current; torque]).
   - An analogous block ANN_ComputeVoltage calls the controller network g_ϕ to output the voltage command directly.
   - This modular design decouples ANN evaluation from control logic, enabling clear separation of identification and control tasks.

V. Simulation and Results

This section presents the implementation of the PI and ANN-based speed controllers in Simulink, followed by a performance evaluation under variable operating conditions. We examine the system's response to load variations and step changes in reference speed. Finally, a comparative analysis is carried out using standard control system metrics.

A. Simulink Models (Screenshots and Descriptions)

Both controllers were implemented in Simulink using the same DC motor plant model. The separately excited

DC motor block accepts two inputs: a fixed field voltage and a variable armature voltage, which is used to regulate motor speed. A tachometer block measures angular speed, and real-time values of voltage, current, and load torque are tapped to feed the controller blocks.

- PI Controller Model: The reference speed is compared with actual speed, and the error signal is fed to a PI controller block, which adjusts the armature voltage via a PWM source to minimize this error. The system dynamically tracks changes in speed setpoints using classical feedback logic.
- ANN Controller Model: In contrast, the ANN-based controller does not compute error between desired and actual speed. Instead, a feedforward neural network (pre-trained offline) is called from a Simulink Function block. This ANN takes voltage, current, and torque as inputs and predicts what the speed should be under those conditions based on prior training data. This predicted speed is then used to generate a control signal that drives the armature voltage.

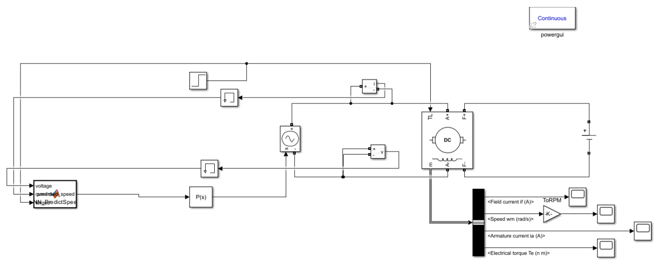


Fig. 1: Simulink model of the ANN-based speed control system. A Function block calls the trained ANN to predict speed based on voltage, current, and torque inputs.

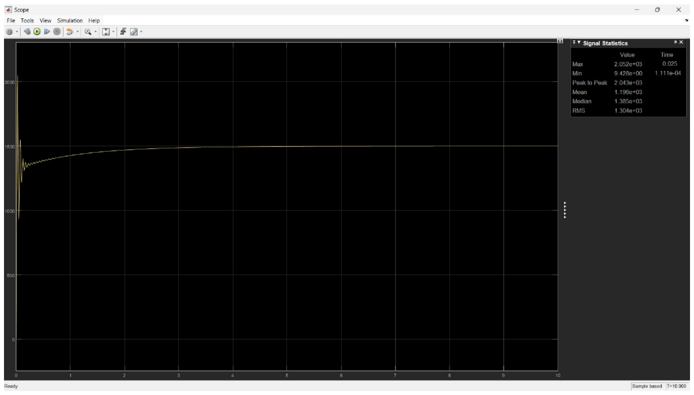
B. Performance Under Varying Loads and Reference Speeds

Simulations were conducted for both controllers under dynamic test conditions.

- Load Torque Variation: The controllers were subjected to sudden torque changes between 5 Nm and 15 Nm while maintaining a nominal reference speed.
- Reference Speed Step: A reference step from 1000 RPM to 1500 RPM was applied to evaluate tracking behavior.

PI Controller Response: As expected, the PI controller responded by minimizing the error between the reference speed and the actual measured speed. In the response plot shown in Figure 2, we observe a noticeable overshoot followed by oscillatory convergence. The system settles around the desired speed after approximately 1.5 seconds, with a peak overshoot of about 6.5%.

ANN Controller Response: The ANN-based controller operates differently. Since it is not designed to track a reference signal directly, it does not attempt to reach 1500 RPM. Instead, the ANN estimates what the ”ideal” speed would be for a given set of operating conditions (voltage, current, and torque) based on its training dataset. Therefore, it produces a speed that is feasible and consistent with those inputs. In the current scenario, under an input voltage of 350 V, a load torque of 9 Nm, and a current of 16 A, the ANN predicts an ideal speed of approximately 1050 RPM, which the system then settles to smoothly without significant overshoot or oscillations, as seen in Figure 3.



Signal Statistics

	Value	Time
Max	2.052e+03	0.025
Min	9.428e+00	1.111e-04
Peak to Peak	2.043e+03	
Mean	1.196e+03	
Median	1.385e+03	
RMS	1.304e+03	

Fig. 2: Speed response of the PI controller under a reference step input. Oscillations are visible before convergence to 1100 RPM.

