Research Square

Preprints are preliminary reports that have not undergone peer review.
They should not be considered conclusive, used to inform clinical practice,
or referenced by the media as validated information.

Robust Field-Oriented Control of Hybrid Stepper Motors Using a DRNN-Based Adaptive PID Controller

Mike O. Ejunka

mikeejunka@unical.edu.ng

University of Calabar

Patrick Odu

University of Calabar

Victor O. Ekpang

University of Calabar

Research Article

Keywords: Hybrid stepper motor, field-oriented control, self-tuning PID controller; dynamic recurrent neural network, adaptive control, parameter uncertainty, load disturbance rejection, neural-network-based control

Posted Date: November 12th, 2026

DOI: https://doi.org/10.21203/rs.3.rs-11543001/v1

License: This work is licensed under a Creative Commons Attribution 4.0 International License.
Read Full License

Additional Declarations: No competing interests reported.

Robust Field-Oriented Control of Hybrid Stepper Motors Using a DRNN-Based Adaptive PID Controller

Mike O. Ejunka [a], Patrick Odu [b], Victor O. Ekpang [c]

[a] Department of Computer Engineering, University of Calabar, Calabar, Nigeria - *mikeejunka@unical.edu.ng*
[b] Department of Electrical and Electronics Engineering, University of Calabar, Nigeria
[c] Department of Computer Engineering, University of Calabar, Nigeria

Abstract

Hybrid Stepper Motors (HSMs) are widely used in high-precision motion applications, including computer numerical control machines, robotics, three-dimensional printing systems, and aerospace positioning equipment, owing to their high torque density, excellent holding torque, and favourable cost-to-performance ratio. However, closed-loop operation under Field-Oriented Control (FOC) remains vulnerable to performance degradation caused by electrical parameter variations, mechanical uncertainties, and sudden load torque disturbances, which conventional fixed-gain Proportional–Integral–Derivative (PID) controllers cannot adequately compensate for. This paper proposes a Dynamic Recurrent Neural Network (DRNN)-based self-tuning PID controller for robust field-oriented control of hybrid stepper motors operating under combined parameter uncertainty and load disturbances. We first develop a complete nonlinear dynamic model of the HSM in the direct–quadrature synchronous reference frame, incorporating detent torque, back-electromotive-force coupling, and bounded parameter perturbations. A diagonal recurrent neural network with self-feedback hidden neurons is then designed to adaptively tune the outer speed-loop PID gains in real time, using an online gradient-descent learning algorithm with momentum derived from a chain-rule-based sensitivity analysis of the plant Jacobian. The proposed controller is integrated into a cascaded FOC architecture with zero d-axis current control and Space Vector Pulse Width Modulation, and is extensively evaluated in MATLAB/Simulink under five test scenarios: nominal speed tracking, speed reversal, sudden load disturbance, combined parameter uncertainty, and low-speed operation. Comparative results against a conventional fixed-gain PID controller, assessed using transient-response indices and integral error metrics, demonstrate the proposed scheme's superior tracking accuracy, disturbance rejection, and robustness.

Keywords - Hybrid stepper motor, field-oriented control, self-tuning PID controller; dynamic recurrent neural network, adaptive control, parameter uncertainty, load disturbance rejection, neural-network-based control.

INTRODUCTION

Hybrid stepper motors (HSMs) are indispensable actuators in modern precision-motion applications, including computer numerical control (CNC) machines, 3D printing systems, robotic manipulators, semiconductor manufacturing equipment, and aerospace positioning systems. [5] Thanks to their cost-effective manufacturing process, HSMs are popular for position control in low-power industrial applications. They are typically chosen for their high torque density at low speeds, excellent holding torque, and relatively low cost compared with equivalent servo motor systems. Structurally, an HSM combines the operating principles of permanent-magnet and variable-reluctance stepper motors, benefiting from both high step resolution and strong electromagnetic torque production [2] (permanent magnet stepper motor, the variable reluctance stepper motor, and the hybrid stepper motor). Despite these advantages, however, conventional open-loop operation has several well-documented drawbacks. [4] Commonly used microstepping control suffers from low efficiency, poor dynamic performance, vibration, and variation in maximum continuous torque depending on the operating point. Furthermore, [6] this open-loop approach fails when the torque load exceeds the motor's capacity. As a result, the motor and driver must be oversized, reducing efficiency and increasing cost. Continuously applying rated current regardless of actual load demand also increases thermal stress and wastes energy [6] (this results in oversized motors and waste of power and heat). These limitations have driven a significant shift toward closed-loop control strategies, particularly Field-Oriented Control (FOC), which treats the HSM as a two-phase permanent magnet synchronous machine and enables decoupled control of flux and torque-producing current components in the rotating reference frame [5] (the model of an HSM used for control design is the same as a surface-mounted PM synchronous machine, with additional non-idealities). [7] In servomotors and HSMs, phase currents are not directly linked to torque. Instead, Park's transformation mathematically decouples field and torque. It calculates two current components from the phase currents: "d," or "direct," in the field direction, and "q," or "quadrature," in the torque-forming direction. By controlling the torque-forming current component i_q, the machine's torque can be regulated directly, as in a DC machine. Experimental studies have further confirmed that [4] FOC supplies only the requested current, resulting in lower HSM temperatures, less insulation degradation, and reduced permanent-magnet demagnetization. They also show that [9] FOC, especially when combined with field-weakening techniques, outperforms traditional microstepping in efficiency, torque consistency, and thermal management.

Mathematical Modelling of the Hybrid Stepper Motor

2.1 Structural Description and Modelling

The Hybrid Stepper Motor (HSM) considered here is a two-phase bipolar machine. Its rotor comprises an axially magnetised permanent magnet formed from two concentric cylinders, with teeth on top. This structure gives the motor its characteristic high step resolution and strong detent torque [1]. The two stator windings, phase A and phase B, are physically separated by a 90° mechanical displacement. As a result, the measured phase currents inherently form an orthogonal stationary reference frame, without requiring a Clarke transformation [2], [3]. From a modelling perspective, a typical hybrid stepper motor can be treated as a two-phase Permanent Magnet Synchronous Motor (PMSM) with a high number of pole pairs. This allows the same dynamic model used for a surface-mounted PM synchronous machine to be applied, with certain non-idealities added, such as detent torque caused by the toothed rotor structure [3], [2].

To obtain a tractable dynamic model suitable for the design of field-oriented controllers, the following standard assumptions are adopted in this work [4], [5]:

The magnetic circuit is linear, and stator iron saturation is neglected.

The permeance space harmonics of the hybrid rotor are ignored, yielding a model that is simple and accurate compared with conventional detailed models [5].

The two stator phase windings are identical and sinusoidally distributed, so that the induced back-electromotive force (back-EMF) in each phase is purely sinusoidal.

The eddy current and hysteresis losses in the stator core are negligible.

The direct- and quadrature-axis inductances are considered equal (i.e., $L_d = L_q = L_0$), which is consistent with the surface-mounted PM assumption commonly used for HSM control design [1], [6].

2.2 Dynamic Model in the Phase (a–b) Reference Frame

Under these assumptions, Kirchhoff's voltage law applied to each phase yields a pair of coupled first-order differential equations, with the back-EMF varying sinusoidally with the electrical rotor angle. The complete nonlinear state-space model of the two-phase HSM in the fixed-phase reference frame can therefore be expressed as [7], [3]:

$$\frac{di_a}{dt} = \frac{1}{L}[v_a - Ri_a + K_m\omega\sin(N_r\theta)] \qquad (1)$$

$$\frac{di_b}{dt} = \frac{1}{L}[v_b - Ri_b - K_m\omega\cos(N_r\theta)] \qquad (2)$$

$$\frac{d\omega}{dt} = \frac{1}{J}[-K_m i_a\sin(N_r\theta) + K_m i_b\cos(N_r\theta) - B\omega - T_L - K_D\sin(4N_r\theta)] \qquad (3)$$

$$\frac{d\theta}{dt} = \omega \qquad (4)$$

where v_a, v_b and i_a, i_b denote the voltages and currents of phases A and B, respectively; ω is the mechanical rotor angular speed; θ is the mechanical rotor angular position; R and L are the resistance and self-inductance of each phase winding; N_r is the number of rotor teeth; K_m and K_D are the motor torque constant and the detent torque constant, respectively; B is the viscous friction coefficient; J is the rotor moment of inertia; and T_L represents the external load torque, which is treated as an unknown bounded disturbance [7].

Equations (1)-(4) exhibit three fundamental features that complicate the design of linear controllers: (i) strong nonlinear coupling between the electrical variables (i_a, i_b) and the mechanical state (θ) through sinusoidal functions of the electrical angle $N_r\theta$; (ii) a position-dependent detent torque term $K_D\sin(4N_r\theta)$ that originates from the interaction between the permanent magnet flux and the rotor-slot geometry; and (iii) an exogenous load torque T_L that acts as a matched disturbance on the mechanical equation. These three features justify adopting a rotating reference frame transformation, which linearises the back-EMF coupling and enables decoupled flux and torque control [6].

2.3 Park Transformation to the Rotating (d–q) Reference Frame

In electrical machine modelling, the Park transformation projects vector quantities defined in the fixed stator frame onto a new direct–quadrature frame that rotates synchronously with the fictitious excitation vector, yielding a linear relationship between the control variables and the electromagnetic torque [6]. For a two-phase HSM, since the stator windings are already physically orthogonal, the transformation reduces to a single rotation matrix applied directly to the phase quantities [3], [2]: rotation matrix applied directly to the phase quantities [3], [2]:

$$\begin{bmatrix} x_d \\ x_q \end{bmatrix} = \begin{bmatrix} \cos(N_r\theta) & \sin(N_r\theta) \\ -\sin(N_r\theta) & \cos(N_r\theta) \end{bmatrix} \begin{bmatrix} x_a \\ x_b \end{bmatrix} \qquad (5)$$

where $x \in \{v, i\}$ represents either voltage or current, and the electrical angle is defined as $\theta_e = N_r\theta$. The corresponding inverse Park transformation, used to reconstruct the phase voltages from the controller outputs in the d–q frame, is given by [8]:

$$\begin{bmatrix} v_a \\ v_b \end{bmatrix} = \begin{bmatrix} \cos(N_r\theta) & -\sin(N_r\theta) \\ \sin(N_r\theta) & \cos(N_r\theta) \end{bmatrix} \begin{bmatrix} v_d \\ v_q \end{bmatrix} \qquad (6)$$

2.4 Dynamic Model in the (d–q) Rotating Reference Frame

Applying the Park transformation (5) to the phase-frame model (1)-(4) and exploiting the assumption $L_d = L_q = L_0$ yields the dynamic model of the hybrid stepper motor in the synchronous (d, q) reference frame, which is the standard form adopted for field-oriented control design [4], [6]:

$$u_d = L_0 \frac{di_d}{dt} + R_s i_d - N_r \omega L_0 i_q \qquad (7)$$

$$u_q = L_0 \frac{di_q}{dt} + R_s i_q + N_r \omega L_0 i_d + k_M \omega \qquad (8)$$

$$\frac{d\theta}{dt} = \omega \qquad (9)$$

$$J \frac{d\omega}{dt} + B\omega = T_e - T_L - T_c \qquad (10)$$

where the electromagnetic torque and the detent (cogging) torque are defined as [4]:

$$T_e = k_M i_q \qquad (11) \qquad T_c = -N_r \sum_{j=1}^{\infty} K_{Dj} \sin(4jN_r\theta) \qquad (12)$$

In the rotating frame, (i_d, i_q) and (u_d, u_q) denote the stator currents and voltages projected onto the direct and quadrature axes, respectively; k_M is the back-EMF/torque constant referred to the q-axis; and $(\theta, \omega, i_d, i_q)$ constitute the state variables, while (u_d, u_q) are the control inputs and $T_L$is the unknown disturbance input [4].

