

**T.C.
GEBZE TECHNICAL UNIVERSITY
GRADUATE SCHOOL**

**SOFT COMPUTING BASED VECTOR CONTROL OF A
PERMANENT MAGNET SYNCHRONOUS MOTOR**



FATİH BAYIR

**A THESIS OF DOCTORATE
DEPARTMENT OF ELECTRONICS ENGINEERING**

ADVISOR: PROF. DR. ERKAN ZERGEROĞLU

FEBRUARY 2026

**T.C.
GEBZE TECHNICAL UNIVERSITY
GRADUATE SCHOOL**

**SOFT COMPUTING BASED VECTOR CONTROL OF A
PERMANENT MAGNET SYNCHRONOUS MOTOR**

FATİH BAYIR

**A THESIS OF DOCTORATE
DEPARTMENT OF ELECTRONICS ENGINEERING**

ADVISOR: PROF. DR. ERKAN ZERGEROĞLU

FEBRUARY 2026

T.C.
GEBZE TEKNİK ÜNİVERSİTESİ
LİSANSÜSTÜ EĞİTİM ENSTİTÜSÜ

KALICI MIKNATISLI MOTORUN ESNEK
HESAPLAMA TABANLI VEKTÖR KONTROLÜ

FATİH BAYIR

DOKTORA TEZİ
ELEKTRONİK MÜHENDİSLİĞİ ANABİLİM DALI

DANIŞMAN: PROF. DR. ERKAN ZERGEROĞLU

ŞUBAT 2026

DOCTORATE JURY APPROVAL FORM

A thesis submitted by Fatih Bayır, defended on 18/02/2026 before the jury formed with the 20/01/2026 date and 2026/05 numbered decision of the GTU Graduate Administration Board, has been accepted as a DOCTOR of PHILOSOPHY thesis in the Department of Electronics Engineering.

JURY

MEMBER

(THESIS ADVISOR) : PROF. DR. ERKAN ZERGEROĞLU

MEMBER

: ASSIST. PROF. DR. ATILLA UYGUR

MEMBER

: ASSOC. PROF. DR. AHMET GÜNEŞ

MEMBER

: PROF. DR. ABDÜLKADİR BALIKÇI

MEMBER

: PROF. DR. ŞEREF NACİ ENGİN

APPROVAL

Gebze Technical University Graduate Administration Board

...../...../..... date and/..... numbered decision.

SIGNATURE/SEAL



In loving memory of my mother

ABSTRACT

This thesis develops advanced nonlinear and soft-computing-based control strategies for Permanent Magnet Synchronous Motors (PMSMs) to enhance robustness, tracking accuracy, and dynamic performance in the presence of parametric uncertainties and external disturbances. Conventional PI-based field-oriented control (FOC) can exhibit degraded performance when the PMSM dynamics are strongly nonlinear, coupled, and sensitive to parameter variations and load changes.

To address these challenges, two adaptive control architectures are proposed. The first integrates a fuzzy-logic-based parameter selection mechanism into an adaptive backstepping controller for the cascaded FOC structure. The fuzzy module adjusts key control/adaptation gains online based on instantaneous tracking-error information, improving responsiveness and robustness without relying on fixed gain values.

The second approach introduces a deep neural network (DNN)-assisted adaptive backstepping controller, where unmodeled dynamics, parameter variations, and load disturbances are treated as a lumped term estimated by the DNN. A hybrid learning strategy is employed: the DNN inner layers are pretrained offline using data collected from a PI-controlled PMSM drive to form a nonlinear feature mapping, while only the output-layer parameters are adapted online via Lyapunov-based update laws. This structure is designed to remain compatible with real-time execution while providing stability guarantees in the sense of bounded closed-loop signals and asymptotic tracking-error convergence under the stated assumptions and gain conditions.

The fuzzy-tuned adaptive backstepping controller is evaluated in MATLAB/Simulink against classical PI-based FOC, demonstrating improved tracking and disturbance rejection under parameter variations and load disturbances. For the DNN-assisted controller, simulation studies investigate the impact of network depth by comparing multiple architectures, and periodic inner-layer retraining using newly collected closed-loop data is additionally examined in simulation to assess the potential benefit of feature refinement. The digital implementation for this proposed controller is also presented. Thus, theoretical findings are supported by numerical and experimental studies.

Keywords: Soft computing, Adaptive Backstepping Control, Deep Neural Networks, Field Oriented Control, Non-linear Control.

ÖZET

Bu tez, Kalıcı Mıknatıslı Senkron Motorların (KMSM) parametre belirsizlikleri ve dış bozucular altında dayanıklılığını, izleme doğruluğunu ve dinamik performansını artırmak amacıyla gelişmiş, doğrusal olmayan ve yumuşak hesaplama tabanlı kontrol stratejileri geliştirmektedir. Geleneksel PI tabanlı alan yönlendirmeli kontrol (AYK), KMSM dinamiklerinin belirgin biçimde doğrusal olmayan, eksenler arası bağlı ve parametre değişimlerine ile yük etkilerine duyarlı olduğu durumlarda performans kaybı gösterebilmektedir.

Bu zorlukları gidermek için iki uyarlamalı kontrol mimarisi önerilmiştir. Birinci yaklaşım, kademeli AYK yapısı için uyarlamalı geri adımlama denetleyicisine bulanık mantık tabanlı bir parametre seçimi mekanizması entegre etmektedir. Bulanık modül, anlık izleme hata bilgisine dayanarak kritik kontrol/uyarlama kazançlarını çevrim içi olarak ayarlar; böylece sabit kazançlara bağımlı kalmadan tepki hızı ve dayanıklılık artırılır.

İkinci yaklaşım ise derin sinir ağı (DSA) destekli uyarlamalı geri adımlama denetleyicisi sunmakta; modellenmemiş dinamikler, parametre değişimleri ve yük bozucuları DSA tarafından tahmin edilen tekil bir terim olarak ele alınmaktadır. Bu yapıda hibrit bir öğrenme stratejisi kullanılmıştır: DSA'nın iç katmanları, doğrusal olmayan bir özellik eşlemesi oluşturmak üzere PI kontrollü bir KMSM sürücüsünden toplanan verilerle çevrim dışı olarak önceden eğitilir; yalnızca çıkış katmanı parametreleri ise Lyapunov tabanlı uyarlama yasalarıyla çevrim içi güncellenir. Bu yapı, gerçek zamanlı yürütmeye uygun olacak şekilde tasarlanmış olup, belirtilen varsayımlar ve kazanç koşulları altında kapalı çevrim sinyallerinin sınırlı kalmasına ve izleme hatalarının asimptotik yakınsamasına ilişkin kararlılık güvenceleri sağlar.

Bulanık ayarlı uyarlamalı geri adımlama denetleyicisi, MATLAB/Simulink ortamında klasik PI tabanlı AYK ile karşılaştırmalı olarak değerlendirilmiş; parametre değişimleri ve yük bozucuları altında daha iyi izleme ve bozucu bastırma performansı sergilediği gösterilmiştir. DSA destekli denetleyici için ise, farklı ağ mimarilerinin karşılaştırılmasıyla ağ derinliğinin etkisi incelenmiş ve ayrıca kapalı çevrimden yeni toplanan veriler kullanılarak iç katmanların periyodik yeniden eğitimi simülasyon ortamında değerlendirilerek özellik iyileştirmenin olası katkısı araştırılmıştır. Önerilen bu kontrolör için dijital uygulama da sunulmuştur. Böylece, teorik bulgular sayısal ve deneysel çalışmalarla desteklenmiştir.

Anahtar Kelimeler: Esnek Hesaplama, Uyarlamalı Geri Adımlamalı Kontrol, Derin Sinir Ağları, Alan Yönlendirmeli Kontrol, Doğrusal Olmayan Kontrol.

ACKNOWLEDGEMENTS

I sincerely thank Prof. Dr. Erkan ZERGEROĞLU, my supervisor, for his guidance, feedback, and support throughout this research. His expertise made this thesis possible. I also thank my wife, Kevser, for her patience and sacrifices, which allowed me to complete my doctoral education. I also would like to thank my children, Yavuz Selim and İpek Sare, for their patience and understanding throughout my thesis work. Finally, I am grateful to my family and all the teachers whose instruction laid the foundation for my academic work and research skills.



TABLE OF CONTENTS

	<u>Page</u>
ABSTRACT	vi
ÖZET	vii
ACKNOWLEDGEMENTS	viii
TABLE OF CONTENTS	ix
LIST OF SYMBOLS AND ABBREVIATIONS	x
LIST OF FIGURES	xi
LIST OF TABLES	xiii
1. INTRODUCTION	1
1.1. Motivation	1
1.2. Literature Review	3
1.3. Contributions of the Thesis	5
1.4. Thesis Organization	7
2. CONCEPTS AND TECHNIQUES APPLIED IN THE STUDY	9
2.1. Permanent Magnet Synchronous Motors	9
2.2. Field Oriented Control	12
2.2.1. Clarke and Park Transformations	14
2.2.2. Inverse Clarke and Park Transformation	15
2.2.3. Space Vector Pulse Width Modulation (SVPWM)	15
2.3. Soft Computing	18
2.3.1. Fuzzy Logic	20
2.3.2. Deep Neural Networks	22
2.4. Backstepping Control	24
2.5. Adaptive Backstepping Control	26
3. CONTROL DESIGN & ANALYSIS	29
3.1. Mathematical Model of PMSM	29
3.2. Backstepping and Adaptive Backstepping Controller Design	32
3.2.1. Design and Stability Analysis of Backstepping Controller	32
3.2.2. Design and Stability Analysis of the Adaptive Backstepping Control	37
3.2.3. Real Time Parameter Selection Mechanism based on Fuzzy Logic	41
3.2.4. Performance Evaluation of the Proposed Controllers	43
3.3. DNN based Adaptive Backstepping Controller Design	52
3.3.1. System Dynamics and Error Definitions	53
3.3.2. Design of the Control Loops	54
3.3.3. Stability Analysis	60
3.3.4. Numerical Studies	62
3.3.5. Simulation Method	63
3.3.6. Digital Implementation	80
4. CONCLUSION AND FUTURE WORK	91
REFERENCES	93
BIOGRAPHY	97
PUBLICATIONS AND PRESENTATIONS FROM THE THESIS	98
APPENDICES	99

LIST OF SYMBOLS AND ABBREVIATIONS

Hz	: Hertz
rad/s	: Radian per Second
s	: Second
t	: Time (s)
$\alpha\beta$	: Alpha-Beta
3D	: 3 Dimensional
ABC	: Adaptive Backstepping Controller
ANN	: Artificial Neural Network
BC	: Backstepping Controller
DNN	: Deep Neural Network
DQ	: Direct Quadrature
EMF	: Electromagnetic Force
FABC	: Fuzzy Adaptive Backstepping Controller
FLC	: Fuzzy Logic Control
FOC	: Field Oriented Control
IAE	: Integral of Absolute Error
ISE	: Integral of Squared Error
ITAE	: Integral of Time-weighted Absolute Error
ITSE	: Integral of Time-weighted Squared Error
MPC	: Model Predictive Control
PI	: Proportional Integral
PID	: Proportional Integral Derivative
PMSM	: Permanent Magnet Synchronous Motor
SMC	: Sliding Mode Control
SVPWM	: Space Vector Pulse Width Modulation
Relu	: Rectified Linear Unit
RMSE	: Root Mean Square Error
RPM	: Rotate per Minute
Tanh	: Hyperbolic Tangent
VSI	: Voltage Source Inverter

LIST OF FIGURES

	<u>Page</u>
Figure 2.1: Classification of motor types.	10
Figure 2.2: Structure of a surface mounted and interior PMSM.	10
Figure 2.3: Three phase electrical model of a PMSM.	11
Figure 2.4: Motor control types.	13
Figure 2.5: General FOC scheme.	14
Figure 2.6: Three phase voltage source inverter.	16
Figure 2.7: SVPWM hexagon.	18
Figure 2.8: Classification of soft computing methods.	19
Figure 2.9: FL block diagram.	21
Figure 2.10: Types of the membership functions.	22
Figure 2.11: Structure of a DNN.	23
Figure 3.1: DQ axis electrical model of the PMSM.	30
Figure 3.2: FLC real-time parameter selection scheme for ABC.	42
Figure 3.3: Matlab block diagram of the proposed system.	44
Figure 3.4: The speed response of the controllers for case 1.	46
Figure 3.5: Three phase stator currents for case 1.	46
Figure 3.6: The produced torques by the motor for case 1.	47
Figure 3.7: The currents i_d and i_q for case 1.	48
Figure 3.8: The speed-tracking performances for case 2.	48
Figure 3.9: Three phase stator currents for case 2.	50
Figure 3.10: The produced torque by the motor for case 2.	50
Figure 3.11: The currents i_d and i_q for case 2.	51
Figure 3.12: Estimated parameters and the disturbance torque for case 2.	51
Figure 3.13: The adaptive laws for case 1 and case 2.	52
Figure 3.14: Block diagram of the proposed DNN based controller.	60
Figure 3.15: PI controller performance for dataset generation.	64
Figure 3.16: Load torque profiles for the simulation study.	67
Figure 3.17: Scenario 1, the speed-tracking performance.	70
Figure 3.18: Zoomed speed views of scenario 1.	70
Figure 3.19: Outer layer adaptations in scenario 1.	70
Figure 3.20: Stator currents and control input in scenario 1.	71
Figure 3.21: Scenario 2, the speed-tracking performance.	71
Figure 3.22: Zoomed views of the speed in Scenario 2.	72
Figure 3.23: Outer layer adaptations in scenario 2.	72
Figure 3.24: Stator currents and control input in scenario 2.	72
Figure 3.25: Scenario 3, the speed-tracking performance.	73
Figure 3.26: Zoomed views of the speed in scenario 3.	73
Figure 3.27: Outer layer adaptations in scenario 3.	74
Figure 3.28: Stator currents and control input in scenario 3.	74
Figure 3.29: Scenario 4, the speed-tracking performance.	75
Figure 3.30: Zoomed views of the speed in Scenario 3.	75
Figure 3.31: Outer layer adaptations in scenario 4.	75
Figure 3.32: Stator currents and control input in Scenario 4.	76
Figure 3.33: Hardware layout of LAUNCHXL F28379D development kit [46].	80
Figure 3.34: Pinout diagram of BOOSTXL DRV8305EVM motor driver [47].	81
Figure 3.35: Designed circuit scheme for the load torque display.	82
Figure 3.36: Bottom PCB view of the load torque display circuit.	83