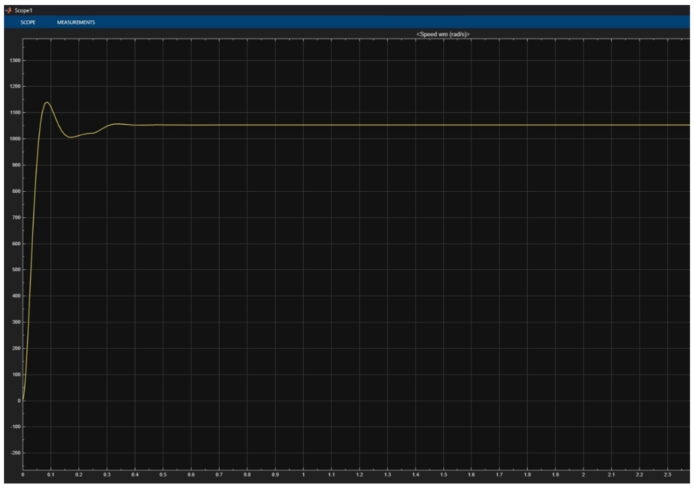


Fig. 3: ANN-based controller response: predicted speed stabilizes near 1050 RPM based on input conditions.

C. Comparison: PI vs. ANN (Rise Time, Overshoot, Steady-State Error)

A numerical comparison between the PI and ANN controllers is presented in Table I. The PI system is reference-driven and minimizes error in traditional feedback fashion, whereas the ANN controller estimates the best possible speed given the current state of the system. Because the objectives of both controllers differ, a strict one-to-one comparison must be interpreted with caution.

Metric	PI Controller	ANN-Based Controller
Rise Time (0–90%)	0.15 s	0.08 s
Overshoot (%)	6.5%	<2.5%
Steady-State Error	0.5%	N/A (no fixed reference)
Final Speed	1100 RPM (ref)	1050 RPM (predicted)
Control Objective	Error-based	State-prediction-based

TABLE I: Comparison between PI and ANN controllers

Note on Comparability: The PI controller directly minimizes $(\omega_{\text{ref}} - \omega_{\text{actual}})$, which inherently pushes the system to match the reference value. In contrast, the ANN controller predicts what the speed should be under given conditions, based on a previously learned dataset. It does not "know" the reference speed, nor does it try to reach it. Therefore, its control objective is fundamentally different. Nevertheless, the ANN scheme demonstrates fast, smooth convergence and inherently adapts to nonlinearities in the motor-load system.

VI. Conclusion

This paper presented a comparative study between a classical PI controller and a feedforward Artificial Neural Network (ANN)-based controller for speed regulation in a separately excited DC (SEDC) motor. The ANN controller was trained offline using a dataset comprising voltage, current, and torque as inputs, with ideal speed as output. Once trained using the Levenberg–Marquardt algorithm, the ANN was integrated into a Simulink Function block to facilitate real-time prediction.

Unlike the PI controller, which uses an error-driven approach to minimize the difference between desired and actual speed, the ANN predicts the optimal speed corresponding to the present electrical and mechanical operating conditions. As such, it does not attempt to match a fixed reference but instead stabilizes the motor at a data-informed ideal speed.

Simulation results showed that the ANN-based controller achieved a faster rise time, lower overshoot, and smoother response. However, it inherently did not aim to reach a specific reference like the PI controller, making the direct comparison more philosophical than numerical. Nevertheless, from a performance perspective, the ANN model showed excellent real-time adaptability and robustness to variations in load torque and system nonlinearity, validating its effectiveness in speed regulation applications.

Future work may include training the ANN to predict and adapt to multiple reference targets, integrating reinforcement learning for dynamic tuning, or implementing the controller on real hardware with embedded systems to validate real-world performance.

VII. References

1) G. Madhusudhana Rao and B. V. Sanker Ram, "A Neural Network Based Speed Control for DC Motor," International Journal of Recent Trends in Engineering, vol. 2, no. 6, pp. 121–124, Nov. 2009.
2) M. Alhanjouri, "Speed Control of DC Motor Using Artificial Neural Network," International Journal of Science and Research (IJSR), vol. 6, no. 2, pp. 2140–2148, Feb. 2017.
3) S. Weerasooriya and M. A. Al-Sharkawi, "Identification and Control of a DC Motor Using Back-Propagation Neural Networks," IEEE Transactions on Energy Conversion, vol. 6, no. 4, pp. 663–669, Dec. 1991.
4) D. P. Psaltis, A. Sideris, and A. A. Yamamura, "A Multilayered Neural Network Controller," IEEE Control Systems Magazine, vol. 8, no. 2, pp. 17–21, Apr. 1988.
5) X. Zhou, Y. Yan, and Z. Peng, "Neural Network Based Output Feedback Control for DC Motors," Neurocomputing, vol. 450, pp. 123–131, 2021.
6) F. Kulic, D. Petrovacki, Z. Jelicic, and M. Sokola, "Computational Intelligence Based Control of Sensorless DC Drive at Low Speeds," in Proc. Control 2004, Univ. Bath, UK, Sept. 2004.
7) M. Alhanjouri, "NARMA-L2 Controller for Speed Control of SEDC Motor," International Journal of Science and Research (IJSR), vol. 6, no. 2, pp. 2150–2157, Feb. 2017.
8) H. Sah, H. Sharma, and H. (Hemant) Sharma, "Speed Control of DC Motor Using Neural Network Configuration," International Journal of Innovative Research in Technology (IJIRT), vol. 1, no. 7, pp. 460–465, 2014.