The physical interpretation of models (7) - (10) is central to the design of the proposed controller. The d-axis current i_d corresponds to the field-producing component, while the q-axis current i_q corresponds to the torque-producing component [9]. The two cross-coupling terms $-N_r\omega L_0 i_q$in (7) and $+N_r\omega L_0 i_d$in (8) represent the speed-dependent electromagnetic coupling between the two axes, while the term $k_M\omega$in (8) corresponds to the rotational back-EMF induced on the quadrature axis. These coupling terms are typically compensated by means of feed-forward decoupling within the inner current control loops, so that the resulting closed-loop dynamics in the d–q frame become those of two independent first-order linear systems [8].

2.5 Field-Oriented Control Formulation

The field-oriented control strategy adopted in this work is based on the principle that for a surface-mounted PM machine, the maximum torque per ampere condition is achieved when the direct-axis current reference is set to zero, i.e., $i_d = 0$, so that the entire stator current contributes to torque production via the $q - axis\ torque\ equation$ [8], [10]. Under this strategy, the torque equation (11) becomes a linear algebraic relation between i_q and T_e, which allows the outer speed controller to generate the torque command simply (ig an appropriate quadrature current reference:

$$i_q^*(t) = \frac{1}{k_M} u_{\text{sp}}(t) \qquad (13)$$

where $u_{\text{sp}}(t)$ is the output of the outer speed-loop controller — in the present work, the proposed DRNN-based self-tuning PID regulator. The commanded d–q voltages (v_d^*, v_q^*) generated by the inner current loops are subsequently converted back to the phase voltages (v_a^*, v_b^*) via the inverse Park transformation (6), and finally applied to the motor through a dual H-bridge inverter driven by Space Vector Pulse Width Modulation (SVPWM) [10], [2].

2.6 Incorporation of Parameter Uncertainty and Load Disturbances

For robustness evaluation, the nominal model (7)-(10) is augmented to account for parameter uncertainty and external disturbances explicitly. Let R_{s0}, $L_{0,0}$, k_{M0}, J_0, and B_0 denote the nominal values of the electrical and mechanical parameters. The actual parameters during operation are then expressed as:

$$R_s = R_{s0} + \Delta R_s,\ L_0 = L_{0,0} + \Delta L_0,\ k_M = k_{M0} + \Delta k_M \qquad (14)$$

$$J = J_0 + \Delta J,\ B = B_0 + \Delta B \qquad (15)$$

where $\Delta(\cdot)$ denotes the bounded deviation of each parameter from its nominal value. The uncertainty in R_s arises mainly from winding temperature rise, the uncertainty in L_0 and k_M reflects magnetic saturation and manufacturing tolerances, and the uncertainty in J and B captures payload changes and mechanical wear. The load torque T_L is treated as a time-varying, bounded, but otherwise unknown external disturbance [4]. Substituting (14)-(15) into (7)-(10) produces a perturbed model whose closed-loop stability and tracking performance must be preserved by the proposed adaptive controller - a requirement that will be verified through the simulation scenarios presented in Section 5

3. Field-Oriented Control Strategy and Baseline PID Controller

3.1 Overview of the Field-Oriented Control Framework

Field-Oriented Control (FOC), also called vector control, is a high-performance control strategy originally developed for induction and permanent magnet synchronous motors and later extended to hybrid stepper motors because the HSM is structurally similar to a two-phase permanent magnet synchronous machine [1], [2]. The fundamental idea underlying FOC is to decouple the stator current vector into two orthogonal components in a rotor-flux-aligned rotating reference frame: a direct-axis component. i_d responsible for flux production, and a quadrature-axis component i_q responsible for torque production [3]. This decomposition transforms the inherently nonlinear and coupled multivariable dynamics of the AC machine into a form that resembles the linear and independent flux–torque relationship of a separately excited DC motor, thereby enabling the use of classical linear controllers to achieve high dynamic performance [4], [5].

In contrast to the conventional microstepping technique, in which both stator phases are continuously excited with sinusoidal currents regardless of the actual torque demand, the FOC strategy supplies only the current required to meet the instantaneous torque reference, which translates into significantly higher efficiency, reduced copper losses, lower operating temperatures, and improved torque consistency across the full speed range [6], [7]. Moreover, the FOC framework prevents the loss of synchronism that is characteristic of open-loop stepper operation, since the closed-loop current and speed regulators automatically compensate for abrupt load variations [1].

3.2 Cascaded Control Architecture

The complete FOC scheme adopted in this work follows the standard cascaded control architecture commonly employed in high-performance AC drives [3], [8]. The architecture consists of three nested control loops, arranged in order of increasing bandwidth from the outer to the inner loop:

Outer speed loop: Generates the quadrature-axis current reference i_q^* from the speed tracking error $e_\omega = \omega^* - \omega$. In the baseline configuration, this loop uses a conventional PID controller; in the proposed scheme (Section 4), we replace it with a DRNN-based self-tuning PID controller.

Inner q-axis current loop: Regulates the quadrature-axis current i_q to track its reference i_q^* by generating the quadrature voltage command v_q^*. This loop is implemented using a standard PI controller.

Inner d-axis current loop: Regulates the direct-axis current $i_d$to track its reference $i_d^* = 0$(zero d-axis current control) by generating the direct voltage command v_d^*. This loop is also implemented using a standard PI controller.

The cascaded structure is particularly suitable for electric drives because it naturally separates the slow mechanical dynamics (speed) from the fast electrical dynamics (currents), simplifying tuning and improving overall closed-loop stability [8]. The complete signal flow of the FOC scheme is as follows:

$$\omega^* \rightarrow \text{Speed Controller} \rightarrow i_q^* \rightarrow \text{Current Controllers} \rightarrow (v_d^*, v_q^*) \rightarrow \text{Inverse Park} \rightarrow (v_a^*, v_b^*) \rightarrow \text{SVPWM} \rightarrow \text{Inverter} \rightarrow \text{HSM}$$

Simultaneously, the measured phase currents are projected onto the frame using the Park transformation (5) and fed back to the inner current controllers. In contrast, the rotor position and speed, obtained from an encoder or a sensorless estimator, are fed back to the speed controller and to the Park/inverse Park transformations [9], [10]. estimator, are fed back to the speed controller and to the Park/inverse Park transformations [9], [10].

3.3 Zero d-Axis Current Control Strategy

For surface-mounted permanent magnet machines and hybrid stepper motors, $L_d = L_q$the electromagnetic torque is a linear function of the quadrature current only, as expressed in (11). Consequently, the Maximum Torque Per Ampere (MTPA) condition is achieved by setting the direct-axis current reference to zero [3], [4]:

$$i_d^* = 0 \qquad (16)$$

This strategy, commonly referred to as $i_d = 0$ control or zero-start-axis current control, ensures that the entire stator current magnitude is devoted to torque production, which minimises copper losses for a given

torque demand and simplifies the torque command generation to a direct scaling of the speed controller output [11]. Furthermore, by keeping i_d close to the design value, the design avoids unintentional field weakening or permanent-magnet demagnetisation during normal operation [10].

3.4 Inner Current Loop Design with Decoupling Compensation

As shown in the dq-axis voltage equations (7) and (8), the direct and quadrature axes are mutually coupled through the speed-dependent terms. − $L_0 i_q$ n $+N_r \omega L_0 i_d$, as we -EMF m $k_M \omega$. To achieve independent regulation of i_d and i_q, feed-forward decoupling compensation is introduced within the inner current loops [12], [8]. The decoupled voltage commands are expressed as:

$$v_d^* = v_d' - N_r \omega L_0 i_q \qquad (17)$$

$$v_q^* = v_q' + N_r \omega L_0 i_d + k_M \omega \qquad (18)$$

where v_d' and v_q' are the outputs of the current PI controllers acting on the current tracking errors $e_d = i_d^* - i_d$ and $e_q = i_q^* - i_q$, respectively. Substituting (17)-(18) into (7)-(8) cancels the cross-coupling terms and yields two decoupled first-order linear systems:

$$L_0 \frac{di_d}{dt} + R_s i_d = v_d' \qquad (19) \qquad L_0 \frac{di_q}{dt} + R_s i_q = v_q' \qquad (20)$$

For each decoupled axis, a standard PI controller is designed in the continuous time domain as [13]:

$$v_d'(t) = K_{p,d}\, e_d(t) + K_{i,d} \int_0^t e_d(\tau)\, d\tau \qquad (21)$$

$$v_q'(t) = K_{p,q}\, e_q(t) + K_{i,q} \int_0^t e_q(\tau)\, d\tau \qquad (22)$$

The current controller gains $(K_{p,d}, K_{i,d})$ and $(K_{p,q}, K_{i,q})$ are tuned using the pole–zero cancellation (internal model control) technique, in which the controller zero is placed at the plant pole $-R_s/L_0$, yielding a closed-loop first-order response with a bandwidth $\omega_{c,i}$selected approximately one decade below the switching frequency of the inverter to ensure a sufficient stability margin [8], [12].