Figure 3.37: Experimental setup.	84
Figure 3.38: Speed-tracking performance of Scenario 1.	86
Figure 3.39: Currents for scenario 1.	87
Figure 3.40: Control inputs for scenario 1.	87
Figure 3.41: Outer layer weights update for Scenario 1.	88
Figure 3.42: Speed-tracking performance of scenario 2.	89
Figure 3.43: Currents for scenario 2.	89
Figure 3.44: Control inputs for scenario 2.	90
Figure 3.45: Outer layer weights update for Scenario 2.	90
Figure C1: 3D-printed output of the motor mounting platform.	103
Figure C2: 3D view of the experimental setup.	103



LIST OF TABLES

	<u>Page</u>
Table 2.1: Switching States of Three Phase VSI	17
Table 3.1: Fuzzy Logic Rule Base for parameters $ka1$ and $\Gamma1$	43
Table 3.2: PMSM parameters for Fuzzy Adaptive Backstepping Controller.	44
Table 3.3: Quantitative performance comparison of the controllers	49
Table 3.4: Teknik M-2310P-LN-04K PMSM parameters	63
Table 3.5: DNN specifications	65
Table 3.6: Controller Design Gains	66
Table 3.7: Load Torque Expressions	68
Table 3.8: DNN Estimation RMSE Evolution through Periodic Retraining	76
Table 3.9: Comprehensive Performance Indices Across Representative Scenarios	79
Table 3.10: Specifications of DC Generator [48]	83
Table 3.11: Components of the Experimental Setup	84

1. INTRODUCTION

1.1. Motivation

Permanent Magnet Synchronous Motors (PMSMs) have become the preferred choice in a wide range of industrial and commercial applications, including electric vehicles, robotics, aerospace systems, and precision manufacturing [1], [2]. These motors are widely favored in industrial applications due to their superior performance, including high power density, low torque ripple, high efficiency, and quiet operation [3]-[5]. Despite their advantages, controlling PMSMs poses significant challenges, as they are inherently nonlinear, multi-input, multi-output (MIMO) systems with strong coupling between electrical and mechanical dynamics. In addition, their performance is highly sensitive to parameter variations (e.g., stator resistance, rotor flux linkage, and load inertia) and external disturbances (e.g., sudden load changes or temperature-induced variations). These nonlinearities and uncertainties complicate the control design process and often lead to degraded performance if not properly addressed; consequently, this demands more sophisticated control approaches than conventional Proportional-Integral-Derivative (PID) controllers offer [6],[7].

Although PID controllers remain popular due to their simplicity and ease of implementation, their limitations become apparent with PMSMs. Since PID-based control scheme assumes linear, time-invariant system dynamics, it is inappropriate for the intrinsic nonlinearities in PMSMs—such as magnetic saturation, back electromotive force (EMF) coupling, and parameter uncertainties—that are not addressed in the classical PID formulation. Furthermore, PID controllers lack adaptability to real-time variations in load torque or operating conditions, often resulting in sluggish transient performance and steady-state errors. Consequently, in high-performance applications where precise torque or speed control is crucial, these shortcomings may result in instability or performance degradation, especially under rapidly changing dynamics or disturbances. Thus, relying solely on PID control is insufficient for robustness and precision in advanced PMSM drive systems, highlighting the need for more advanced strategies to ensure improved performance and reliability. As PMSMs become prevalent across sectors, addressing their control

complexities is essential, with robust stability in uncertain or disturbed environments now a central research challenge.

This dissertation focuses on the evolution of PMSM control strategies, emphasizing a shift from traditional methods to more advanced, adaptive solutions, particularly using nonlinear control techniques enhanced by soft computing methods. For example, the backstepping control has emerged as a promising solution in this context, offering improved robustness and performance for PMSM drive systems [8], [9]. When all motor parameters are known, the backstepping controller shows enhanced performance. However, when motor parameters are uncertain, this approach may be insufficient. In such cases, an adaptive backstepping control structure that can estimate the unknown parameters offers a more effective solution. To further enhance this method, the study also proposes a Fuzzy Logic (FL) Parameter Selection algorithm, a soft-computing technique that adjusts adaptive control parameters in real time. This allows the controller to adapt dynamically and improve the responsiveness of the speed-tracking loop. By leveraging the real-time adaptability of FL, the proposed method aims to fine-tune key parameters on the fly, thereby enhancing both responsiveness and robustness under varying operating conditions [10].

In addition to the FL structure, this thesis incorporates Deep Neural Networks (DNNs) to enhance the adaptive control architecture. The proposed framework combines an adaptive backstepping speed controller with a DNN-assisted q -axis current loop. Lyapunov-based adaptive laws update the uncertainty-compensation term online. The DNN approximates complex or uncertain motor dynamics. To balance computational efficiency with stability, the DNN is pre-trained offline using data from a conventional PI-controlled PMSM drive. Its inner-layer weights remain fixed during real-time operation. Only output-layer weights are updated online using Lyapunov-based adaptation laws. This hybrid approach improves robustness and tracking in the presence of disturbances and uncertainties. By integrating DNNs in this manner, the system addresses the limitations of purely model-based designs and enables intelligent, real-time-capable PMSM controllers.

Ultimately, this thesis aims to advance intelligent control strategies for PMSM systems by combining the structural stability of nonlinear methods with the adaptability of soft computing techniques, enabling high-performance operation under real-world uncertainties.

1.2. Literature Review

In most industrial applications, closed-loop control systems can be effectively regulated using PI type controllers. However, for systems exhibiting strong nonlinearities and fast dynamics, such as PMSM drives, alternative control strategies become not only beneficial but also often necessary to guarantee stability and achieve the desired performance. Recent studies show a focus on advanced control strategies, especially nonlinear strategies, to improve the robustness of PMSM systems against parametric uncertainties and torque disturbances.

Sliding mode control (SMC) has been shown to perform well in several studies, such as in [11], [12]. However, its usage is limited by chattering. Neural network-based strategies in [13], [14] improve accuracy and adaptability, yet they require extensive training data and may overfit. Model predictive control (MPC) in [15] achieves promising results but relies on precise system models, which are not always available. In contrast, backstepping control is highly effective for nonlinear systems with uncertainties, including mismatched uncertainties, as discussed in [16], [17]. Its ability to ensure global stability and excellent tracking performance has made it a popular choice for PMSM drives.

An adaptive backstepping controller in [18] successfully tackled position tracking under unknown load torque. Comparative studies in [9] highlighted its superior disturbance rejection and tracking accuracy. Despite these advances, previous research has largely neglected the impact of uncertainties in the current control loop on overall system stability. Fuzzy control techniques have also been applied to uncertain systems, such as Markov jump systems [19], [20], showing robust performance even without exact system models. Although MJS and PMSM systems differ structurally, both require adaptive methods to address nonlinearity and uncertainty. Building on these observations, the current study uses fuzzy logic not as a direct controller, as in MJS, but as an adaptive gain-tuning mechanism within the adaptive backstepping framework. In this strategy, fuzzy logic adjusts adaptation gains in real time based on velocity tracking errors. This adjustment improves controller performance across diverse operating conditions. Other related research includes [21], where the authors developed a neuroadaptive fractional-order backstepping control scheme for strict-feedback nonlinear systems. This scheme guarantees boundedness while avoiding

complexity explosion. Additionally, the study in [22] introduced a neural network-based backstepping controller for permanent magnet linear motor, employing radial basis function (RBF) networks to approximate uncertain system parameters. While this controller ensures system stability, it has relatively long settling times. In another example, [23] explored a deep neural network (DNN)-based adaptive control architecture for systems with unknown drift dynamics and uncertain control effectiveness matrices. The approach was validated on a two-state Van der Pol oscillator model. In another study [24], an extreme learning machine-based super-twisting integral terminal sliding mode controller for PMSM speed regulation is presented. This controller effectively mitigates chattering and compensates for disturbances. Nonetheless, addressing significant uncertainties and ensuring real-time implementation remain ongoing challenges in this area.

In [25], an adaptive backstepping controller integrated with a three-layer recurrent neural network for PMSM speed control was developed, where the NN weights are adjusted online using Lyapunov-derived learning algorithms to estimate electrical parameter variations. However, all network weights are updated simultaneously during real-time operation, which increases computational burden and may compromise stability guarantees on resource-constrained embedded platforms. In [26], the authors presented an adaptive neural network control for fractional-order PMSM using command-filtered backstepping, employing radial basis function networks to approximate unknown nonlinearities with Lyapunov-based weight adaptation laws. While the command filter technique successfully addresses the complexity explosion problem, the approach was validated only through simulation, and the full online adaptation of all network parameters raises concerns about real-time feasibility on DSP hardware. In [27], the authors proposed an event-triggered neural adaptive backstepping control for coupled chaotic PMSM systems, demonstrating improved transient performance through neural approximation. Nevertheless, the study did not address the computational constraints of embedded implementation or consider a hybrid training strategy that could reduce the online learning burden.

An offline-online hybrid deep reinforcement learning approach for hybrid electric vehicle powertrain control was introduced in [28], where the neural network is pretrained offline to establish initial knowledge before online policy refinement. Although this hybrid scheme accelerates convergence and improves learning stability,

the reinforcement-learning framework differs fundamentally from Lyapunov-based adaptation laws that can provide formal stability certificates. Furthermore, in [29], the authors developed a neural adaptive prescribed performance controller for chaotic PMSM using RBFNN to approximate unknown nonlinearities, incorporating tracking differentiators to avoid complexity explosion. While comprehensive simulation results were presented, the study did not explore inner-layer pretraining strategies or validate the controller on real-time embedded hardware, leaving practical implementation challenges unaddressed.

1.3. Contributions of the Thesis

This thesis presents advanced nonlinear and learning-based control strategies for PMSM drives, aiming to improve robustness, tracking accuracy, and disturbance rejection under significant model uncertainties and changing operating conditions. The main contributions are summarized below:

- I.** An adaptive backstepping controller is developed using the full-order PMSM model in the dq -axis reference frame. The control law is formulated to be fully compatible with standard FOC structures. A Lyapunov-based design and analysis establishes global asymptotic convergence of the speed and current tracking errors under the stated assumptions, in the presence of uncertainties in stator resistance, rotor inertia, viscous friction coefficient, and load torque.
- II.** A FL parameter selection algorithm is proposed to address the limitations of fixed adaptive gains. The FL block monitors the speed-tracking error and its rate of change to adjust key adaptation parameters in real time, such as Γ_1 (learning rate) and k_{a1} . This improves transient speed tracking, reduces control oscillations, and enhances robustness to disturbances. The resulting hybrid fuzzy-adaptive backstepping controller is shown, in simulation, to outperform conventional PI, standard backstepping, and fixed-gain adaptive backstepping controllers.
- III.** A hybrid learning-based adaptive backstepping control architecture is proposed. A deep neural network (DNN)-based adaptive backstepping controller is developed within an FOC framework for PMSM speed regulation, in which all uncertain and unmodeled terms in the torque-producing iq -channel are treated as a lumped nonlinear function and estimated by a DNN exploiting its universal approximation capability. A

cascade PI-controlled PMSM drive is first used to generate a structured and diverse dataset under ramp, constant, and sinusoidal reference profiles and multiple load conditions; this dataset is then employed to pretrain two candidate DNN architectures and to select a compact structure that offers a good trade-off between approximation accuracy and time-domain performance indices. In the final design, the pretrained inner-layer weights are kept fixed and act as nonlinear feature extractors (transfer learning-based DNN), while only the output-layer weights are updated online via Lyapunov-based adaptive laws. Moreover, in simulation studies, periodic offline retraining events for the inner-layer weights using data collected during operation are investigated to assess their impact on approximation accuracy and robustness. This hybrid (offline pretraining and online adaptation) approach enables real-time learning of the lumped uncertainty, maintains closed-loop stability, and ensures that the speed and current tracking errors converge asymptotically to zero under suitable gain conditions.

IV. All controllers developed in this thesis —adaptive backstepping, fuzzy-tuned adaptive backstepping, and DNN-assisted adaptive backstepping— are validated through extensive MATLAB/Simulink simulations. Performance is assessed under diverse operating scenarios, including sudden reference changes, sinusoidal speed tracking, and step-load disturbances. The analysis considers both transient and steady-state behavior using standard performance indices: root mean square error (RMSE), integral of squared error (ISE), integral of absolute error (IAE), integral of time-weighted squared error (ITSE), and integral of time-weighted absolute error (ITAE).

V. A systematic comparison is carried out between the proposed backstepping, adaptive backstepping and fuzzy-tuned adaptive backstepping controllers and conventional PI-based FOC. The results show that PI controllers perform adequately under nominal conditions but degrade rapidly under disturbances or parameter variations. In contrast, the proposed methods maintain high tracking fidelity, reduced overshoot, smaller steady-state errors, and faster settling times even under challenging operating conditions, thereby demonstrating improved robustness and practical performance.