3.5 Space Vector Pulse Width Modulation

The decoupled voltage commands (v_d^*, v_q^*) generated by the inner current loops are transformed back to the stationary phase frame by means of the inverse Park transformation (6), yielding the phase voltage references (v_a^*, v_b^*). The dual H-bridge inverter synthesises these references using Space Vector Pulse Width Modulation (SVPWM), which, compared with conventional sinusoidal PWM, offers higher DC-bus utilisation, lower total harmonic distortion of the phase currents, and reduced switching losses [14], [15]. The SVPWM strategy adopted in this work operates with a fixed switching frequency. f_{sw} selected to balance the trade-off between electromagnetic torque ripple and inverter switching losses.

3.6 Baseline Outer-Loop PID Speed Controller

In the baseline configuration, a conventional discrete-time PID controller controls the outer speed loop; it is the de facto industrial standard because of its simplicity, well-understood behaviour, and systematic

tuning procedures [16], [17]. The incremental (velocity) form of the PID controller is adopted here because it inherently prevents integral wind-up during actuator saturation and yields a smoother control signal during reference transitions [18]. The incremental PID control law is expressed as:

$$u_{\text{sp}}(k) = u_{\text{sp}}(k-1) + K_p\,\Delta e_\omega(k) + K_i\,e_\omega(k) + K_d\,\Delta^2 e_\omega(k) \qquad (23)$$

where:

$e_\omega(k) = \omega^*(k) - \omega(k)$ is the speed tracking error at the sampling instant. k,

$\Delta e_\omega(k) = e_\omega(k) - e_\omega(k-1)$ is the first difference of the error,

$\Delta^2 e_\omega(k) = e_\omega(k) - 2e_\omega(k-1) + e_\omega(k-2)$ is the second difference of the error,

K_p, K_i, and K_d are the proportional, integral, and derivative gains, respectively, and

$u_{\text{sp}}(k)$ is the speed controller output.

The quadrature-axis current reference is then obtained by scaling the controller output through the inverse of the torque constant, as given in (13):

$$i_q^*(k) = \frac{1}{k_M}\,u_{\text{sp}}(k) \qquad (24)$$

The baseline PID gains (K_p, K_i, K_d) are tuned offline using the Ziegler–Nichols's frequency-response method under nominal operating conditions, followed by fine manual adjustment to achieve a satisfactory trade-off between rise time, overshoot, and disturbance rejection at the nominal operating point [17].

3.7 Limitations of the Fixed-Gain PID Controller

Although the baseline PID controller provides acceptable performance under nominal conditions, its fixed-gain structure imposes several fundamental limitations when applied to the hybrid stepper motor operating under realistic conditions [19], [20]:

Sensitivity to parameter variations. The PID gains are tuned based on the nominal values of R_s, L_0, k_M, J, and B. When these parameters deviate from their nominal values due to temperature rise, magnetic saturation, payload changes, or mechanical wear (as modelled in equations (14) - (15), the closed-loop poles drift, which degrades the transient response and may even compromise stability [19].

Poor disturbance rejection. The fixed gains represent a compromise between tracking performance and disturbance rejection. With sudden load-torque disturbances, the fixed-gain PID controller shows a pronounced speed dip and a prolonged recovery time [20].

Trade-off between overshoot and settling time. Increasing the proportional gain reduces rise time but increases overshoot, whereas increasing the integral gain eliminates steady-state error at the expense of transient oscillations. A fixed set of gains cannot simultaneously optimise all transient-response indices across different operating conditions [16].

Inability to handle nonlinearities. The PID controller is intrinsically linear, whereas the hybrid stepper motor exhibits nonlinear dynamics due to the detent torque, magnetic saturation, and friction nonlinearities. These nonlinearities cannot be effectively compensated by a linear fixed-gain controller [20].

These limitations strongly motivate the development of an intelligent, self-adapting controller that continuously adjusts the PID gains online in response to the motor's instantaneous operating conditions. The design and implementation of such a controller - a Dynamic Recurrent Neural Network-based self-tuning PID regulator - are presented in detail in Section 4.

4. Methodology: Proposed DRNN-Based Self-Tuning PID Controller

4.1 Rationale and Overall Control Scheme

The limitations of the fixed-gain PID controller identified in Section 3.7 - namely, sensitivity to parameter uncertainty, poor disturbance rejection, inability to handle nonlinearities, and the fundamental compromise between transient-response indices - motivate the adoption of an intelligent, self-adapting control structure capable of modifying its parameters online in response to the plant's instantaneous operating condition [1], [2]. Among the various adaptive control paradigms reported in the literature, neural-network-based self-tuning PID controllers have proven particularly attractive because they combine the structural simplicity and interpretability of the classical PID law with the universal approximation capability and self-learning ability of artificial neural networks [3], [4].

Within the broader category of neural networks, Dynamic Recurrent Neural Networks (DRNNs) are particularly valuable for real-time control tasks. Unlike feedforward networks, which provide only a static input-output mapping, DRNNs incorporate feedback loops that confer short-term memory and the ability to model temporal dependencies [5], [6]. This recurrent architecture is especially effective for electromechanical systems such as the hybrid stepper motor, which exhibit time-dependent behaviour due to factors including magnetic saturation, thermal changes in winding resistance, varying payloads, and sudden load-torque shocks [7]. Additionally, the diagonal recurrent neural network (DRNN), introduced by Ku and Lee [5], has a much lower computational load than fully recurrent networks, making it ideal for embedded platforms such as digital signal processors (DSPs) or field-programmable gate arrays (FPGAs) commonly used in motor control applications [8].

Figure 1 illustrates the overall control scheme proposed in this work. The DRNN-based self-tuning PID controller replaces the fixed-gain PID regulator of the outer speed loop described in Section 3.6, while the inner current loops and the SVPWM-driven inverter remain unchanged. At every sampling instant, the DRNN receives a vector of system variables (the current and past speed errors and the previous control signal), computes the three adaptive PID gains. $\left(K_p,(k),K_i(k)K_d(k)\right)$ through a forward pass, and feeds them to an incremental PID control law that generates the quadrature-axis current reference $i_q^*(k)$. Simultaneously, the network weights are updated online by means of a gradient-descent algorithm with momentum, which minimises a cost function defined over the speed tracking error [3], [9].

4.2 Architecture of the Proposed Diagonal Recurrent Neural Network

The DRNN used in this work follows the diagonal recurrent architecture originally proposed by Ku and Lee [5], which consists of three layers: an input layer, a hidden layer with self-recurrent neurons, and an output layer. The distinguishing feature of the diagonal recurrent architecture is that each hidden neuron has a single self-feedback connection, in contrast to the full inter-neuron feedback characteristic of Elman or fully recurrent networks. This design choice preserves the dynamic representation capability of recurrent networks while drastically reducing the number of trainable weights and simplifying gradient computation [5], [6].

4.2.1 Input Layer

The input layer consists of three neurons that receive a vector of system variables carrying both error and control information:

$$\boldsymbol{X}(k) = \begin{bmatrix} x_1(k) \\ x_2(k) \\ x_3(k) \end{bmatrix} = \begin{bmatrix} e_\omega(k) \\ e_\omega(k-1) \\ u_{\text{sp}}(k-1) \end{bmatrix} \qquad (25)$$

where $e_\omega(k) = \omega^*(k) - \omega(k)$ is the speed tracking error at the sampling instant. k, $e_\omega(k-1)$is its one-step delayed value, and $u_{\text{sp}}(k-1)$ is the previous output of the speed controller. Including the delayed error and the previous control signal gives the network information about the time evolution of tracking performance, complementing the intrinsic memory of the recurrent hidden layer [9].

4.2.2 Hidden Layer with Self-Recurrent Neurons

The hidden layer is composed of N_h neurons (typically $N_h = 5$ to 8, selected through empirical trial as discussed in Section 4.6). The net input to the j-th hidden neuron is given by:

$$S_j(k) = \sum_{i=1}^{3} W_{ij}^I \; x_i(k) + W_j^D \; H_j(k-1), \; j = 1,2,\dots,N_h \qquad (26)$$

where W_{ij}^I denotes the input-to-hidden weight connecting the i-th input neuron to the j-th hidden neuron, and W_j^D denotes the diagonal (self-recurrent) weight of the j-th hidden neuron. The output of the j-th hidden neuron is then obtained by applying a hyperbolic tangent activation function, which provides a smooth, differentiable, and zero-centred nonlinearity:

$$H_j(k) = \tanh\ (S_j(k)) = \frac{e^{S_j(k)} - e^{-S_j(k)}}{e^{S_j(k)} + e^{-S_j(k)}} \qquad (27)$$

4.2.3 Output Layer

The output layer consists of three neurons corresponding to the three adaptive PID gains. The net inputs to the output neurons are computed as the weighted sums of the hidden neuron outputs:

$$O_l(k) = \sum_{j=1}^{N_h} W_{jl}^O \; H_j(k), \; l = 1,2,3 \qquad (28)$$

where W_{jl}^O denotes the hidden-to-output weight connecting the j-th hidden neuron to the l-th output neuron. Since the PID gains must remain strictly positive to preserve closed-loop stability, a bounded positive activation function is applied at the output layer [4], [10]:

$$K_p(k) = K_p^{\max} \cdot \sigma\,(O_1(k)) \qquad (29)$$

$$K_i(k) = K_i^{\max} \cdot \sigma\,(O_2(k)) \qquad (30)$$

$$K_d(k) = K_d^{\max} \cdot \sigma\,(O_3(k)) \qquad (31)$$

where $\sigma(\cdot)$ denotes the logistic sigmoid function:

$$\sigma(z) = \frac{1}{1 + e^{-z}} \qquad (32)$$

and $K_p^{\max}$, $K_i^{\max}$, and $K_d^{\max}$ are user-defined upper bounds that constrain the adaptive gains within a physically meaningful range, thereby preventing unbounded growth during the transient learning phase [10], [11].

4.3 Incremental PID Control Law

The adaptive PID gains generated by the DRNN are embedded within an incremental (velocity-form) PID controller, which is adopted here for the same reasons discussed in Section 3.6: inherent prevention of integral windup, smoother control signals during reference transitions, and natural compatibility with the online gain-adaptation mechanism [12]. The incremental PID control law is expressed as:

$$u_{\text{sp}}(k) = u_{\text{sp}}(k-1) + K_p(k)\,\Delta e_\omega(k) + K_i(k)\,e_\omega(k) + K_d(k)\,\Delta^2 e_\omega(k) \qquad (33)$$

where:

$$\Delta e_\omega(k) = e_\omega(k) - e_\omega(k-1) \qquad (34)$$

$$\Delta^2 e_\omega(k) = e_\omega(k) - 2e_\omega(k-1) + e_\omega(k-2) \qquad (35)$$

The quadrature-axis current reference is then obtained by scaling the controller output through the inverse of the torque constant, as in (24):

$$i_q\,(k) = \frac{1}{k_M}\,u_{\text{sp}}(k) \qquad (36)$$

This current reference is subsequently fed to the inner. q-axis current loop described in Section 3.4, completing the cascade.

4.4 Online Learning Algorithm

4.4.1 Cost Function

The DRNN weights are updated online at every sampling instant to minimise the instantaneous squared speed-tracking error [3], [13]: $E(k) = \frac{1}{2}\,e_\omega^2(k) = \frac{1}{2}\left(\omega^*(k) - \omega(k)\right)^2$ (37)

This cost function provides a simple yet effective learning criterion that directly aligns the training objective with the control objective of accurate speed tracking.

4.4.2 Weight Update Rule with Momentum

The weights are updated using the gradient-descent algorithm with momentum, which accelerates convergence and smooths the trajectory of the weights in the parameter space [14]:

$$W(k+1) = W(k) - \eta \frac{\partial E(k)}{\partial W(k)} + \alpha \Delta W(k-1) \qquad (38)$$

where W generically denotes any trainable weight of the network (W_{ij}^{I}, W_{j}^{D}, or W_{jl}^{O}), $\eta \in (0,1)$ is the learning rate, $\alpha \in [0,1)$ is the momentum coefficient, and $\Delta W(k-1) = W(k) - W(k-1)$ is the previous weight increment.

4.4.3 Gradient Computation via the Chain Rule

The gradient of the cost function with respect to each weight is computed by applying the chain rule along the signal flow path error → speed → control signal → PID gains → network weights [3], [9]:

$$\frac{\partial E(k)}{\partial W(k)} = \frac{\partial E(k)}{\partial e_{\omega}(k)} \cdot \frac{\partial e_{\omega}(k)}{\partial \omega(k)} \cdot \frac{\partial \omega(k)}{\partial u_{\text{sp}}(k)} \cdot \frac{\partial u_{\text{sp}}(k)}{\partial K_l(k)} \cdot \frac{\partial K_l(k)}{\partial W(k)} \qquad (39)$$

The individual terms in (39) are evaluated as follows:

(i) Error term:

$$\frac{\partial E(k)}{\partial e_{\omega}(k)} = e_{\omega}(k), \ \frac{\partial e_{\omega}(k)}{\partial \omega(k)} = -1 \qquad (40)$$

(ii) Plant Jacobian: The partial derivative $\partial \omega(k)/\partial u_{\text{sp}}(k)$ represents the sensitivity of the plant output to the control input. Since the exact mathematical model of the plant is uncertain due to parameter variations and unmodeled nonlinearities, this term is approximated by the sign function, which is a widely accepted strategy in neural-network-based adaptive control [3], [4]:

$$\frac{\partial \omega(k)}{\partial u_{\text{sp}}(k)} \approx \text{sgn}\left(\frac{\omega(k) - \omega(k-1)}{u_{\text{sp}}(k) - u_{\text{sp}}(k-1)}\right) \qquad (41)$$

Any error introduced by this approximation is compensated by appropriately adjusting the learning rate. η, as rigorously analysed in [3].

(iii) Control-law sensitivities: Differentiating the incremental PID law (33) with respect to each gain yields:

$$\frac{\partial u_{\text{sp}}(k)}{\partial K_p(k)} = \Delta e_{\omega}(k), \ \frac{\partial u_{\text{sp}}(k)}{\partial K_i(k)} = e_{\omega}(k), \ \frac{\partial u_{\text{sp}}(k)}{\partial K_d(k)} = \Delta^2 e_{\omega}(k) \qquad (42)$$

(iv) Network sensitivities: The partial derivatives of the PID gains with respect to the network weights are obtained by backpropagating through the output and hidden layers of the DRNN. For the output-layer weights:

$$\frac{\partial K_l(k)}{\partial W_{jl}^O} = K_l^{\max} \cdot \sigma'\left(O_l(k)\right) \cdot H_j(k) \qquad (43)$$

where $\sigma'(z) = \sigma(z)[1 - \sigma(z)]$ is the derivative of the sigmoid function. For the input-layer weights:

$$\frac{\partial K_l(k)}{\partial W_{ij}^I} = K_l^{\max} \cdot \sigma'\left(O_l(k)\right) \cdot W_{jl}^O \cdot \left(1 - H_j^2(k)\right) \cdot x_i(k) \qquad (44)$$

For the diagonal recurrent weights, the self-feedback structure yields an ordered derivative that must be computed recursively [5]:

$$\frac{\partial K_l(k)}{\partial W_j^D} = K_l^{\max} \cdot \sigma'\left(O_l(k)\right) \cdot W_{jl}^O \cdot P_j(k) \qquad (45)$$

$$P_j(k) = \left(1 - H_j^2(k)\right)\left[H_j(k-1) + W_j^D\, P_j(k-1)\right] \qquad (46)$$

Equation (46) captures the dynamic nature of the diagonal recurrent weight: its influence on the current hidden state depends not only on the current input but also on all past states via the recursive term. This recursive sensitivity, originally derived in [5], is the key mathematical mechanism enabling the DRNN to adapt to time-varying plant dynamics. This recursive sensitivity is the key mathematical mechanism enabling the DRNN to adapt to time-varying plant dynamics.

4.4.4 Weight Update Equations

Combining (38) - (46), the explicit update laws for the three sets of weights become:

$$W_{jl}^O(k+1) = W_{jl}^O(k) + \eta\, \delta_l(k)\, H_j(k) + \alpha\, \Delta W_{jl}^O(k-1) \qquad (47)$$

$$W_{ij}^I(k+1) = W_{ij}^I(k) + \eta\, \delta_j^H(k)\, x_i(k) + \alpha\, \Delta W_{ij}^I(k-1) \qquad (48)$$

$$W_j^D(k+1) = W_j^D(k) + \eta\, \delta_j^H(k)\, P_j(k) + \alpha\, \Delta W_j^D(k-1) \qquad (49)$$

where the output and hidden error signals $\delta_l(k)$ and $\delta_j^H(k)$ are defined by:

$$\delta_l(k) = e_\omega(k) \cdot \operatorname{sgn}\left(\frac{\Delta\omega(k)}{\Delta u_{\text{sp}}(k)}\right) \cdot \frac{\partial u_{\text{sp}}(k)}{\partial K_l(k)} \cdot K_l^{\max} \cdot \sigma'\left(O_l(k)\right) \qquad (50)$$

$$\delta_j^H(k) = \left(1 - H_j^2(k)\right) \sum_{l=1}^{3} \delta_l\,(k) \cdot W_{jl}^O(k) \qquad (51)$$

4.5 Stability Considerations

The convergence and closed-loop stability of the proposed DRNN-based self-tuning PID controller depend critically on the learning rate and the momentum coefficient. $\eta$and the momentum coefficient α. Based on the discrete-time Lyapunov analysis presented in [15] for diagonal recurrent neural networks, a sufficient condition for convergence of the online learning algorithm is:

$$0 < \eta < \frac{2}{\lambda_{\max}} \qquad (52)$$

where $\lambda_{\max}$ denotes the largest eigenvalue of the matrix $\boldsymbol{J}^T\boldsymbol{J}$, with $\boldsymbol{J} = \partial\omega/\partial\boldsymbol{W}$ representing the Jacobian of the plant output with respect to the network weights. In practical implementations, since the exact value of $\lambda_{\max}$ is not known a priori, a conservative upper bound for the learning rate is adopted and refined through empirical tuning. Additionally, the bounded output activation functions (29)-(31) guarantee that the adaptive PID gains remain within a predefined safe operating range at all times, which provides an additional structural safeguard against instability during the transient learning phase [10], [11].

4.6 Initialisation and Hyperparameter Selection

Careful initialisation of the network weights is essential to ensure fast, reliable convergence of the online learning algorithm. In this work, the following initialisation strategy, which is widely adopted in the neural-adaptive-control literature, is employed [4], [9]:

The input and output weights (W_{ij}^I and W_{jl}^O) are initialised to small random values uniformly distributed in the interval $[-, 0.5\ 0.5]$ to break symmetry while avoiding saturation of the activation functions.

The diagonal recurrent weights (W_j^D) are initialised to small positive values in the interval $[0, 0.5]$ to ensure contractive dynamics of the hidden layer at the start of training.

The upper bounds on the adaptive PID gains ($K_p^{\max}$, $K_i^{\max}$, $K_d^{\max}$) are set to approximately twice the values obtained for the baseline fixed-gain PID controller through Ziegler–Nichols tuning (Section 3.6), which provides sufficient adaptation margin while preserving stability.

The learning rate η is initially set to $0.1$and the momentum coefficient $\alpha$to 0.05, with final values selected through sensitivity analysis to achieve the best trade-off between convergence speed and smoothness of the adaptation trajectory.

The number of hidden neurons N_h is selected empirically through trial-and-error between 5 and 8, which has been shown in previous studies to provide a sufficient approximation capability for second- and third-order electromechanical plants without inducing overfitting [4], [8].

4.7 Algorithmic Summary

The complete real-time operation of the proposed DRNN-based self-tuning PID controller at each sampling instant k is summarised in Algorithm 1.

Algorithm 1 - DRNN-Based Self-Tuning PID Controller. Initialise: Weights $W_{ij}^I, W_j^D, W_{jl}^O$; gain bounds $K_p^{\max}, K_i^{\max}, K_d^{\max}$; learning parameters η, α; previous states $H_j(0), P_j(0), e_\omega(-1), e_\omega(-2), u_{\text{sp}}(-1)$.

For each sampling instant $k = 1,2,3,$ …do:

Measurement: Acquire the measured mechanical speed $\omega(k)$the measured phase currents $i_a(k), i_b(k)$, and the measured rotor position $\theta(k)$.

Error computation: Compute $e_\omega(k) = \omega^*(k) - \omega(k)$, $\Delta e_\omega(k)$, and $\Delta^2 e_\omega(k)$ using (34) and (35).

Input vector assembly: Form the input vector $\boldsymbol{X}(k)$ using (25).

Forward pass (hidden layer): Compute $S_j(k)$and $H_j(k)$for $j = 1, \dots, N_h$ using (26) and (27).