VI. A major contribution of this thesis is the successful digital implementation and real-time validation of the proposed DNN-based adaptive backstepping controller. The

hardware platform consists of the Texas Instruments LAUNCHXL-F28379D digital signal processor (DSP), the BOOSTXL-DRV8305EVM three-phase inverter, and the Teknic M-2310P PMSM. The work includes developing real-time C code using the TI Driver Library (driverlib) and Code Composer Studio (CCS), integrating encoder-based position feedback, ADC-based motor current measurements, PWM signal generation, and Serial Peripheral Interface (SPI) communication with the DRV8305 gate driver. The implementation applies the DNN adaptation laws in a fixed-step 20 kHz current control loop, while logging real-time data such as speed, phase currents, and DNN weight evolution. This hardware realization demonstrates that the proposed learning-based controller is practical and fully implementable on embedded DSP hardware, confirming its feasibility for real-world motor-drive applications.

1.4. Thesis Organization

This thesis is structured into four main chapters, as follows:

Chapter 1 introduces the motivation for the study by presenting the challenges associated with high-performance PMSM control under parameter uncertainties and external disturbances. A comprehensive literature review is provided to position the proposed work within the current state of research, followed by a summary of the key contributions and the overall structure of the thesis.

Chapter 2 presents the fundamental concepts and techniques that form the basis of the proposed methods. The chapter begins with an overview of PMSM operation and modeling principles, followed by a detailed explanation of vector control, including Clarke-Park transformations, inverse transformations, and Space Vector PWM. The chapter then introduces soft computing tools—Fuzzy Logic and Deep Neural Networks—and concludes with a review of the backstepping control methodology used throughout the thesis.

Chapter 3 focuses on the design, theoretical development, and analysis of the proposed controllers. The chapter begins with the mathematical modeling of the PMSM in the dq-frame, followed by the design and Lyapunov-based stability analysis of both backstepping and adaptive backstepping controllers. A fuzzy logic-based parameter selection mechanism is introduced for real-time adjustment of adaptive gains. The chapter then presents the formulation of the DNN-based adaptive backstepping

controller, including its system dynamics, control loop design, stability proofs, and numerical studies. Comprehensive simulation results evaluating all proposed controllers, as well as the digital implementation of the DNN-based adaptive backstepping control, are also included.

Chapter 4 provides the overall results and concluding remarks. This chapter consolidates the findings of the simulation and analysis studies, highlights the advantages and limitations of the proposed approaches, and discusses the general implications of the research. Suggestions for future work are provided at the end of the chapter.



2. CONCEPTS AND TECHNIQUES APPLIED IN THE STUDY

This chapter provides the foundational knowledge and technical background essential for the development and implementation of the proposed PMSM control strategies. It begins with an overview of the permanent magnet synchronous motor (PMSM), outlining its key characteristics and dynamic behavior. The chapter then discusses the field-oriented control (FOC) technique, which forms the basis of the motor control framework used throughout this research. Next, the soft computing methods employed to enhance the adaptability and robustness of the control system, namely fuzzy logic and deep neural networks, are introduced. Finally, fundamental concepts of nonlinear control and backstepping control theory are reviewed to establish the theoretical basis for the adaptive control designs developed later in the thesis.

2.1. Permanent Magnet Synchronous Motors

Permanent Magnet Synchronous Motors (PMSMs) are a class of synchronous AC motors distinguished by their use of permanent magnets embedded in or attached to the rotor to generate the magnetic field, eliminating the need for rotor windings and excitation currents. This design offers several performance advantages compared to other motor types, making PMSMs highly favored in modern industrial and commercial applications.

Among various types of electric motors, including induction motors, brushed DC motors, and synchronous motors, PMSMs stand out for their high efficiency, high power density, and precise controllability [30]. They have become the preferred choice in applications demanding high performance, such as electric vehicles, robotics, aerospace, and precision manufacturing. **Figure 2.1** shows a sample classification of the electric motor types.

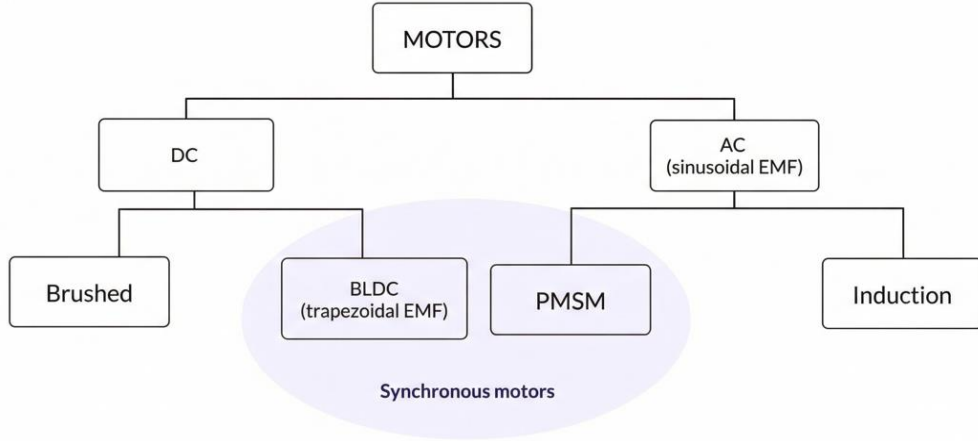


Figure 2.1: Classification of motor types.

The structure of a typical PMSM consists of a stator with three-phase windings that produce a rotating magnetic field when energized by an AC supply, and a rotor equipped with permanent magnets that generate a constant magnetic flux. The interaction between the stator's rotating magnetic field and the rotor's permanent magnets produces torque and causes the rotor to rotate synchronously with the stator field. PMSMs are commonly classified into two main types based on rotor magnet placement [31] as surface-mounted PMSMs (SPMSMs), where magnets are mounted on the rotor surface, and interior PMSMs (IPMSMs), where magnets are embedded within the rotor core, as seen in **Figure 2.2**. Each type exhibits distinct magnetic and mechanical characteristics suited to different applications.

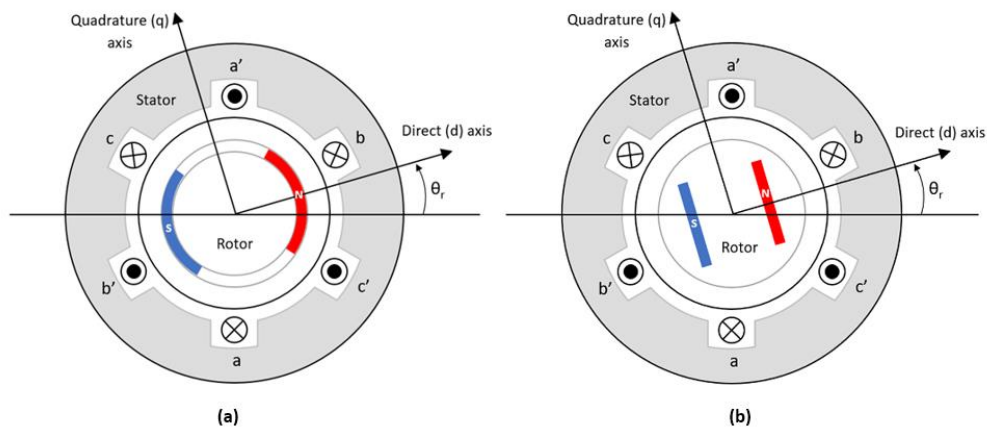


Figure 2.2: Structure of a surface mounted and interior PMSM.

The primary advantages of PMSMs include high efficiency due to reduced rotor losses, compact size owing to higher power density, low torque ripple resulting in smooth

operation, and quiet performance with minimal acoustic noise. Furthermore, the absence of brushes reduces maintenance requirements and enhances reliability. These characteristics contribute to the increasing adoption of PMSMs in systems where energy efficiency and dynamic response are critical. **Figure 2.3** shows the electrical equivalent of a PMSM.

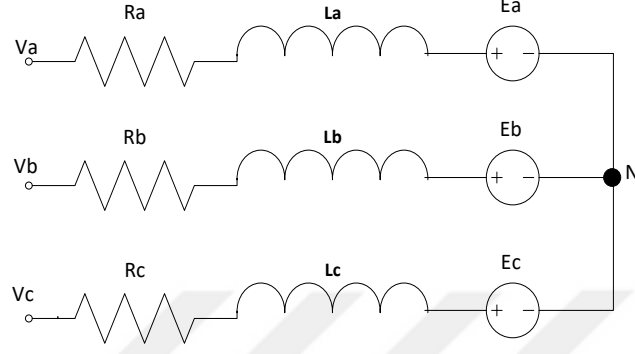


Figure 2.3: Three phase electrical model of a PMSM.

The electrical circuit part of PMSM model can be expressed as follows:

$$\begin{bmatrix} V_a \\ V_b \\ V_c \end{bmatrix} = \begin{bmatrix} R_a & 0 & 0 \\ 0 & R_b & 0 \\ 0 & 0 & R_c \end{bmatrix} \begin{bmatrix} i_a \\ i_b \\ i_c \end{bmatrix} + p \begin{bmatrix} \lambda_a \\ \lambda_b \\ \lambda_c \end{bmatrix} \quad (2.1)$$

where V_a, V_b, V_c , are the per phase stator voltages, $R_a = R_b = R_c = R$ are the per phase stator resistances, i_a, i_b, i_c are the per phase stator currents and λ_a, λ_b , and λ_c are the stator flux linkages and p stands for $\frac{d}{dt}$.

The stator currents i_a, i_b and i_c can be defined in matrix form as

$$\begin{bmatrix} i_a \\ i_b \\ i_c \end{bmatrix} = |\vec{I}_s| \cdot \begin{bmatrix} \sin wt \\ \sin(wt - \frac{2\pi}{3}) \\ \sin(wt + \frac{2\pi}{3}) \end{bmatrix} \quad (2.2)$$

where $|\vec{I}_s|$ is the magnitude of stator current, w is the stator field's frequency. The flux linkages λ_a, λ_b and λ_c can be expressed as

$$\begin{bmatrix} \lambda_a \\ \lambda_b \\ \lambda_c \end{bmatrix} = L \cdot \begin{bmatrix} i_a \\ i_b \\ i_c \end{bmatrix} + \begin{bmatrix} \Lambda_m \sin wt \\ \Lambda_m \sin(wt - \frac{2\pi}{3}) \\ \Lambda_m \sin(wt + \frac{2\pi}{3}) \end{bmatrix} \quad (2.3)$$

where Λ_m is the constant flux vector and L is the self-inductance of each stator winding ($L_a = L_b = L_c = L$). As can be observed from equation (4.1) it is clear that derivative of λ_{abc} must be defined to model the PMSM. Taking the derivative of equation (2.3) gives the following expression:

$$p \begin{bmatrix} \lambda_a \\ \lambda_b \\ \lambda_c \end{bmatrix} = L \cdot p \begin{bmatrix} i_a \\ i_b \\ i_c \end{bmatrix} + \begin{bmatrix} \Lambda_m \cdot w \cdot \cos \Theta \\ \Lambda_m \cdot w \cdot \cos(\Theta - \frac{2\pi}{3}) \\ \Lambda_m \cdot w \cdot \cos(\Theta + \frac{2\pi}{3}) \end{bmatrix} \quad (2.4)$$

Finally, substituting equation (2.4) into equation (2.1) and rearranging the matrix, the model equation can be expressed as follows:

$$\begin{bmatrix} V_a \\ V_b \\ V_c \end{bmatrix} = \begin{bmatrix} R \cdot i_a + L \cdot p \cdot i_a + \Lambda_m \cdot w \cdot \cos \Theta \\ R \cdot i_b + L \cdot p \cdot i_b + \Lambda_m \cdot w \cdot \cos(\Theta - \frac{2\pi}{3}) \\ R \cdot i_c + L \cdot p \cdot i_c + \Lambda_m \cdot w \cdot \cos(\Theta + \frac{2\pi}{3}) \end{bmatrix} \quad (2.5)$$

2.2. Field Oriented Control

Field-Oriented Control (FOC), one of the vector control methods, is a widely adopted method for controlling AC motors such as PMSMs. The main goal of vector control is to achieve independent control of the motor's torque and flux components, similar to the control of a separately excited DC motor [32]. This decoupling allows precise regulation of speed and torque, leading to enhanced dynamic performance and efficiency.

Vector control operates by transforming the three-phase stator currents into a rotating reference frame aligned with the rotor flux using mathematical transformations—namely Clarke and Park transformations [33]. This transformation converts the three-

phase currents into two orthogonal components: the direct-axis (d-axis) component, which controls the flux, and the quadrature-axis (q-axis) component, which controls the torque. By regulating these components independently, the controller can achieve fast and accurate response to changes in load and speed.

The advantages of vector control include improved torque response, higher efficiency, reduced torque ripple, and enhanced system stability under varying operating conditions. Compared to scalar control methods, such as Volts per Hertz (V/f) control, vector control provides superior dynamic performance, making it the preferred choice in high-performance PMSM applications.

Within the broader classification of motor control strategies, vector control represents an advanced closed-loop control technique positioned between basic open-loop methods and sophisticated nonlinear or adaptive control algorithms (see **Figure 2.4**). It serves as a fundamental control scheme upon which further enhancements, such as adaptive, fuzzy logic, or neural network-based controllers, can be built.

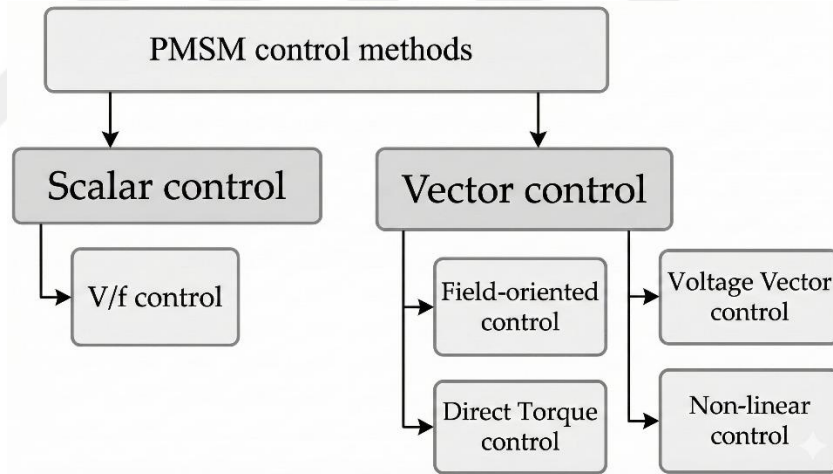


Figure 2.4: Motor control types.