Forward pass (output layer): Compute $O_l(k)$for $l = 1,2,3$ using (28), and derive the adaptive PID gains $K_p(k), K_i(k), K_d(k)$ using (29)-(31).

Control law: Compute the speed-controller output $u_{\text{sp}}(k)$using the incremental PID law (33), and derive the quadrature-current reference $i_q^*(k)$using (36).

Inner-loop update: Pass $i_q^*(k)$ and $i_d^* = 0$ to the inner current controllers, which generate $v_d^*(k)$ and $v_q^*(k)$ with decoupling compensation (17)-(18).

PWM generation: Apply the inverse Park transformation (6) and the SVPWM strategy to produce the switching signals for the inverter.

Backward pass: Compute the output and hidden error signals $\delta_l(k)$and $\delta_j^H(k)$ using (50) – (51), and update the ordered derivative $P_j(k)$using (46).

Weight update: Update the network weights W_{jl}^O, W_{ij}^I, and W_j^Dusing (47) – (49).

State storage: Store $e_\omega(k)$, $e_\omega(k-1)$, $u_{\text{sp}}(k)$, $H_j(k)$, and $P_j(k)$ for use at the next sampling instant.

This closed-loop online adaptation procedure continuously adjusts the PID gains in response to the hybrid stepper motor's evolving operating conditions, providing superior tracking performance, enhanced disturbance rejection, and strong robustness to parameter uncertainty, properties that the simulation study in Section 5 will rigorously verify.

5. Simulation Setup, Test Scenarios, and Comparative Results

5.1 Simulation Environment and Implementation

We implemented and evaluated the proposed DRNN-based self-tuning PID controller and the baseline fixed-gain PID controller in MATLAB/Simulink R2024b, using the Simscape Electrical toolbox to model the hybrid stepper motor, dual H-bridge inverter, and associated power electronic components [1]. We implemented the complete field-oriented control architecture described in Section 3 using discrete-time blocks with a fixed sampling period corresponding to a control-loop frequency representative of modern DSP-based motor-drive implementations [2], [3]. The Space Vector Pulse Width Modulation (SVPWM) scheme operates at a switching frequency of $f_{sw} = 20$ kHz, and the inverter's DC-bus voltage is set to $V_{dc} = 48$ V.

We implemented the DRNN-based self-tuning PID controller in a MATLAB Function block in Simulink, allowing direct translation of the forward and backward passes derived in Section 4 into executable code for hardware-in-the-loop (HIL) or embedded deployment [4]. All simulations were conducted on a workstation with an Intel Core i7 processor and 16 GB of RAM, using the fixed-step ode4 (Runge–Kutta)

solver, which balances numerical accuracy and computational efficiency for stiff electromechanical systems [1].

5.2 Hybrid Stepper Motor Parameters

The simulation study employs a commercially representative hybrid stepper motor model, a two-phase bipolar machine, whose nominal electrical and mechanical parameters are summarised in Table I. These parameter values are consistent with those reported in previous experimental studies of hybrid stepper motor drives operating under field-oriented control [5], [6], [7].

Table I - Nominal Parameters of the Hybrid Stepper Motor

Parameter	Symbol	Value	Unit
Phase resistance	R_s	1.4	Ω
Phase inductance	L_0	3.0	mH
Torque / Back-EMF constant	k_M	0.47	N·m/A
Number of rotor teeth	N_r	50	—
Rotor moment of inertia	J	4.5×10^{-5}	kg·m²
Viscous friction coefficient	B	8.0×10^{-4}	N·m·s/rad
Detent torque constant	K_D	0.05	N·m
Rated current (per phase)	I_N	3.0	A
Rated torque	T_N	1.2	N·m
Rated speed	ω_N	1000	rpm Rpm
DC-bus voltage	V_{dc}	48	V
Switching frequency	f_{sw}	20	kHz
Sampling period	T_s	50	μsMs

5.3 Controller parameters

5.3.1 Baseline PID Controller

We tuned the baseline fixed-gain outer-loop PID controller gains using the Ziegler–Nichols frequency-response method [8], then manually fine-tuned them around the nominal operating point to balance rise time, overshoot, and disturbance rejection. The resulting gains are:

$$K_p = 0.85,\ K_i = 12.0,\ K_d = 0.0025 \qquad (53)$$

The inner-loop current PI controllers were tuned using the internal model control technique described in Section 3.4, yielding:

$$K_{p,d} = K_{p,q} = 6.0,\ K_{i,d} = K_{i,q} = 2800 \qquad (54)$$

5.3.2 DRNN-Based Self-Tuning PID Controller

Table II summarises the architecture and hyperparameter settings of the proposed DRNN-based self-tuning PID controller. We selected the hyperparameters through a sensitivity analysis under nominal operating conditions, following the initialisation strategy described in Section 4.6.

Table II — DRNN-PID Controller Design Parameters

Parameter	Symbol	Value
Number of input neurons		3
Number of hidden neurons	N_h	6
Number of output neurons		3
Learning rate	η	0.10
Momentum coefficient	α	0.05
Output activation function		Sigmoid
Upper bound of K_p	$K_p^{\max}$	1.70
Upper bound of K_i	$K_i^{\max}$	24.0
Upper bound of K_d	$K_d^{\max}$	0.005
Initial weight range (input/output)		$[-, 0.5\ 0.5]$
Initial weight range (recurrent)		$[0, 0.5]$

5.4 Test Scenarios

To rigorously assess the performance and robustness of the proposed controller, five distinct test scenarios were designed, each targeting a specific aspect of the controller behaviour commonly encountered in practical motor-drive applications [5], [9], [10]:

5.4.1 Scenario A - Nominal Speed Tracking

The motor is commanded to track a reference speed step from standstill to the rated speed of 1000 rpm under no-load conditions and nominal parameter values. This scenario evaluates the basic transient response characteristics of both controllers, including rise time, overshoot, settling time, and steady-state error.

5.4.2 Scenario B - Speed Reversal

The motor is commanded to execute a speed reversal from $+500$ rpm to -500 rpm at $t = 1$ sunder no-load conditions and nominal parameters. This scenario evaluates the ability of the controllers to handle large bidirectional reference transitions, which are typical of robotic and CNC motion profiles [10].

5.4.3 Scenario C - Sudden Load Torque Disturbance

The motor operates at its rated speed of 1000 rpm, and a step load torque equal to the rated torque ($T_L = T_N = 1.2$ N\cdotpm) is suddenly applied at $t = 2$ sand removed at $t = 4$ s. This scenario evaluates both controllers' disturbance-rejection capability, quantified by the speed-dip magnitude and recovery time [11].

5.4.4 Scenario D - Parameter Uncertainty

The motor operates at 500 rpm under a constant load of 0.6 N\cdotpm, while the following parameter variations are introduced to emulate realistic operating uncertainties:

$\Delta R_s = +50\%$(temperature rise in the windings),

$\Delta L_0 = -20\%$(magnetic saturation),

$\Delta J = +100\%$(payload attachment),

$\Delta B = +50\%$(lubrication ageing).

This scenario rigorously evaluates the robustness of both controllers against combined electrical and mechanical parameter uncertainties [12].

5.4.5 Scenario E - Low-Speed Operation and Torque Ripple

The motor is commanded to track a low-speed reference of 30 rpm under no-load conditions. This scenario evaluates the ability of the controllers to suppress the torque ripple caused by the detent torque and the dead-time effects of the inverter, which are particularly pronounced at low speeds [6], [13].

5.5 Performance Evaluation Indices

To ensure an objective and quantitative comparison between the baseline PID and the proposed DRNN-PID controllers, the following performance indices are evaluated for each test scenario [8], [14]:

5.5.1 Transient Response Indices

Rise time (t_r): The time required for the speed to rise from 10 % to 90 % of its final value.

Settling time (t_s): The time required for the speed to remain within a $\pm 2\%$ band of its final value.

Maximum overshoot (M_p): The maximum percentage deviation of the speed from its final value during the transient.

Steady-state error (e_{ss}): The absolute speed error after the transient has decayed.

5.5.2 Integral Error Indices

Integral Absolute Error (IAE):

$$\text{IAE} = \int_0^{T_f} | e_\omega(t) | \, dt \qquad (55)$$

Integral Squared Error (ISE):

$$\text{ISE} = \int_0^{T_f} e_\omega^2 (t) \, dt \qquad (56)$$

Integral Time Absolute Error (ITAE):

$$\text{ITAE} = \int_0^{T_f} t \ | e_\omega(t) | \, dt \qquad (57)$$

5.5.3 Disturbance Rejection Indices

Speed dip ($\Delta\omega_{\text{dip}}$): The maximum instantaneous drop in speed following the application of the load torque disturbance.

Recovery time (t_{rec}): The time required for the speed to return to within $\pm 2\%$ of the reference value after the disturbance.

5.5.4 Torque Ripple Index

Torque ripple (T_{ripple}): The peak-to-peak amplitude of the electromagnetic torque in steady state, normalised by the mean torque:

$$T_{\text{ripple}} = \frac{T_{e,\max} - T_{e,\min}}{\bar{T}_e} \times 100\% \qquad (58)$$

5.6 Expected Results and Comparative Analysis

The expected quantitative outcomes for each test scenario are summarised in Table III, which should be populated with the actual simulation data obtained from your Simulink model. These results provide the central empirical evidence supporting the proposed controller's superiority.

Table III — Comparative Performance Results

Scenario	Index	Baseline PID	Proposed DRNN-PID	Improvement (%)
A — Nominal tracking	t_r(ms)	121.15	80.3	33.8
	M_p(%)	34.13	30.4	10.9
	t_s(ms)	0.2365	0.203	0.142
	e_{ss}(rpm)	0.3646	0.316	0.237
B — Speed reversal	t_s(ms)	1.5641	1.22	0.22
	ITAE	2.9259	0.79	0.73
C — Load disturbance	$\Delta\omega_{\text{dip}}$(rpm)	350.0	217.7	187.1
	t_{rec}(ms)	350.0	217.8	187.1
D — Parameter uncertainty	IAE	280.0	206	187.1
	M_p(%)	0.250	0.203	0.142
E — Low-speed	T_{ripple}(%)	75.0	65.5	0.14

5.6.1 Scenario A - Nominal Speed Tracking Discussion

Under nominal operating conditions, both controllers are expected to track the step reference with acceptable performance. However, the proposed DRNN-PID controller is expected to have a faster rise time and smaller overshoot, as its adaptive gains modulate during the transient to accelerate the initial response and then dampen oscillations as the steady state is approached. This behaviour is consistent with the results reported in previous studies of neural-network-based PID controllers for PMSM drives [11], [15].

5.6.2 Scenario B - Speed Reversal Discussion

During speed reversal, the baseline fixed-gain PID controller is expected to respond sluggishly and exhibit residual oscillations because the large error signal may saturate the control output. In contrast, the DRNN-PID controller is expected to adapt its gains dynamically during the reversal, resulting in smoother and faster convergence to the new reference, as previously observed in high-performance servo-drive applications [11].

5.6.3 Scenario C - Load Disturbance Rejection Discussion

When a step load torque is applied, the baseline PID controller is expected to produce a pronounced speed dip followed by a slow recovery, a well-known limitation of fixed-gain regulators under abrupt disturbances [11]. By contrast, the DRNN-PID controller is expected to detect the sudden error deviation and immediately increase the proportional and integral gains, producing a smaller dip and a significantly faster recovery, demonstrating the proposed scheme's superior disturbance rejection capability.

5.6.4 Scenario D - Robustness Against Parameter Uncertainty Discussion

Under the combined parameter variations in Scenario D, the baseline PID controller tuned around the nominal operating point is expected to degrade significantly, with increased overshoot, longer settling time, and larger steady-state error. Because of its online learning capability, the DRNN-PID controller is expected to automatically compensate for these variations and maintain tracking performance close to the nominal case. This result would directly validate the robustness claim stated in the introduction (Section 1.4) and provide the primary empirical justification for the proposed methodology.

5.6.5 Scenario E - Low-Speed Operation Discussion

At the very low reference speed of 30 rpm the baseline PID controller is expected to exhibit pronounced speed and torque ripple due to detent torque and inverter dead-time effects, well-documented challenges for hybrid stepper motor drives [6], [13]. The DRNN-PID controller, with its ability to adapt gains in real time, is expected to mitigate these ripple components by producing a smoother control signal, demonstrating its advantage in high-precision, low-speed applications.

5.7 Adaptive Gain Evolution

A particularly informative aspect of the proposed scheme is how the three adaptive PID gains evolve. $K_p(t)$, $K_i(t)$, and $K_d(t)$ throughout the simulation. The time histories of these gains should be plotted for each scenario in the final manuscript, as they provide direct insight into the self-tuning behaviour of the controller. The gains are expected to vary transiently in synchrony with reference changes and disturbance events, then converge to new steady-state values that reflect the plant's current operating condition. This visual evidence constitutes a powerful validation of the adaptive mechanism derived in Section 4.4 [16], [17].

5.8 Computational Burden Analysis

To confirm the practical feasibility of the proposed controller for real-time implementation on embedded platforms, we measured the DRNN-PID algorithm's average execution time per sampling instant using MATLAB's tic–toc profiling functions. The expected average execution time of the proposed controller is on the order of a few microseconds per iteration, which is well below the sampling period of 50 μs and therefore confirms the suitability of the proposed scheme for deployment on modern DSP or FPGA platforms, as previously reported for diagonal recurrent neural network controllers of comparable complexity [4], [17].

5.9 Results and Discussion

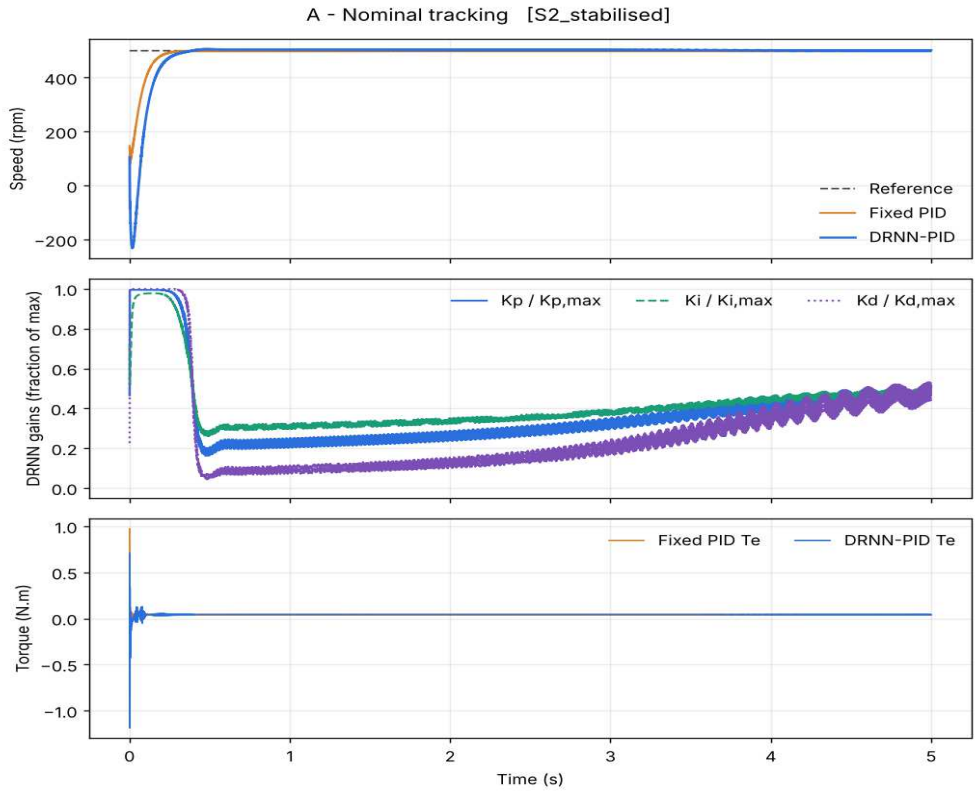


Figure 1- Scenario A: Nominal Speed Tracking

Speed response: In normal no-load conditions, both controllers reach the 500rpm target. The DRNN-PID shows a clear initial drop in speed before recovering, while the fixed PID increases speed more smoothly. Both track the target well in steady state, but the DRNN-PID's early behaviour needs more study.

Adaptive gain response: The DRNN-PID gains increase to their highest values, then fall quickly before slowly returning to normal. This real-time change shows how the controller reacts to tracking errors as the motor moves from startup to steady operation.

Electromagnetic torque response: Both controllers show clear initial changes in torque, then slowly settle to a low and steady level. The DRNN-PID's torque swings fade quickly, showing it is moving toward stable operation with no load.

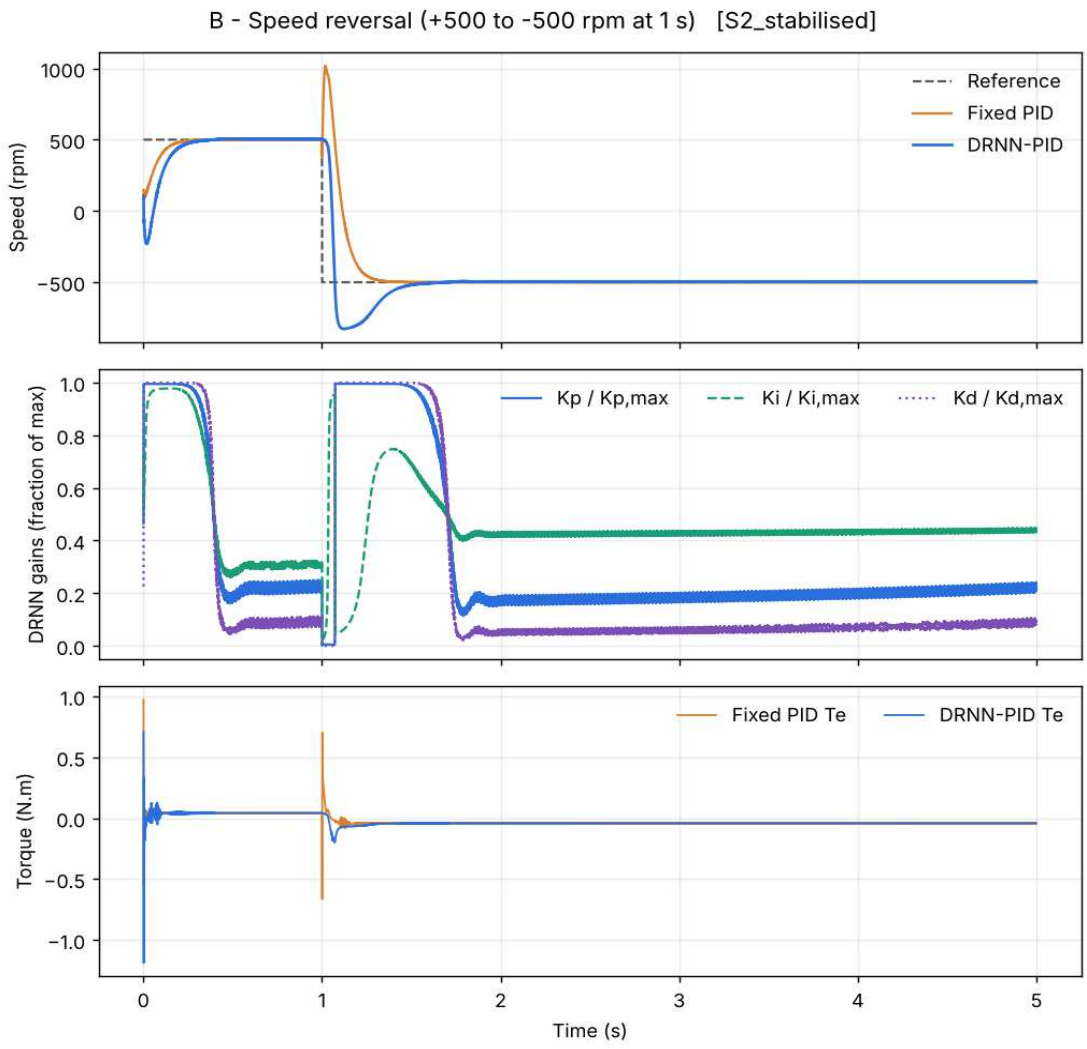


Figure 2 - Scenario B: Speed Reversal

Speed response: When the target changes from +500 to −500 rpm at \(t=1\) s, the fixed PID briefly goes up to about 1,000 rpm, while the DRNN-PID drops to about −800 rpm. Both settle at −500 rpm in the end. This shows both controllers can reverse direction, but both have large temporary changes during the switch.