A typical FOC scheme includes current sensors, coordinate transformation blocks (Clarke and Park transformations), a speed or position controller, and pulse width modulation (PWM) generation. The control commands are then applied to the inverter driving the motor. The overall structure is illustrated in the block diagram shown in **Figure 2.5**.

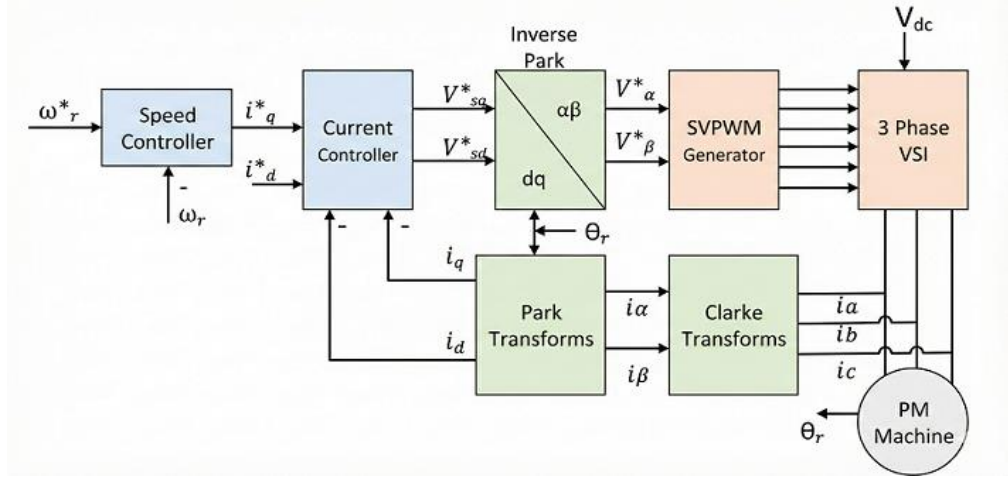


Figure 2.5: General FOC scheme.

2.2.1. Clarke and Park Transformations

The Clarke and Park transformations are fundamental mathematical tools used in the vector control of PMSMs to simplify and decouple motor dynamics.

The Clarke transformation first converts the three-phase stator quantities (abc) into a two-axis orthogonal coordinate system known as the stationary $\alpha\beta$ (alpha-beta) reference frame. This transformation projects the three-phase system onto a fixed Cartesian plane, reducing system complexity while preserving dynamic information. It assumes a balanced three-phase system without zero-sequence components and is expressed as a linear combination of the original phase variables.

Once in the $\alpha\beta$ frame, the Park transformation is applied to rotate the stationary reference frame into the dq (direct-quadrature) rotating frame, which aligns with the rotor's magnetic field. This transformation enables independent control of flux and torque, with the d -axis component regulating the magnetic flux and the q -axis component controlling the torque. The Park transformation depends on the rotor position angle, which must be known or estimated in real time.

Together, these transformations allow the AC motor to be controlled as if it were a DC motor, significantly enhancing dynamic performance, precision, and overall efficiency in high-performance applications. Equation (2.6) shows the Clarke and Park transformations combined in a matrix representation.

$$\begin{bmatrix} u_d \\ u_q \\ u_0 \end{bmatrix} = \frac{2}{3} \begin{bmatrix} \cos(\omega t) & \cos\left(\omega t - \frac{2\pi}{3}\right) & \cos\left(\omega t + \frac{2\pi}{3}\right) \\ -\sin(\omega t) & -\sin\left(\omega t - \frac{2\pi}{3}\right) & -\sin\left(\omega t + \frac{2\pi}{3}\right) \\ \frac{1}{2} & \frac{1}{2} & \frac{1}{2} \end{bmatrix} \begin{bmatrix} u_a \\ u_b \\ u_c \end{bmatrix} \quad (2.6)$$

2.2.2. Inverse Clarke and Park Transformations

The inverse Clarke and Park transformations are used to reconstruct the three-phase voltages (or currents) from the dq rotating reference frame back to the original three-phase abc stationary frame. This step is crucial in vector control schemes, as the control laws are applied in the dq domain, but the inverter must be supplied with three-phase signals to drive the motor.

The inverse Park transformation first converts the control voltages from the dq frame into $\alpha\beta$ stationary frame components by rotating them according to the rotor's electrical position. Subsequently, the inverse Clarke transformation maps the $\alpha\beta$ components into the original three-phase system. This two-step inverse transformation ensures that the computed control efforts are correctly translated into physical voltages applied to the motor windings. The combined transformation is represented by the following expression:

$$\begin{bmatrix} V_a \\ V_b \\ V_c \end{bmatrix} = \frac{2}{3} \begin{bmatrix} \cos(\omega t) & -\sin(\omega t) \\ \cos\left(\omega t - \frac{2\pi}{3}\right) & -\sin\left(\omega t - \frac{2\pi}{3}\right) \\ \cos\left(\omega t + \frac{2\pi}{3}\right) & -\sin\left(\omega t + \frac{2\pi}{3}\right) \end{bmatrix} \begin{bmatrix} u_d \\ u_q \end{bmatrix} \quad (2.7)$$

Equation (2.7) ensures accurate reconstruction of the three-phase quantities from the dq control inputs, allowing the inverter to generate the appropriate PWM signals for real-time motor control.

2.2.3. Space Vector Pulse Width Modulation (SVPWM)

Space Vector Pulse Width Modulation (SVPWM) is an advanced and widely used modulation technique for generating the gate signals of a three-phase Voltage Source

Inverter (VSI), which supplies the actual voltages to the stator windings of a PMSM. In the FOC framework, after the control algorithm computes the desired voltage components in the dq reference frame, these values are transformed back to the three-phase abc domain using the inverse Clarke and Park transformations. The resulting three-phase voltage references are then converted into switching commands through SVPWM, which ensures that these voltages are synthesized efficiently and accurately by the VSI.

The VSI is a power electronic device that converts a constant DC input voltage into three-phase AC voltages with controllable magnitude and frequency. It consists of six power switches (typically IGBTs or MOSFETs) arranged in a three-leg configuration, each leg corresponding to one motor phase. The role of the VSI is to generate the required AC voltage waveforms at the motor terminals based on the gate signals produced by the modulation algorithm. Without proper modulation, directly switching the VSI would result in high harmonic distortion and inefficient operation. **Figure 2.6** shows the schematic of a VSI.

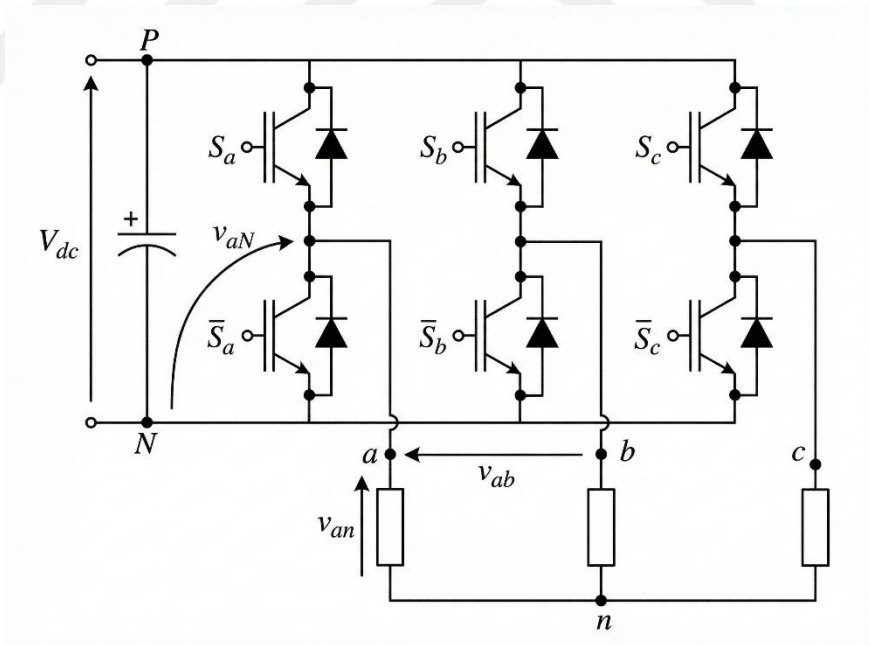


Figure 2.6: Three phase voltage source inverter.

The operating principle of a three-phase VSI is conceptually straightforward. Each of the three inverter legs is controlled by a binary signal—denoted as S_a , S_b , and S_c —which governs the switching state of the upper transistor in each leg. These control signals can only take values of 0 or 1, where 1 indicates that the upper switch is in the

“ON” state (conducting), and 0 indicates that it is “OFF” (non-conducting). To ensure safe and reliable operation, the lower switches of each leg are controlled using the logical complement of their respective upper switch signals. This complementary control avoids two critical issues: simultaneous conduction of both switches in a leg, which would short-circuit the DC bus; and simultaneous turn-off of both switches, which would lead to undefined output voltages at the motor terminals. Since there are three legs, and each can be in one of two states (ON or OFF), the VSI has a total $2^3 = 8$ possible switching states. Each of these states corresponds to a unique voltage space vector and set of phase voltages applied to the motor. These vectors form the foundation of the SVPWM technique and are typically illustrated in a space vector diagram, with their magnitudes and directions determining the inverter’s ability to approximate the desired reference voltage. **Table 2.1** lists these eight switching states and their corresponding space vectors.

Table 2.1: Switching states of three phase VSI.

Switching States	Gating Signals			Output Voltage			Space Vector
	S _a	S _b	S _c	V _{an}	V _{bn}	V _{cn}	
V ₀	0	0	0	0	0	0	V ₀ =0
V ₄	1	0	0	V _{dc}	0	0	V ₁ = $\frac{2}{3}V_{dc}$
V ₆	1	1	0	V _{dc}	V _{dc}	0	V ₂ = $\frac{2}{3}V_{dc} \cdot e^{j(\pi/3)}$
V ₂	0	1	0	0	V _{dc}	0	V ₃ = $\frac{2}{3}V_{dc} \cdot e^{j(2\pi/3)}$
V ₃	0	1	1	0	V _{dc}	V _{dc}	V ₄ = $-\frac{2}{3}V_{dc}$
V ₁	0	0	1	0	0	V _{dc}	V ₅ = $\frac{2}{3}V_{dc} \cdot e^{j(4\pi/3)}$
V ₅	1	0	1	V _{dc}	0	V _{dc}	V ₆ = $\frac{2}{3}V_{dc} \cdot e^{j(5\pi/3)}$
V ₇	1	1	1	V _{dc}	V _{dc}	V _{dc}	V ₇ = 0

SVPWM improves upon traditional sinusoidal PWM methods by treating the three-phase voltage references as a rotating space vector in a two-dimensional $\alpha\beta$ plane. It determines the nearest two active voltage vectors and the zero vector from the inverter’s switching table that can best approximate the desired reference vector within a PWM cycle. This approach provides better DC bus voltage utilization—up to 15% more than sinusoidal PWM—and lower total harmonic distortion in the output waveform [34]. These features translate to smoother torque output, increased

efficiency, and reduced heating in the motor. The phasor diagram of SVPWM is depicted in **Figure 2.7** as follows.

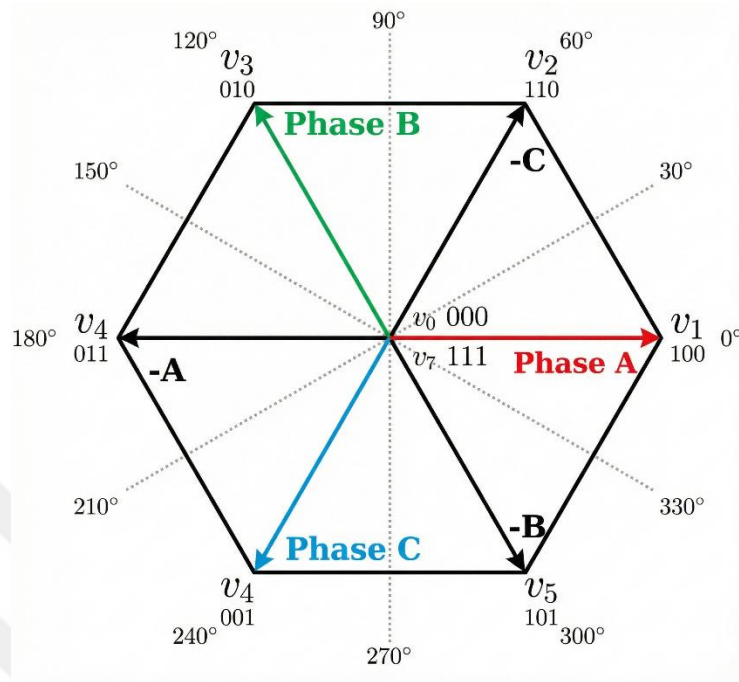


Figure 2.7: SVPWM hexagon.

In the context of FOC algorithm, SVPWM plays a vital role in ensuring that the computed control voltages from the dq controller are accurately and efficiently applied to the motor through the VSI. The combined use of Clarke and Park transformations, real-time modulation via SVPWM, and the power-stage execution by the VSI creates a complete and effective closed-loop control system for high-performance PMSM applications.

2.3. Soft Computing

Soft computing refers to a collection of computational techniques inspired by human reasoning and biological systems, designed to deal with imprecision, uncertainty, partial truth, and approximation [35]. Unlike traditional methods, which require well-defined mathematical models and deterministic logic, soft computing offers a more flexible and tolerant approach to problem solving [36]. This makes it particularly valuable for complex, nonlinear, and uncertain systems where precise analytical models are difficult or impossible to derive.

As seen the classification in **Figure 2.8**, The primary components of soft computing include Fuzzy Logic (FL), Artificial Neural Networks (ANNs), Evolutionary Algorithms (EAs), and Probabilistic Reasoning. Each of these techniques contributes to different aspects of decision-making, learning, and optimization:

- **Fuzzy Logic** provides a means to model reasoning with vague or linguistic information rather than binary logic. It is especially useful for control problems where system behavior is only partially known or hard to model precisely.
- **Artificial Neural Networks (ANNs)** are learning-based models inspired by the structure of the human brain. They are capable of approximating complex nonlinear functions and learning system dynamics from data.
- **Evolutionary Algorithms**, including Genetic Algorithms and Particle Swarm Optimization, are optimization techniques that simulate natural selection to find near-optimal solutions in large search spaces.
- **Probabilistic Reasoning** allows systems to handle uncertainty in a structured way, often through techniques such as Bayesian networks.

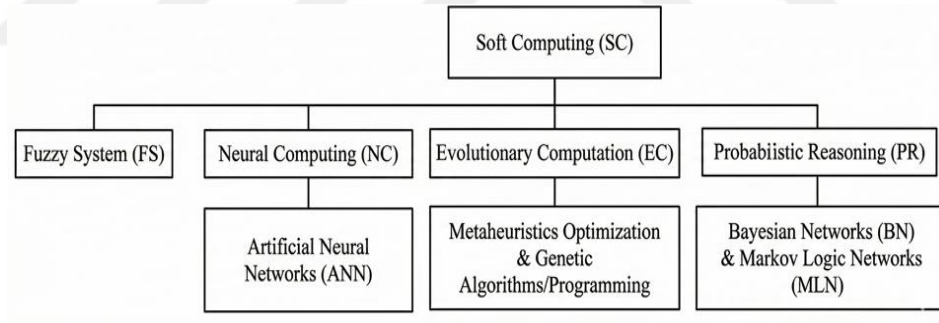


Figure 2.8: Classification of soft computing methods.

In the context of this thesis, Fuzzy Logic and Deep Neural Networks (DNNs), a specialized form of ANNs, are employed to enhance the performance and adaptability of nonlinear control strategies for PMSM drive systems. FL is used as a real-time gain adjustment mechanism within an adaptive backstepping framework, enabling the controller to respond dynamically to varying operating conditions without requiring an exact system model. On the other hand, DNNs are employed to approximate complex and uncertain motor dynamics that are difficult to capture through traditional modeling or simple adaptive laws. This hybrid approach leverages the strengths of soft

computing, namely learning, adaptability, and robustness, to address the challenges of PMSM control under uncertainty and disturbance.

The incorporation of soft computing techniques into control design not only improves robustness and flexibility but also facilitates intelligent behavior in real-time systems. This aligns closely with the goals of the present study, which seeks to develop an advanced, adaptive control framework for PMSMs that is capable of high-performance operation under practical conditions.

2.3.1. Fuzzy Logic

Fuzzy Logic (FL) is a soft computing paradigm that facilitates reasoning and decision-making under uncertainty, imprecision, and partial truth [37]. Unlike classical Boolean logic, which represents information using binary true/false values, fuzzy logic enables variables to assume values within a continuous interval $([0, 1])$. This capability allows fuzzy systems to effectively model nonlinear, time-varying, or poorly defined dynamics, making them particularly suitable for control problems in which precise mathematical models are difficult to derive or are sensitive to external conditions.

A fuzzy set A defined on a universe of discourse $X \subset \mathbb{R}$ is characterized by a membership function $\mu_a: X \rightarrow [0, 1]$ where $\mu_a(x)$ denotes the degree to which an element $x \in X$ belongs to the fuzzy set A . For example, if the linguistic term “*error is small*” is represented by the fuzzy set A_{small} , then $\mu_a(small)(e) \approx 1$ when the magnitude of the error $|e|$ is close to zero, and gradually decreases as $|e|$ increases. Let the fuzzy system input vector be $x = [x_1, x_2, \dots, x_n]^T$ where in this thesis, x_1 may represent the tracking error and x_2 its derivative. Each input variable has multiple linguistic terms $T_i = \{A_{i1}, A_{i2}, \dots, A_{iM}\}$ with corresponding membership functions $\mu_{A_{ij}}(x_i)$. A fuzzy rule R_r can be expressed as:

IF x_1 is A_{1j1} AND x_2 is A_{2j2} AND ... AND x_n is A_{njn} THEN y is B_r

In a FL controller, control knowledge is represented linguistically through rules such as “if the error is small, then decrease the control effort slightly,” mimicking heuristic human reasoning and providing interpretability. As illustrated in **Figure 2.9**, a typical FL controller consists of four fundamental components:

- **fuzzification**, which maps numerical inputs to fuzzy sets;

- **a rule base**, containing the set of linguistic if-then rules;
- **an inference engine**, which evaluates the rules and determines the resulting fuzzy output;
- **defuzzification**, which converts the fuzzy output into a crisp control signal. Both Mamdani and Sugeno type inference mechanisms are commonly employed, with Sugeno systems often preferred in real-time control due to their reduced computational burden and ease of implementation.

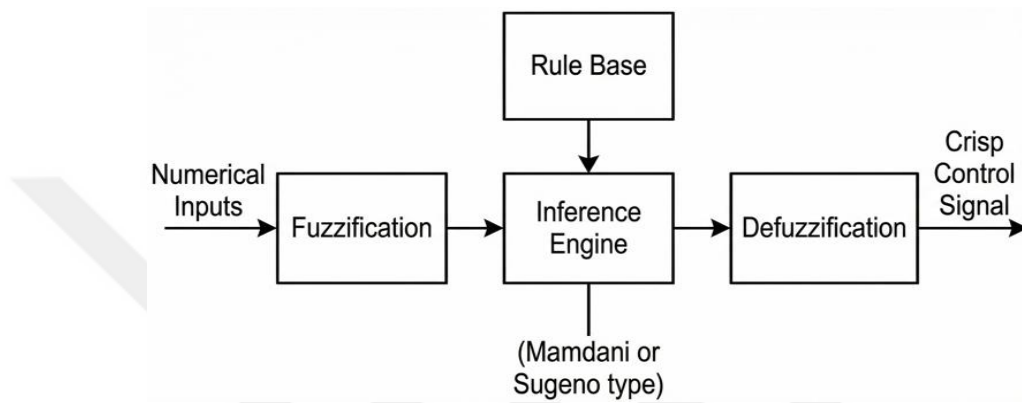


Figure 2.9: FL block diagram.

Membership functions are used to describe linguistic terms such as *small*, *medium*, or *large*, and may take various forms including triangular, trapezoidal, Gaussian, bell-shaped, or sigmoid functions (see **Figure 2.10**). The choice of membership function influences smoothness, computational complexity, and interpretability of the controller.

FL has found widespread application in domains such as robotics, automotive systems, renewable energy systems, and power electronics. In this thesis, fuzzy logic is not used as a standalone controller; rather, it serves as a parameter-adjustment mechanism within the adaptive backstepping control architecture. Specifically, a fuzzy inference system dynamically updates the adaptation gains based on real-time tracking error behavior. This online tuning enhances responsiveness and robustness of the PMSM control system under varying operating conditions, disturbances, and parameter uncertainties, circumstances under which constant adaptation gains may yield limited performance or slow convergence.

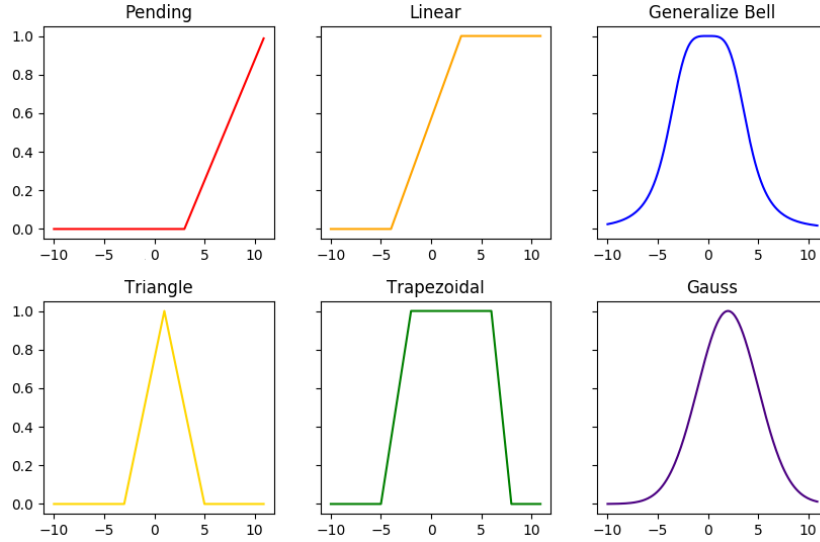


Figure 2.10: Types of the membership functions.

By embedding FL into the adaptive controller, the proposed methodology combines the interpretability of rule-based reasoning with the rigorous stability guarantees of Lyapunov-based adaptation, resulting in a more flexible and intelligent control strategy for PMSM drives.

2.3.2. Deep Neural Networks

Deep Neural Networks (DNNs) are advanced machine learning models inspired by the structure and functioning of the human brain. They consist of multiple layers of interconnected processing units (neurons) that can approximate highly complex and nonlinear relationships between inputs and outputs.

As a subclass of Artificial Neural Networks (ANNs), DNNs differ primarily in their depth, the use of several hidden layers, allowing them to learn hierarchical representations of data [38]. This makes DNNs particularly effective for tasks involving pattern recognition, function approximation, and control in systems with significant nonlinearities and uncertainties.

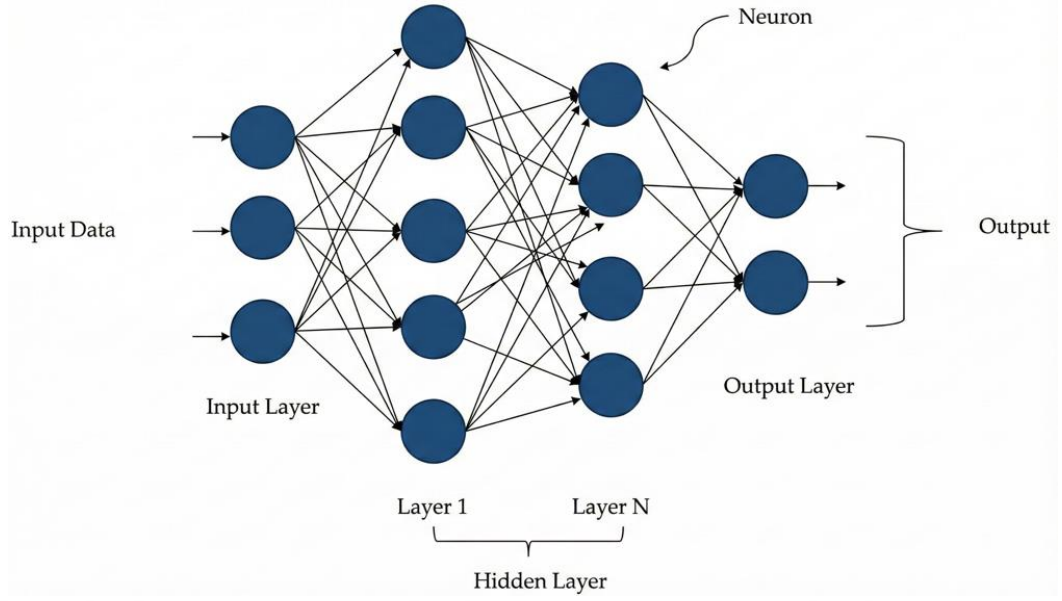


Figure 2.11: Structure of a DNN.

Mathematically, the operation of a single neuron in a neural network layer can be described as follows:

$$y = \varphi \left(\sum_{i=1}^n x_i w_i + b \right) \quad (2.8)$$

where

- x_i are the input features to the neuron, with $i = 1, 2, \dots, n$,
- w_i are the corresponding weights,
- b is the bias term,
- $\varphi(\cdot)$ is the activation function (e.g., sigmoid, ReLU, tanh),
- y is the output of the neuron.

In control engineering, DNNs serve as powerful universal function approximators [39]. Unlike traditional model-based approaches that require accurate system modeling, DNNs learn system dynamics directly from input-output data. This learning-based capability is especially valuable for systems like PMSMs, where exact modeling of drift dynamics, parameter uncertainties, or unmodeled nonlinearities can be difficult. Once trained, a DNN can generalize well to unseen operating conditions, provided that the training dataset is sufficiently representative.

In this thesis, a DNN is integrated into the PMSM control architecture to enhance the adaptability and performance of the system. Specifically, it is used to approximate uncertain or unmodeled dynamics within the q -axis current loop. The DNN is pre-trained offline using a dataset generated from a simulated PI-controlled PMSM system under various trajectories. During online operation, only the output-layer weights of the DNN are updated in real-time using Lyapunov-based adaptive laws, while the inner-layer weights remain fixed to preserve computational efficiency and ensure stability.

This hybrid use of offline learning and online adaptation allows the control system to leverage both the expressive power of DNNs and the theoretical guarantees of nonlinear adaptive control. By embedding DNNs into the backstepping framework, the proposed controller is capable of handling complex nonlinear dynamics and uncertainties that would otherwise degrade performance using conventional adaptive methods. This integration marks a step forward in the design of intelligent, high-performance PMSM controllers suitable for real-time industrial applications.

2.4. Backstepping Control

Backstepping is a recursive nonlinear control design methodology developed to stabilize a class of nonlinear systems known as strict-feedback systems [40]. It is particularly useful in systems where the control input does not appear directly in the first differential equation but instead affects the system through a chain of integrators. This approach constructs both the control law and a Lyapunov function step-by-step, ensuring system stability at each recursive stage.