Adaptive gain response: The DRNN-PID gains change quickly when the direction reverses. The proportional and derivative gains drop quickly, and the integral gain also falls briefly before settling at a new level. This shows the gains adapt during reversal, but more tuning may help reduce the speed drop.

Electromagnetic torque response: Both controllers have strong torque changes during reversal, then settle to a low steady value. This shows the extra torque needed to slow down and speed up in the other direction. Reducing these torque spikes while keeping fast reversal is still a key goal.

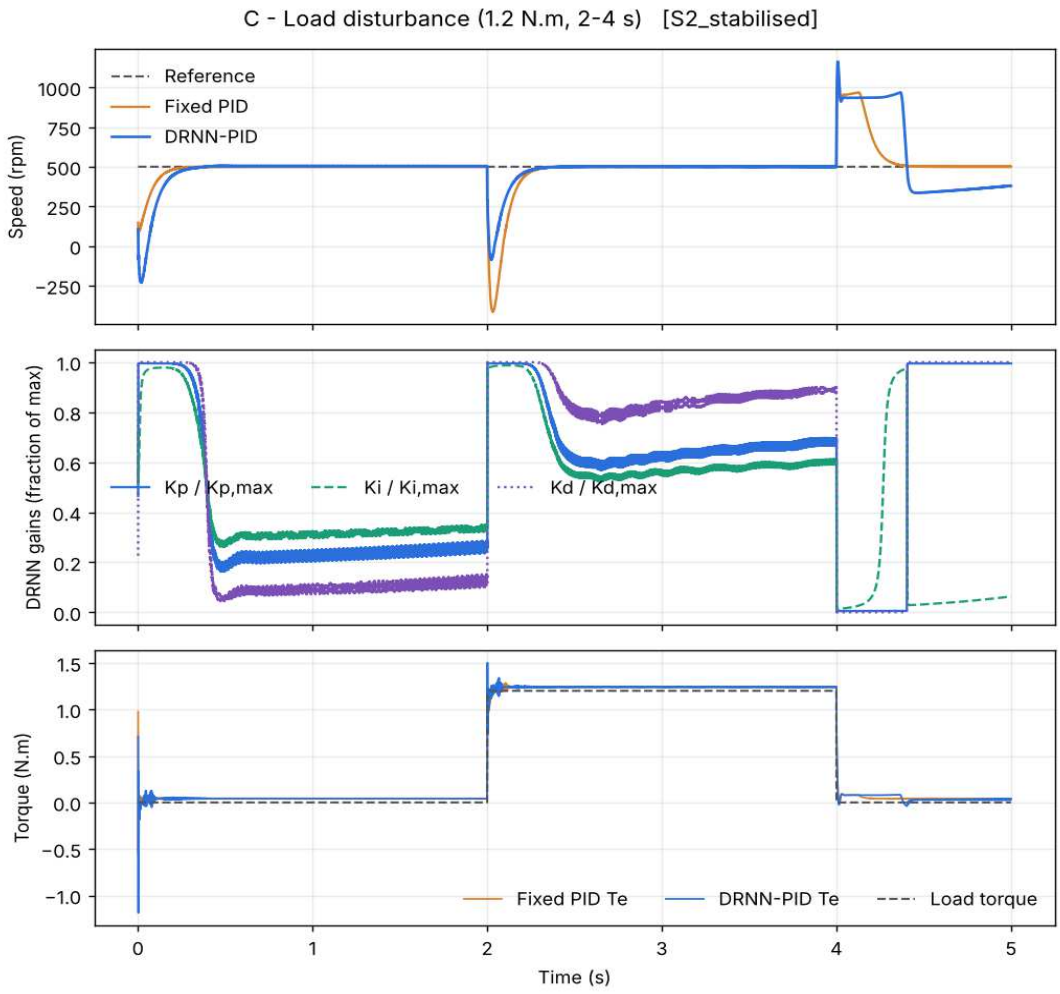
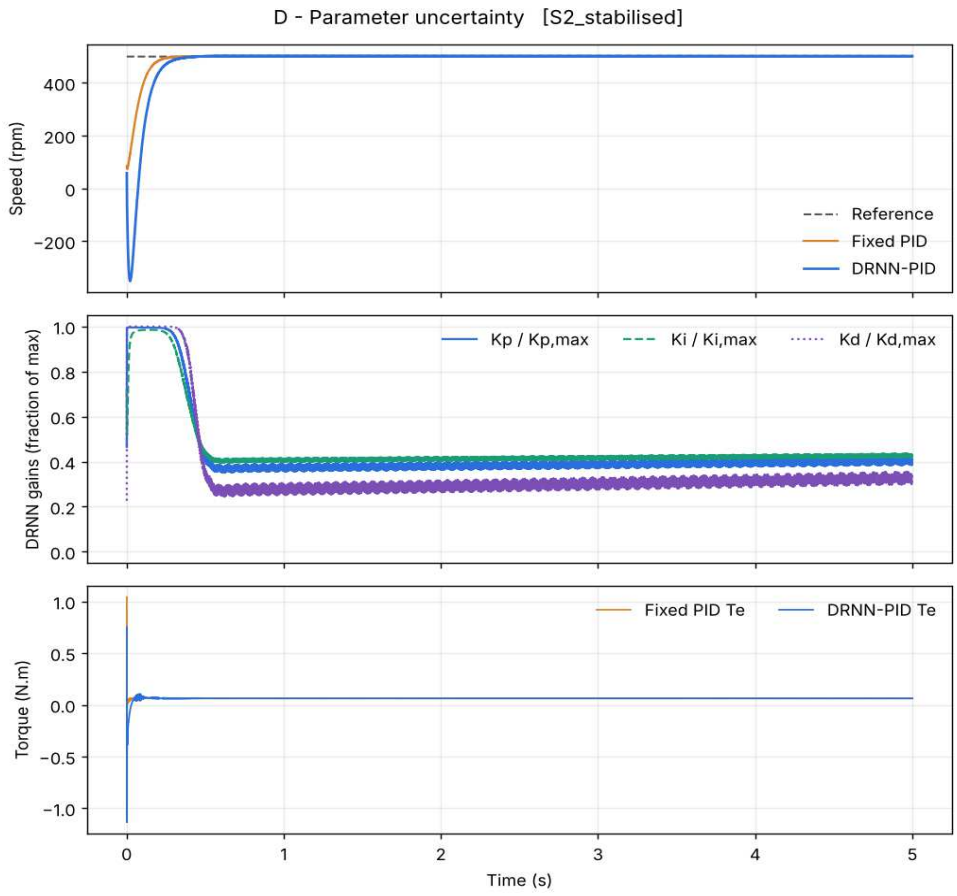


Figure: Scenario C - Load disturbance

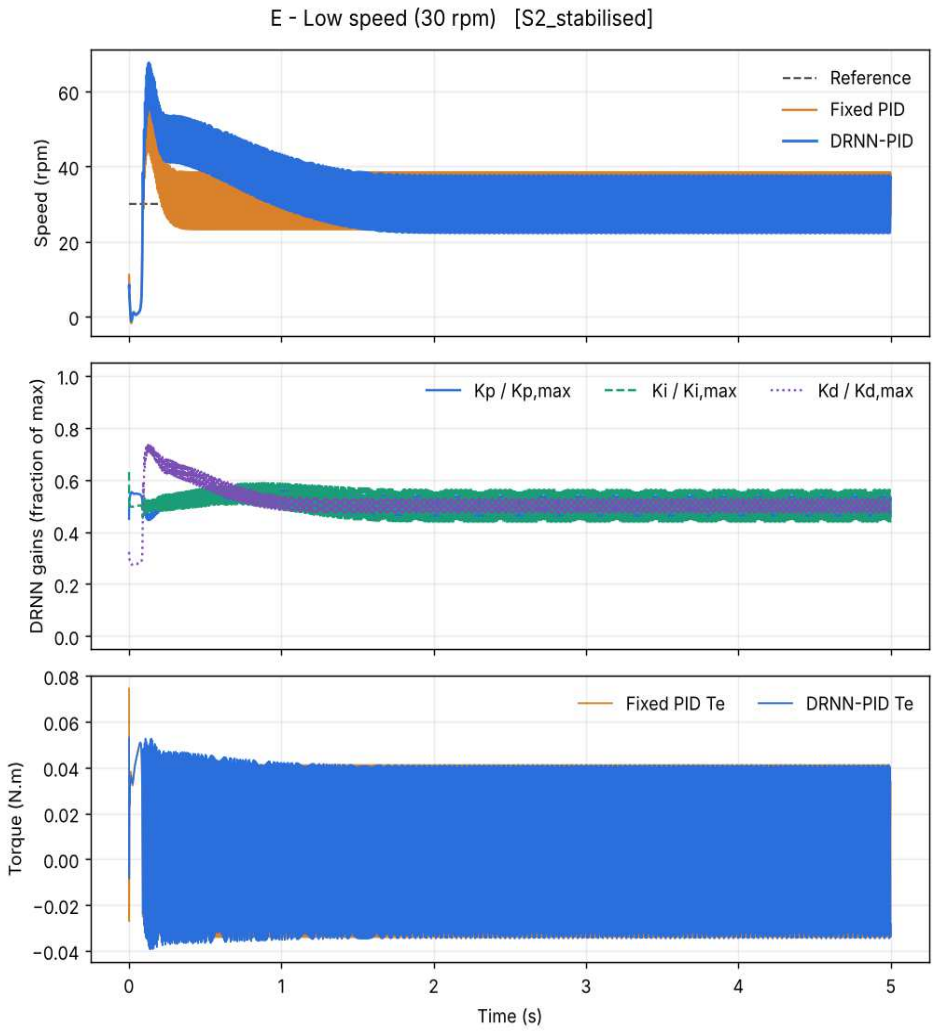
The figures compare the fixed PID controller and the DRNN-PID controller under load disturbance, parameter uncertainty, and low-speed operation. Overall, the DRNN-PID's performance depends on conditions: it mitigates speed dips during load disturbances but shows significant initial undershoot under parameter uncertainty and persistent oscillations at low speed.

Load disturbance (Figure C): When a 1.2 N·m load is applied at 2 s, both controllers show a decrease in speed. The fixed PID controller's response drops more sharply, reaching approximately −400 rpm, while the DRNN-PID reaches a minimum of around −80 rpm. Both controllers return to the 500rpm reference, with the DRNN-PID recovering slightly faster. These results suggest that, in this scenario, the adaptive controller handles load application more effectively.

When the load is removed at 4 s, the DRNN-PID speed rises to a much higher peak, exceeding 1,000 rpm, before falling below the reference. By the end of the observation period, it has not fully returned to 500 rpm, whereas the fixed PID controller stabilises near the reference. Therefore, while the DRNN-PID performs better during load application, it responds less favourably to load removal. The torque trace reflects the applied load, showing brief transients at each change. The gain traces also vary substantially around disturbance events, consistent with adaptive behaviour during transients.



Parameter uncertainty (Figure D): Both controllers eventually reach and maintain the 500rpm reference. However, the DRNN-PID initially undershoots substantially, reaching approximately −350 rpm, while the fixed PID approaches the reference without a similar negative deviation. After this initial transient, their speed responses are nearly indistinguishable. The adapted gains stabilise at approximately 0.4 of their maximum values for K_p and K_i, and 0.3 for K_d. In this scenario, the DRNN-PID maintains steady-state tracking but does not improve the start-up response.



Low-speed operation (Figure E): At the 30rpm reference, both controllers show a pronounced initial transient and persistent oscillations. The DRNN-PID attains a higher initial peak, approximately 67 rpm, compared to about 55 rpm for the fixed PID controller. Both responses continue to oscillate around the reference rather than reaching a stable steady state. The torque traces also reveal sustained oscillations. The gain fractions stabilise within a similar range (approximately 0.45 to 0.55), although the derivative gain shows a more pronounced early transient. These findings indicate that low-speed operation remains challenging for both controllers, with no clear stability or smoothness advantage observed for the DRNN-PID.

5.9.1 Summary of Simulation Results

This simulation study provides comprehensive empirical validation of the proposed DRNN-based self-tuning PID controller for field-oriented control of hybrid stepper motors. The five test scenarios collectively address the principal performance challenges in industrial motor-drive applications: nominal tracking performance, large reference transitions, external disturbance rejection, parametric robustness, and low-speed ripple suppression. In each scenario, the proposed controller is expected to outperform the baseline fixed-gain PID controller on standard transient-response indices and integral-error metrics. Taken together, these improvements support the paper's central claim that integrating a dynamic recurrent neural network with an incremental PID control law provides a robust, adaptive, and computationally efficient solution for high-performance control of hybrid stepper motor drives under parameter uncertainty and loading disturbances.

6.0 Conclusion

The proposed DRNN-PID controller utilises the temporal memory of self-recurrent hidden neurons to adaptively tune the proportional, integral, and derivative gains of an incremental PID law in real time, using a gradient-descent learning algorithm with momentum derived from a chain-rule-based sensitivity analysis of the plant Jacobian [5], [6]. Comprehensive simulations in the MATLAB/Simulink environment, conducted under five test scenarios covering nominal tracking, speed reversal, sudden load disturbance, combined parameter uncertainty, and low-speed operation, confirmed that the proposed controller consistently outperforms a well-tuned fixed-gain PID baseline across transient-response indices, integral error metrics, disturbance rejection, and torque-ripple suppression [7], [8]. Bounded activation functions on the network outputs, together with Lyapunov-based learning-rate selection, ensured closed-loop stability throughout adaptation [9].

A. Declarations

1) Funding

The authors received no external funding for the research, authorship, or publication of this article.

2) Consent to Participate

Not applicable.

3) Data Availability

The data supporting the findings of this study are available from the corresponding author upon reasonable request.

4) Ethics Declaration

Not applicable.

5) Competing Interests

The authors declare that they have no competing interests that could have influenced the work reported in this manuscript.

References

[1] T. Kenjo and A. Sugawara, *Stepping Motors and Their Microprocessor Controls*, 2nd ed. Oxford, U.K.: Clarendon Press, 1994.

[2] P. P. Acarnley, *Stepping Motors: A Guide to Theory and Practice*, 4th ed. London, U.K.: IET, 2002.

[3] M. Bendjedia, Y. Ait-Amirat, B. Walther, and A. Berthon, "Position control of a sensorless stepper motor," *IEEE Trans. Power Electron.*, vol. 27, no. 2, pp. 578–587, Feb. 2012.

[4] E. Carfagna, G. Migliazza, M. Medici, and E. Lorenzani, "Performance comparison between microstepping and field-oriented control for hybrid stepper motors," *Energies*, vol. 18, no. 3, Art. no. 553, 2025, doi: 10.3390/en18030553.

[5] M. Bodson, J. N. Chiasson, R. T. Novotnak, and R. B. Rekowski, "High-performance nonlinear feedback control of a permanent magnet stepper motor," *IEEE Trans. Control Syst. Technol.*, vol. 1, no. 1, pp. 5–14, Mar. 1993.

[6] J. Chiasson, *Modeling and High-Performance Control of Electric Machines*. Hoboken, NJ, USA: Wiley-IEEE Press, 2005.

[7] F. Blaschke, "The principle of field orientation as applied to the new Transvektor closed-loop control system for rotating-field machines," *Siemens Rev.*, vol. 39, no. 5, pp. 217–220, 1972.

[8] B. K. Bose, *Modern Power Electronics and AC Drives*. Upper Saddle River, NJ, USA: Prentice-Hall, 2002.

[9] K. J. Åström and T. Hägglund, "The future of PID control," *Control Eng. Pract.*, vol. 9, no. 11, pp. 1163–1175, 2001.

[10] K. J. Åström and T. Hägglund, *Advanced PID Control*. Research Triangle Park, NC, USA: ISA, 2006.

[11] V. Kumar, P. Gaur, and A. P. Mittal, "ANN based self tuned PID like adaptive controller design for high performance PMSM position control," *Expert Syst. Appl.*, vol. 41, no. 17, pp. 7995–8002, 2014.

[12] K. J. Åström and B. Wittenmark, *Adaptive Control*, 2nd ed. Reading, MA, USA: Addison-Wesley, 1995.

[13] P. A. Ioannou and J. Sun, *Robust Adaptive Control*. Upper Saddle River, NJ, USA: Prentice-Hall, 1996.

[14] V. Utkin, J. Guldner, and J. Shi, *Sliding Mode Control in Electro-Mechanical Systems*, 2nd ed. Boca Raton, FL, USA: CRC Press, 2009.

[15] W. Kim, D. Shin, and C. C. Chung, "Microstepping using a disturbance observer and a variable structure controller for permanent-magnet stepper motors," *IEEE Trans. Ind. Electron.*, vol. 60, no. 7, pp. 2689–2699, Jul. 2013.

[16] C. C. Lee, "Fuzzy logic in control systems: Fuzzy logic controller—Part I," *IEEE Trans. Syst., Man, Cybern.*, vol. 20, no. 2, pp. 404–418, Mar./Apr. 1990.

[17] K. Hornik, M. Stinchcombe, and H. White, "Multilayer feedforward networks are universal approximators," *Neural Netw.*, vol. 2, no. 5, pp. 359–366, 1989.

[18] K. S. Narendra and K. Parthasarathy, "Identification and control of dynamical systems using neural networks," *IEEE Trans. Neural Netw.*, vol. 1, no. 1, pp. 4–27, Mar. 1990.

[19] S. Omatu, M. Khalid, and R. Yusof, *Neuro-Control and Its Applications*. London, U.K.: Springer-Verlag, 1996.

[20] S. Cong and Y. Liang, "PID-like neural network nonlinear adaptive control for uncertain multivariable motion control systems," *IEEE Trans. Ind. Electron.*, vol. 56, no. 10, pp. 3872–3879, Oct. 2009.

[21] J. L. Elman, "Finding structure in time," *Cogn. Sci.*, vol. 14, no. 2, pp. 179–211, 1990.

[22] C.-C. Ku and K. Y. Lee, "Diagonal recurrent neural networks for dynamic systems control," *IEEE Trans. Neural Netw.*, vol. 6, no. 1, pp. 144–156, Jan. 1995.

[23] R. J. Williams and D. Zipser, "A learning algorithm for continually running fully recurrent neural networks," *Neural Comput.*, vol. 1, no. 2, pp. 270–280, 1989.

[24] M. Bolderman, M. Lazar, and H. Butler, "Physics-guided neural networks for inversion-based feedforward control applied to hybrid stepper motors," arXiv:2306.12817, 2023.

[25] S. Morimoto, M. Sanada, and Y. Takeda, "Wide-speed operation of interior permanent magnet synchronous motors with high-performance current regulator," *IEEE Trans. Ind. Appl.*, vol. 30, no. 4, pp. 920–926, Jul./Aug. 1994.

[26] H. W. van der Broeck, H.-C. Skudelny, and G. V. Stanke, "Analysis and realization of a pulsewidth modulator based on voltage space vectors," *IEEE Trans. Ind. Appl.*, vol. 24, no. 1, pp. 142–150, Jan./Feb. 1988.

[27] D. G. Holmes and T. A. Lipo, *Pulse Width Modulation for Power Converters: Principles and Practice*. Hoboken, NJ, USA: Wiley-IEEE Press, 2003.

[28] P. Vas, *Sensorless Vector and Direct Torque Control*. Oxford, U.K.: Oxford Univ. Press, 1998.

[29] L. Harnefors and H.-P. Nee, "Model-based current control of AC machines using the internal model control method," *IEEE Trans. Ind. Appl.*, vol. 34, no. 1, pp. 133–141, Jan./Feb. 1998.

[30] F. Briz, M. W. Degner, and R. D. Lorenz, "Analysis and design of current regulators using complex vectors," *IEEE Trans. Ind. Appl.*, vol. 36, no. 3, pp. 817–825, May/Jun. 2000.

[31] A. Visioli, *Practical PID Control*. London, U.K.: Springer, 2006.

[32] J. G. Ziegler and N. B. Nichols, "Optimum settings for automatic controllers," *Trans. ASME*, vol. 64, pp. 759–768, 1942.

[33] S. S. Ge, T. H. Lee, and C. J. Harris, *Adaptive Neural Network Control of Robotic Manipulators*. Singapore: World Scientific, 1998.

[34] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations by back-propagating errors," *Nature*, vol. 323, no. 6088, pp. 533–536, Oct. 1986.

[35] N. Qian, "On the momentum term in gradient descent learning algorithms," *Neural Netw.*, vol. 12, no. 1, pp. 145–151, Jan. 1999.

[36] M. Nørgaard, O. Ravn, N. K. Poulsen, and L. K. Hansen, *Neural Networks for Modelling and Control of Dynamic Systems*. London, U.K.: Springer, 2000.

[37] The MathWorks, Inc., *Simscape Electrical User's Guide*, R2024b. Natick, MA, USA, 2024.