The popularity of backstepping stems from its flexibility in handling matched uncertainties and nonlinearities and its ability to provide systematic controller designs even when the system is under actuated or partially known. It has been widely used in various applications such as robotic manipulators, aerospace vehicles, power converters, and electrical machines including PMSMs.

Consider a general strict-feedback nonlinear system of the form:

$$\begin{aligned}
\dot{x}_1 &= f_1(x_1) + g_1(x_1)x_2 \\
\dot{x}_1 &= f_1(x_1) + g_1(x_1)x_2 \\
\dot{x}_2 &= f_2(x_1, x_2) + g_2(x_1, x_2)x_3 \\
&\dots \\
\dot{x}_{n-1} &= f_{n-1}(x_1, \dots, x_{n-1}) + g_{n-1}(x_1, \dots, x_{n-1})x_n \\
\dot{x}_n &= f_n(x) + g_n(x)u
\end{aligned} \tag{2.9}$$

where $x = [x_1, x_2, \dots, x_n]^T \in \mathbb{R}^n$ is the state vector and $u \in \mathbb{R}$ is the control input. The functions $f_i(\cdot)$ and $g_i(\cdot)$ are assumed to be sufficiently smooth, and $g_i(x) \neq 0$ for all x to ensure control effectiveness.

The main idea of backstepping is to define virtual control inputs α_i at each step, and recursively stabilize each subsystem by treating the next state variable as the input to the previous one [41]. The final step designs the actual control input u such that the entire system becomes asymptotically stable.

Step 1: Choose a desired value x_{1d} and define the error by $z_1 = x_1 - x_{1d}$. Construct a Lyapunov function $V_1 = \frac{1}{2}z_1^2$, and choose a virtual control law $\alpha_1(x_1)$ such that $\dot{V}_1$ is negative definite.

Step 2: Define the error $z_2 = x_2 - \alpha_1(x_1)$ and augment the Lyapunov function:

$$V_2 = V_1 + \frac{1}{2}z_2^2 \tag{2.10}$$

Choose the virtual control α_2 to stabilize z_2 , and proceed similarly for subsequent steps.

Step 3: Design the actual control law u in the final step to stabilize the full system and ensure $\dot{V}_n < 0$.

At each backstepping stage, a composite Lyapunov function is constructed as:

$$V(x) = \sum_{i=1}^n \frac{1}{2}z_i^2 \tag{2.11}$$

where z_i denotes the tracking error at the i -th step. The control objective is to make $\dot{V} \leq -\sum_{i=1}^n c_i z_i^2$, where $c_i > 0$ design constants are, guaranteeing global asymptotic stability of the origin under the proposed control law.

2.5. Adaptive Backstepping Control

While classical backstepping assumes full knowledge of the system parameters, many real-world nonlinear systems including PMSMs are subject to parametric uncertainties such as unknown inertia, friction coefficients, or load torque. Adaptive backstepping extends the classical backstepping framework by introducing online parameter estimation laws that guarantee stability and performance even when such parameters are unknown. Adaptive backstepping is particularly well-suited for systems in strict-feedback form, where the unknown parameters can be linearly parameterized or bounded. The method combines Lyapunov stability theory with online adaptation laws, resulting in a control structure that ensures asymptotic convergence of tracking errors and boundedness of all internal signals [42].

Consider the following strict-feedback system with unknown parameters $\theta \in \mathbb{R}^p$:

$$\begin{aligned}\dot{x}_1 &= f_1(x_1) + g_1(x_1)x_2 \\ \dot{x}_2 &= f_2(x_1, x_2, \theta) + g_2(x_1, x_2)x_3 \\ &\vdots \\ \dot{x}_n &= f_n(x, \theta) + g_n(x)u\end{aligned}\tag{2.12}$$

where f_i depend on unknown constant parameters θ , and $u \in \mathbb{R}$ is the control input. Let the tracking error be defined as $z_1 = x_1 - x_{1d}$, and define virtual controls α_i at each step to stabilize each subsystem. The control objective is to design:

- A control law $u(x, \hat{\theta})$,
- An adaptive law $\dot{\hat{\theta}}$ for the parameter estimates $\hat{\theta}$, such that the tracking errors $z_i \rightarrow 0$ as $t \rightarrow \infty$, and the parameter estimation error $\tilde{\theta} = \theta - \hat{\theta}$ remains bounded.

At each step, a Lyapunov function is constructed recursively. For example, for a second-order system:

$$\begin{aligned}\dot{x}_1 &= x_2 \\ \dot{x}_2 &= \theta^\top \phi(x) + u\end{aligned}\tag{2.13}$$

where $\phi(x) \in \mathbb{R}^p$ is a known basis function vector, and θ is unknown.

1. Define tracking error: $z_1 = x_1 - x_{1d}$,
2. Choose virtual control: $\alpha_1 = -k_1 z_1 + \dot{x}_{1d}$
3. Define second error: $z_2 = x_2 - \alpha_1$,
4. Construct Lyapunov function:

$$V = \frac{1}{2} z_1^2 + \frac{1}{2} z_2^2 + \frac{1}{2\gamma} \tilde{\theta}^\top \tilde{\theta}\tag{2.14}$$

where $\gamma > 0$ is the adaptation gain.

5. Choose control input:

$$u = -k_2 z_2 - \hat{\theta}^\top \phi(x)\tag{2.15}$$

6. Choose adaptation law:

$$\dot{\hat{\theta}} = \gamma z_2 \phi(x)\tag{2.16}$$

Then the derivative of the Lyapunov function becomes:

$$\dot{V} = -k_1 z_1^2 - k_2 z_2^2 \leq 0\tag{2.17}$$

This ensures all signals in the closed-loop system are uniformly bounded, and $z_1, z_2 \rightarrow 0$ as $t \rightarrow \infty$, satisfying asymptotic tracking.

In PMSM systems, many parameters such as load torque, inertia, friction, or even magnetic saturation effects, may be partially known or vary with time. Adaptive backstepping is particularly useful in such systems for the following reasons [43]:

- It does not require precise knowledge of motor parameters.

- The control law remains robust under slowly time-varying or uncertain conditions.
- It allows online learning of unknown dynamics using adaptation laws.
- It maintains the recursive backstepping structure, enabling decoupled torque and flux control in dq-domain systems.



3. CONTROL DESIGN & ANALYSIS

This section presents the design and theoretical analysis of the proposed control strategies for the PMSM system. It begins with a detailed mathematical modeling of the PMSM in the rotating dq -reference frame, followed by the definition of key tracking and estimation errors necessary for control development. Based on this foundation, two advanced control architectures are introduced.

The first approach builds upon the adaptive backstepping method and integrates a fuzzy logic-based gain tuning mechanism to enhance robustness and responsiveness. The second approach extends the framework by incorporating a DNN to approximate unknown system dynamics, providing a hybrid learning-based solution for handling nonlinearities and uncertainties. For both methods, Lyapunov-based stability analysis is employed to guarantee convergence of the tracking errors and boundedness of all internal signals. These control designs form the theoretical basis for the simulation and experimental studies presented in the subsequent chapters.

3.1. Mathematical Model of PMSM

To enable decoupled control of the torque and flux components in PMSM, three-phase quantities are transformed into a synchronous rotating reference frame using the Park Transformation in equation (2.6). In this frame, the stator voltage equations become:

$$\begin{bmatrix} v_d \\ v_q \end{bmatrix} = \begin{bmatrix} L_d \frac{di_d}{dt} \\ L_q \frac{di_q}{dt} \end{bmatrix} + \begin{bmatrix} Ri_d \\ Ri_q \end{bmatrix} - \omega \begin{bmatrix} -L_q i_q \\ L_d i_d \end{bmatrix} + \begin{bmatrix} 0 \\ \omega \Phi_m \end{bmatrix} \quad (3.1)$$

where:

- v_d is the direct-axis stator voltage component (V),
- v_q is the quadrature-axis stator voltage component (V),
- i_d is the d -axis stator current (A),
- i_q is the q -axis stator current (A),
- R is the stator resistance (Ω),

- L_d is the d -axis stator inductance (H),
- L_q is the q -axis stator inductance (H),
- ϕ_m is the permanent magnet flux linkage (Wb),
- ω is the electrical angular velocity of the rotor (rad/s),
- $\frac{di_d}{dt}, \frac{di_q}{dt}$ are the time derivatives of i_d and i_q , respectively.

This form clearly separates the resistive, inductive, and coupling terms. The cross-coupling components $-\omega L_q i_q$ and $\omega L_d i_d$ reflect the interactions between d - and q -axis dynamics. The term $\omega \phi_m$ in the q -axis represents the back electromotive force (EMF) generated by the rotor magnets. **Figure 3.1** shows the electrical schematic of PMSM model in dq axis.

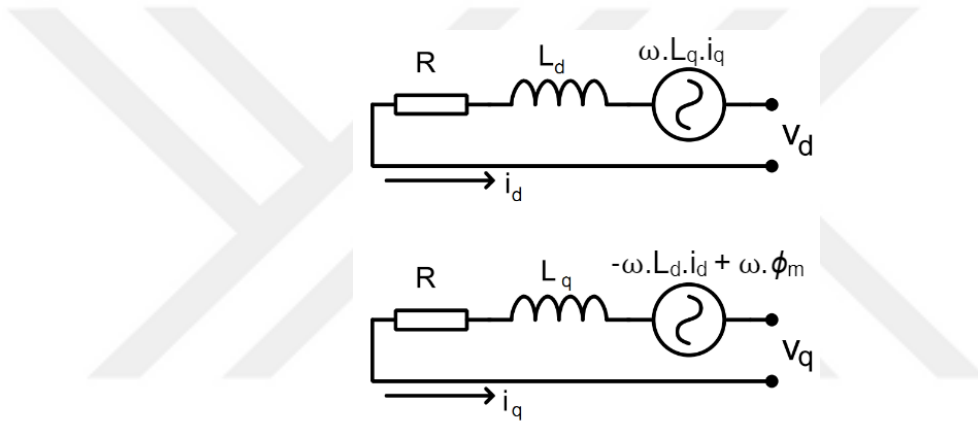


Figure 3.1: DQ axis electrical model of the PMSM.

The mechanical behavior of a PMSM can be described by Newton's second law for rotational systems, which equates the net torque acting on the rotor to the product of its moment of inertia and angular acceleration. The general form is:

$$J \frac{d\omega_r}{dt} = T_e - B\omega_r - T_L \quad (3.2)$$

where:

- J is the total moment of inertia of the rotor ($\text{kg} \cdot \text{m}^2$),
- ω_r is the mechanical angular velocity (rad/s),
- T_e is the electromagnetic torque generated by the motor (Nm),
- B is the viscous friction coefficient ($\text{N} \cdot \text{m} \cdot \text{s/rad}$),
- T_L is the load torque acting on the shaft (Nm).

The electromagnetic torque T_e for a surface-mounted or interior PMSM in the d-q reference frame is given by:

$$T_e = \frac{3}{2}P(\phi_m i_q + (L_q - L_d)i_d i_q) \quad (3.3)$$

where:

- P is the number of pole pairs,
- ϕ_m is the permanent magnet flux linkage (Wb),
- i_d, i_q are the d - and q -axis stator currents (A),
- L_d, L_q are the d - and q -axis inductances (H).

The first term $\phi_m i_q$, represents the torque generated due to the interaction between the permanent magnet field and the q -axis current. The second term $(L_q - L_d)i_d i_q$, appears only in interior PMSMs and represents the reluctance torque due to saliency (i.e., difference in d- and q -axis inductances). Since $L_q = L_d$ in the surface mounted PMSMs, this term becomes zero. The following assumptions are made before proceeding the controller design procedure:

- The effects of magnetic saturation and magnetic hysteresis are neglected.
- For brevity, $(\frac{d}{dt})$ will be denoted as $(\dot{})$ in future calculations.
- Motor parameters R , B , and J , as well as the applied torque disturbance T_L , are unknown or immeasurable, for adaptive backstepping design. The true values of these parameters are positive constants.
- $\tilde{R} = R - \hat{R}$, $\tilde{B} = B - \hat{B}$, $\tilde{J} = J - \hat{J}$ and $\tilde{T}_L = T_L - \hat{T}_L$ where the terms with $(\sim)$ are estimation errors, the terms with $(\hat{})$ are estimations, and the symbols without modifiers represent the true (actual) values of the Parameters.
- R, B, J , and T_L are uncertain/unknown and bounded, while the inductances L_d and L_q are known with small deviations.
- These parameters are expressed as: $R = R_N + \Delta_R$, $B = B_N + \Delta_B$, $T_L(t) = T_{L_N} + \Delta_{T_L}(t)$, $J = J_N + \Delta_J$, $L_d = L_{d_N} + \Delta_{L_d}$, $L_q = L_{q_N} + \Delta_{L_q}$ where $R_N, B_N, T_{L_N}, J_N, L_{d_N}$, and $L_{q_N} \in \mathbb{R}^+$ are known nominal values and $\Delta_R, \Delta_B, \Delta_J \in \mathbb{R}$, $\Delta_{T_L}(t), \Delta_{L_d}$, and Δ_{L_q}

$\Delta_{L_q} \in \mathbb{R}$ represent slowly time-varying bounded uncertainties/disturbances, with Δ_{L_d} and Δ_{L_q} being small deviations that can be ignored.

3.2. Backstepping and Adaptive Backstepping Controller Design

In this subsection, we will show the various proposed controller designs for PMSM. Firstly, we will introduce the backstepping controller design. Then, an adaptive backstepping controller is proposed for better handling uncertainties and disturbances. Stability analysis of the both proposed methods based on Lyapunov Stability Analysis is also realized. Lastly, a real-time adaptation mechanism based on Fuzzy Logic will be introduced.

3.2.1. Design and Stability Analysis of Backstepping Controller

Based on Equation (3.1), (3.2) and (3.3), the system dynamics can be re-defined as follows:

$$L_d \frac{di_d}{dt} = -Ri_d + \omega L_q i_q + v_d \quad (3.4)$$

$$L_q \frac{di_q}{dt} = -Ri_q - \omega L_d i_d + \omega \phi_m + v_q \quad (3.5)$$

$$J \frac{d\omega}{dt} = \frac{3}{2} P \phi_m i_q + \frac{3}{2} P (L_q - L_d) i_d i_q - B\omega - T_L \quad (3.6)$$

The main control goal is to guarantee that PMSM follows a desired speed trajectory despite parametric uncertainties throughout the electromechanical model (i.e., in mechanical and electrical models) and load disturbances. The PMSM's velocity $\omega(t)$ and the phase currents $i_d(t), i_q(t)$ are considered available states. Ensuring closed-loop stability is also crucial.

The speed tracking error term $e_\omega(t) \in \mathbb{R}$ is defined as:

$$e_\omega = \omega - \omega^* \quad (3.7)$$

where $\omega^*(t) \in \mathbb{R}$ is the reference speed. Multiplying the each term of equation (3.7) by J gives:

$$J\dot{e}_\omega = J\dot{\omega} - J\dot{\omega}^* \quad (3.8)$$

After taking the derivative of each side, we reach the following:

$$J\dot{e}_\omega = J\dot{\omega} - J\dot{\omega}^* \quad (3.9)$$

Substituting $J\dot{\omega}$ from the system dynamics in equation (3.6), gives:

$$J\dot{e}_\omega = \frac{3}{2}P\phi_m i_q + \frac{3}{2}P(L_q - L_d)i_d i_q - B\omega_r - T_L - J\dot{\omega}^* \quad (3.10)$$

Defining a variable Ω , which equals to:

$$\Omega = \frac{3}{2}P(L_q - L_d)i_d i_q - B\omega_r - T_L - J\dot{\omega}^* \quad (3.11)$$

The term Ω encapsulates the effect of nonlinear coupling, damping, load torque, and reference dynamics, allowing the separation of linearizable and uncertain components in the control design. Thus, equation (3.10) evolves into the following format:

$$J\dot{e}_\omega = \frac{3}{2}P\phi_m i_q + \Omega \quad (3.12)$$

As seen in equation (3.12), the virtual control input i_q^* should be introduced before proceeding. Thus, after summing and subtracting the term $\frac{3}{2}P\phi_m i_q^*$ into equation (3.12) and regrouping the related terms to get the error term of the virtual control input $(i_q - i_q^*)$, we can obtain the following equation:

$$J\dot{e}_\omega = \frac{3}{2}P\phi_m (i_q - i_q^*) + \Omega + \frac{3}{2}P\phi_m i_q^* \quad (3.13)$$

Let's define a variable $\eta = i_q - i_q^*$, thus, equation (3.13.) evolves into the following format:

$$J\dot{e}_\omega = \frac{3}{2}P\phi_m\eta + \Omega + \frac{3}{2}P\phi_m i_q^* \quad (3.14)$$

Now, we shall design a virtual control input i_q^* such that the speed tracking error dynamics are stabilized through backstepping, ultimately ensuring asymptotic convergence of the error:

$$i_q^* = \frac{2}{3P\phi_m} (-k_3 e_\omega - \Omega) \quad (3.15)$$

Substituting virtual control input that in equation above into equation (3.14), we observe the speed-tracking error dynamics equation as follows:

$$J\dot{e}_\omega = \frac{3}{2}P\phi_m\eta - k_3 e_\omega \quad (3.16)$$

Now, we can proceed with the current errors i_d and i_q . Starting d -axis current (i_d) error definition as $e_d = i_d - i_d^*$, and after taking derivative of both side, gives:

$$\dot{e}_d = \dot{i}_d - \dot{i}_d^* \quad (3.17)$$

where i_d^* is reference for i_d and it is kept as zero ($i_d^* = 0$). By multiplying each side with L_d , equation becomes:

$$L_d \dot{e}_d = L_d \dot{i}_d \quad (3.18)$$

Substituting $L_d \dot{i}_d$ from the system dynamics in equation (3.4) results:

$$L_d \dot{e}_d = v_d - R e_d + \omega L_q i_q \quad (3.19)$$

v_d as the virtual control input is chosen as follows:

$$v_d = -k_1 e_d - \omega L_q i_q \quad (3.20)$$

By substituting virtual control input v_d in equation (3.20) into equation (3.19), the d -axis current error dynamics is obtained as follows:

$$L_d \dot{e}_d = -(k_1 + R)e_d \quad (3.21)$$

Similarly, we can define q -axis current error by $e_q = i_q - i_q^*$. After taking derivative and multiplying the both side by L_q , the equation becomes:

$$L_q \dot{e}_q = L_q \dot{i}_q - L_q \dot{i}_q^* \quad (3.22)$$

Substituting $L_q \dot{i}_q$ from the system dynamics in equation (3.5) results:

$$L_q \dot{e}_q = -Ri_q - \omega L_d i_d + \omega \phi_m + v_q - L_q \dot{i}_q^* \quad (3.23)$$

Substituting the derivative of the virtual control input i_q^* into equation (3.23), the following equation is reached:

$$L_q \dot{e}_q = v_q - Ri_q - \omega L_d i_d + \omega \phi_m - L_q \left(\frac{2}{3P\phi_m} (-k_3 \dot{e}_\omega - \dot{\Omega}) \right) \quad (3.24)$$

where the term $\dot{\Omega}$ is used to represent the time derivative of the term Ω previously defined. Note that the calculation of $\dot{\Omega}$ requires the time derivative of the terms i_d, i_q and $\dot{\omega}$ where equations (3.4), (3.5), and time derivative of equation (3.6) are used.

Substituting derivative of e_ω from (3.12) into (3.24), the equation becomes:

$$\begin{aligned} L_q \dot{e}_q = & v_q - Ri_q - \omega L_d i_d + \omega \phi_m \\ & + \frac{2L_q}{3P\phi_m} \left(-k_3 \left(\frac{3P}{2J} \phi_m i_q + \frac{3P}{2} (L_q - L_d) i_q i_d - B\omega_r - T_L \right. \right. \\ & \left. \left. - J\dot{\omega}^* \right) + \dot{\Omega} \right) \end{aligned} \quad (3.25)$$

At this stage, to reduce complexity and derive the virtual control input v_q more straightforwardly, the nonlinear terms and feedback terms in equation (3.25) can be represented separately as follows:

$$L_q \dot{e}_q = v_q - U_1 + U_2 \quad (3.26)$$

where U_1 and U_2 are defined as follows, respectively:

$$U_1 = -Ri_q - \omega L_d i_d + \omega \phi_m \quad (3.27)$$

$$U_2 = \frac{2L_q}{3P\phi_m} (\Delta + \dot{\Omega}) \quad (3.28)$$

where Δ consists of feedback and error terms. Now, we can choose a virtual control input v_q that $L_q \dot{e}_q$ converges to zero, as follows:

$$v_q = U_1 - U_2 - k_2 \eta - \frac{3}{2} P \phi_m e_\omega \quad (3.29)$$

where $\frac{3}{2} P \phi_m e_\omega$ is a cross term. By substituting (3.29) into (3.26), we finally observe the error dynamics as follows:

$$L_q \dot{\eta} = -k_2 \eta - \frac{3}{2} P \phi_m e_\omega \quad (3.30)$$

After all error dynamics are introduced, stability analysis of the design can be employed. Now, a Lyapunov stability function that will represent the total virtual energy of the system is chosen as follows:

$$V = \frac{1}{2} L_d e_d^2 + \frac{1}{2} J e_\omega^2 + \frac{1}{2} L_q \eta^2 \quad (3.31)$$

After taking derivative of (3.31) and substituting (3.16), (3.21), and (3.30) into it, the following equation is obtained:

$$\dot{V} = -(k_1 + R) e_d^2 - k_3 e_\omega^2 + \frac{2P}{3} \phi_m \eta e_\omega - k_2 \eta^2 - \frac{2}{3} P \phi_m \eta e_\omega \quad (3.32)$$

In equation (3.32), after cross terms cancel each other, the derivative function becomes $\dot{V} = -(k_1 + R) e_d^2 - k_3 e_\omega^2 - k_2 \eta^2$. All gain parameters satisfy $k_1, k_2, k_3 \geq 0$, the derivative is negative semi-definite. Hence, $V(t)$ is non-increasing and all closed-loop signals remain bounded. Moreover, because $\dot{V}(t)$ is continuous and the error terms are square integrable, Barbalat's Lemma guarantees that $e_\omega \rightarrow 0$, $e_d \rightarrow 0$ and $e_q \rightarrow 0$ as

$t \rightarrow \infty$. Therefore, the equilibrium point of the closed-loop system is globally asymptotically stable [44].

3.2.2. Design and Stability Analysis of the Adaptive Backstepping Control

The addressed backstepping control design demonstrates superior performance when all parameters are known or measurable; however, it may face uncertainties. In such uncertain conditions, opting for an adaptive backstepping becomes a viable approach. This technique combines backstepping control with adaptive laws, enhancing robustness for systems dealing with matched and mismatched uncertainties. To uphold the stability of the overall control system, the introduction of a Lyapunov function becomes essential. This function guarantees the efficient tracking of the desired signal for the angular speed of the PMSM, effectively counteracting the effects of parameter uncertainties [45], as we can witness in the simulation results section.

We can start the adaptive backstepping controller design by speed-tracking error definition $J\dot{e} = J\dot{\omega} - J\dot{\omega}^*$, previously introduced. Similarly, we will use equation (3.6) which is derived by substituting $J\dot{\omega}$ from system dynamics. We are adding the term $\pm \frac{3}{2}P\phi_m i_q^*$ into equation (3.10) will form the equation as follows:

$$J\dot{e} = \frac{3}{2}P\phi_m e_q + \frac{3}{2}P(L_q - L_d)i_d i_q - B\omega_r - T_L - J\omega^* + \frac{3}{2}P\phi_m i_q^* \quad (3.33)$$

where $e_q = i_q - i_q^*$. As highlighted in the earlier sections, we'll proceed with the assumption that B, T_L and J are the uncertain or immeasurable parameters. We can rewrite the uncertain part in a matrix form as:

$$Y_1\Phi = [\omega_r \quad 1 \quad \omega^*][B \quad T_L \quad J]^T \quad (3.34)$$

If we replace equation (3.34) into equation (3.33), we get:

$$J\dot{e} = \frac{3}{2}P\phi_m e_q + \frac{3}{2}P(L_q - L_d)i_d i_q - Y_1\Phi + \frac{3}{2}P\phi_m i_q^* \quad (3.35)$$

We choose a suitable virtual control input i_q^* such that $J\dot{e}$ may converge to zero, as follows:

$$i_q^* = \frac{2}{3P\phi_m} \left(-k_1 e - \frac{3}{2} P(L_q - L_d) i_d i_q + Y_1 \hat{\Phi} \right) \quad (3.36)$$

where $\hat{\Phi}$ is the matrix of estimated parameters. Substituting the virtual control input equation i_q^* in (3.36) into (3.35) and performing the necessary manipulations, we obtain the first error dynamics for the adaptive backstepping controller design as follows:

$$J\dot{e} = -k_1 e - Y_1 \tilde{\Phi} + \frac{3}{2} P\phi_m e_q \quad (3.37)$$

where $\tilde{\Phi}$ is estimation error matrix which equals to $\Phi - \hat{\Phi}$.

For current loop controller design, we can use the error definition of e_q , which is previously introduced as $L_q \dot{e}_q = L_q \dot{i}_q - L_q \dot{i}_q^*$ in equation (3.22). After substituting $L_q \dot{i}_q$ from the system dynamics equation in (3.5) and substituting the derivative of virtual control input i_q^* , following equation is reached:

$$\begin{aligned} L_q \dot{e}_q = & -Ri_q - \omega L_d i_d + \omega \phi_m + v_q + \frac{2L_q}{3P\phi_m} k_1 \dot{e} \\ & + \frac{L_q}{\phi_m} (L_q - L_d) (\dot{i}_d i_q + i_d \dot{i}_q) + \frac{2L_q}{3P\phi_m} (Y_1 \dot{\hat{\Phi}}) \end{aligned} \quad (3.38)$$

By substituting the derivatives of e , i_d and i_q into equation (3.38), the following equation can be obtained as follows:

$$\begin{aligned} L_q \dot{e}_q = & -Ri_q - \omega L_d i_d + \omega \phi_m + v_q \\ & + \frac{2L_q}{3P\phi_m} k_1 \left(\frac{1}{J} \left(\frac{3}{2} P\phi_m i_q + \frac{3}{2} P(L_q - L_d) i_d i_q - B\omega_r - T_L - J\omega^* \right) \right. \\ & + \frac{L_q}{\phi_m} (L_q - L_d) \left(\frac{1}{L_d} (-Ri_d + \omega L_q i_q + v_d) i_q \right. \\ & \left. \left. + i_d \left(\frac{1}{L_q} (-Ri_q - \omega L_d i_d + \omega \phi_m + v_q) \right) \right) \right) + \frac{2L_q}{3P\phi_m} (Y_1 \dot{\hat{\Phi}}) \end{aligned} \quad (3.39)$$

Since equation (3.39) seems complicated, we can reshape this equation in the following format:

$$L_q \dot{e}_q = U_1 + Y_2 \theta + v_q \quad (3.40)$$

where U_1 represents all the known or measurable parameters, while θ consists the unknowns as follows, respectively:

$$U_1 = -\omega L_d i_d + \omega \phi_m - \frac{2L_q k_1 \dot{\omega}^*}{3P\phi_m} + \frac{(L_q - L_d)i_d}{\phi_m} (-\omega L_d i_d + \omega \phi_m) + \frac{L_q(L_q - L_d)i_q}{\phi_m L_d} (\omega L_q i_q + v_d) + \frac{2L_q}{3P\phi_m} (Y_1 \dot{\hat{\phi}}) \quad (3.41)$$

$$Y_2 \theta = \begin{bmatrix} -i_q - \frac{i_d i_q L_q (L_q - L_d)}{\phi_m L_d} - \frac{i_d i_q (L_q - L_d)}{\phi_m} \\ L_q i_q k_1 \frac{(\phi_m + (L_q - L_d)i_d)}{\phi_m} \\ \frac{-2L_q k_1 \omega_r}{3P\phi_m} \\ \frac{-2L_q k_1}{3P\phi_m} \end{bmatrix}^T \begin{bmatrix} R \\ A \\ b \\ \tau \end{bmatrix} \quad (3.42)$$

where $A = \frac{1}{j}$, $b = \frac{B}{j}$ and $\tau = \frac{TL}{j}$. Now, it is time to choose a proper control input v_q for equation (3.40) such that $L_q \dot{e}_q$ may converges to zero. Thus, control input v_q will be obtained as follows:

$$v_q = -k_2 e_q - Y_2 \hat{\theta} - U_1 - \frac{3}{2} P \phi_m e \quad (3.43)$$

where $\hat{\theta}$ is estimation of the unknown parameters and $\frac{3}{2} P \phi_m e$ is a cross term. By substituting control input v_q into equation (3.40), we get q -axis current error dynamics as follows:

$$L_q \dot{e}_q = -k_2 e_q + Y_2 \tilde{\theta} - \frac{3}{2} P \phi_m e \quad (3.44)$$

where $\tilde{\theta}$ is the error matrix which equals to $\theta - \hat{\theta}$.

For the last error dynamics e_d , we can use the previously defined equation $L_d \dot{e}_d = L_d \dot{i}_d$, (recall that $L_d \dot{i}_d^* = 0$). If we substitute $L_d \dot{i}_d$ from the system dynamics in equation (3.4), the following equation is reached:

$$L_d \dot{e}_d = -R i_d + \omega L_q i_q + v_d \quad (3.45)$$

Considering the previous assumptions, that the stator resistance R is unknown or immeasurable, the control input v_d can be chosen as follows:

$$v_d = -k_3 i_d - \omega L_q i_q + \hat{R} i_d \quad (3.46)$$

By substituting virtual control input v_d in equation (3.46) into equation (3.45), we can get the error dynamics as

$$L_d \dot{e}_d = -k_3 i_d - \tilde{R} i_d \quad (3.47)$$

where $\tilde{R}$ is the error term which equals to $R - \hat{R}$.

Now that all three error dynamics are determined, we can proceed to the stability analysis of the proposed adaptive backstepping controller design. By leveraging Lyapunov's Stability Theory, we aim to demonstrate that the closed-loop system is stable and that the tracking errors converge to zero as time progresses. For this purpose, we choose a suitable Lyapunov function as follows:

$$V = \frac{1}{2} \tilde{\Phi}^T \Gamma_1^{-1} \tilde{\Phi} + \frac{1}{2} \tilde{\theta}^T \Gamma_2^{-1} \tilde{\theta} + \frac{1}{2} \gamma^{-1} \tilde{R}^2 + \frac{1}{2} L_d e_d^2 + \frac{1}{2} L_q e_q^2 + \frac{1}{2} J e^2 \quad (3.48)$$

After taking derivative of V and by using the property of $(\dot{\sim}) = -(\dot{\hat{\sim}})$, we can obtain the following equation:

$$\dot{V} = J \dot{e} + \tilde{\Phi}^T \Gamma_1^{-1} (-\dot{\hat{\Phi}}) + L_q \dot{e}_q e_q + \tilde{\theta}^T \Gamma_2^{-1} (-\dot{\hat{\theta}}) + L_d \dot{e}_d i_d + \gamma^{-1} (-\dot{\hat{R}}) \tilde{R} \quad (3.49)$$

Substituting the closed loop dynamics $J \dot{e}$, $L_q \dot{e}_q$ and $L_d \dot{e}_d$ from equations (3.37), (3.44) and (3.47) respectively into (3.49), we get the following relationship:

$$\begin{aligned}
\dot{V} = & \left(\frac{3}{2}P\phi_me_q - k_1e - Y_1\tilde{\Phi}\right)e - \tilde{\Phi}^T\Gamma_1^{-1}\dot{\hat{\Phi}} \\
& + (-k_2e_q + Y_2\tilde{\theta} - \frac{3}{2}P\phi_me)e_q - \theta^T\Gamma_2^{-1}\dot{\hat{\theta}} \\
& + (-k_3i_d - \tilde{R}i_d - \gamma^{-1}\tilde{R}(\dot{\hat{R}}))
\end{aligned} \tag{3.50}$$

For equation (3.50), the adaptation laws can be determined by the following expressions:

$$\dot{\hat{\Phi}} = -\Gamma_1 Y_1^T e \tag{3.51}$$

$$\dot{\hat{\theta}} = \Gamma_2 Y_2^T e_q \tag{3.52}$$

$$\dot{\hat{R}} = -\gamma e_d^2 \tag{3.53}$$

where $\Gamma_1 \in \mathbb{R}^{3 \times 3}$ and $\Gamma_2 \in \mathbb{R}^{4 \times 4}$ are diagonal, positive-definite, gain matrices and γ is the positive definite scalar adaptation gain parameter. By substituting these adaptive laws from equations (3.51), (3.52) and (3.53) into equation (3.50) and by simplifying it, we obtain the following:

$$\dot{V} = -k_1e^2 - k_2e_q^2 - k_3i_d^2 \tag{3.54}$$

Therefore, for the positive values of k_1 , k_2 and k_3 , the closed system is asymptotically stable, and the tracking errors e , e_q and e_d will converge to zero as $t \rightarrow \infty$, according to Barbalat's Lemma. Barbalat's Lemma indicates that if the differentiable function V has a finite limit as $t \rightarrow \infty$, and if it is uniformly continuous then $\dot{V} \rightarrow 0$ as $t \rightarrow \infty$.

3.2.3. Real Time Parameter Selection Mechanism based on Fuzzy Logic

The backstepping and the adaptive backstepping control strategies presented in the previous sections are designed to ensure the stability of PMSM drives under disturbances and parametric uncertainties. However, the use of fixed adaptive gains in the standard adaptive backstepping controller can limit the dynamic performance of the speed-tracking loop. Specifically, static gain values may hinder the system's ability

to rapidly respond to varying operating conditions. To overcome this limitation, we propose the integration of a FL as a soft-computing based tuning mechanism.

By enabling real-time adjustment of the adaptive gains based on the system's tracking error dynamics, the FL enhances the adaptability and robustness of the PMSM FOC system. This dynamic gain tuning allows the tracking error to converge more rapidly within a narrow bound, thereby improving overall control performance.

In the proposed design, as seen in **Figure 3.2**, the FL controller takes rotor speed error e and change in rotor speed error de as input and adjust the adaptive gain parameters Γ_1 and k_{a1} . Since we aim to improve the speed-tracking error of the proposed controller under disturbances, the variables related to the speed control loop will be produced at the output of the fuzzy logic parameter selection algorithm. Output FLC parameters of $\Delta\Gamma_1$ and Δk_{a1} range from $-0.4\Gamma_1^*$ to $0.4\Gamma_1^*$ and from $-0.1\Delta k_{a1}^*$ to $0.1\Delta k_{a1}^*$ respectively, where Γ_1^* and k_{a1}^* are relative optimal parameters accordingly.

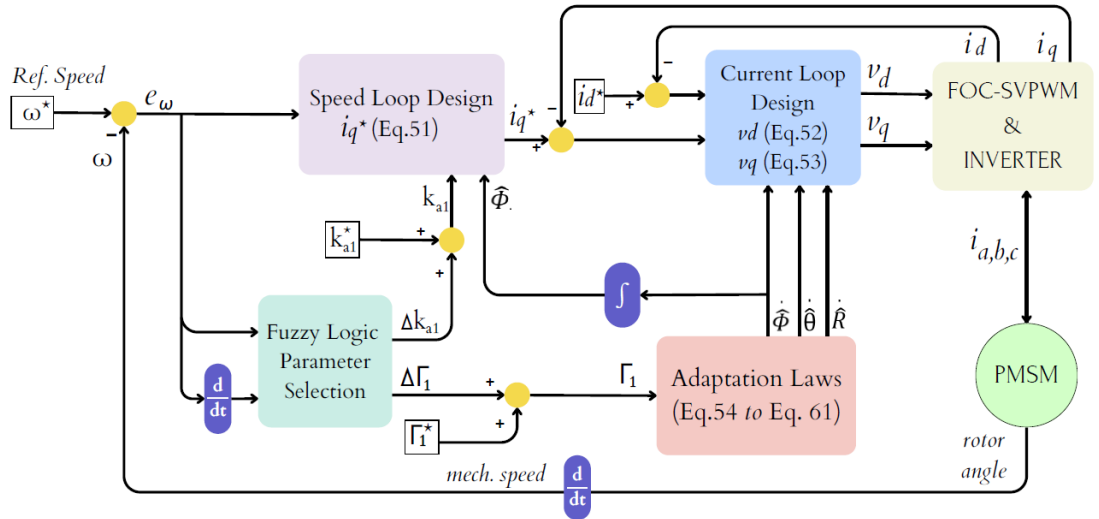


Figure 3.2: FLC real-time parameter selection scheme for ABC.

The fuzzy variables are categorized as follows: Negative large (NL), negative medium (NM), negative small (NS), zero (ZO), positive small (PS), positive medium (PM), and positive large (PL). **Table 3.1** indicates the rules of FLC for $\Delta\Gamma_1$ and Δk_{a1} , and triangular membership functions are employed for all fuzzy sets for this study. It should be noted that this structure is designed for comparison with the performances of other proposed controllers in simulation results without being included in the stability analysis.

Table 3.1: FL rule base for parameters k_{a1} and Γ_1 .

e / de	NL	NM	NS	ZO	PS	PM	PL
NL	PL	PL	PM	PM	PS	ZO	ZO
NM	PL	PL	PM	PS	PS	ZO	NS
NS	PM	PM	PM	PS	ZO	NS	NS
ZO	PM	PM	PS	ZO	NS	NM	NM
PS	PS	PS	ZO	NS	NS	NM	NM
PM	PS	ZO	NS	NM	NM	NM	NL
PL	ZO	ZO	NM	NM	NM	NL	NL

The FL component in this study serves as a real-time parameter-tuning mechanism for the adaptive backstepping controller, focusing on enhancing robustness without over-complicating the control structure. The FL mechanism dynamically adjusts the adaptation gains (Γ_1 and k_{a1}) based on speed tracking error, $e(t)$, and time derivative of the error signal, $\dot{e}(t)$, acting as a supervisory layer over the core adaptive laws. The rule base and membership functions were streamlined for computational efficiency and real-time feasibility, avoiding over-specialization to fuzzy logic theory.

3.2.4. Performance Evaluation of the Proposed Controllers

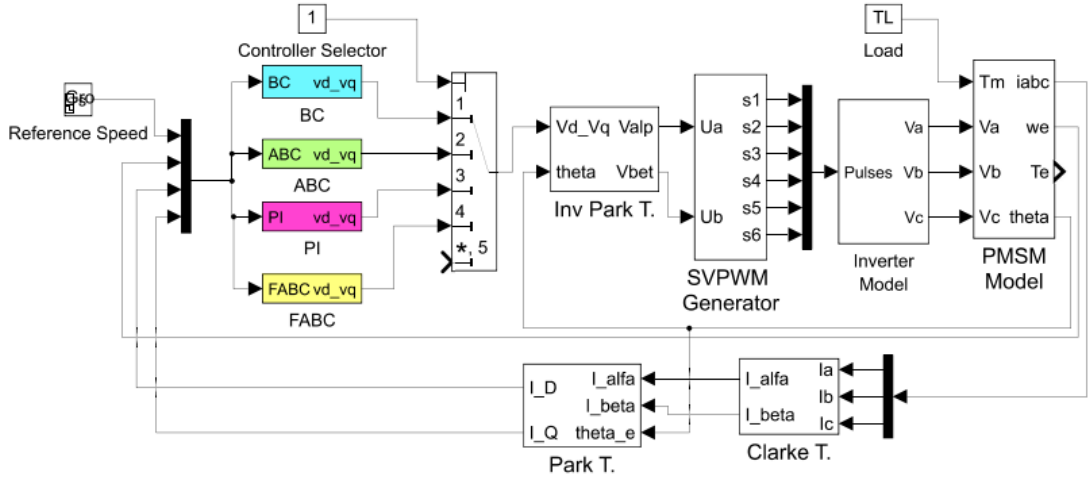
To evaluate the effectiveness of the proposed controller designs, the controllers and PMSM FOC driver are implemented in MATLAB/Simulink environment. The PMSM parameters used in the simulation are presented in **Table 3.2**.

The computational complexity of the proposed method arises from the adaptive backstepping control laws and the fuzzy logic parameter tuning. For adaptive backstepping, the primary operations involve solving Lyapunov-based error dynamics in equations (3.37), (3.44), (3.47) and updating parameter estimates in equations (3.51), (3.52), (3.53) which require matrix multiplications with small-dimension gain matrices. See the appendix-A for all the equations in their open forms which were used in simulation study. These operations are computationally efficient because of the diagonal structure of adaptive gain matrices, reducing the arithmetic complexity.

Table 3.2: PMSM parameters used in FABC simulation.

Item	Quantity
Stator Resistance (R)	1.2 Ohm
Inductance (L_q)	0.0076 H
Inductance (L_d)	0.0072 H
PM flux linkage (ϕ_m)	0.455 V/rad/s
Moment of Inertia (J)	0.0086 kgm ²
Viscous Friction Coeff. (B)	0.0005 Nm/rad/s
Number of Pole Pairs (P)	4

For the fuzzy logic component, the 49-rule base (Table 3.1) and triangular membership functions involve lightweight real-time calculations, which are well-suited for embedded systems. Compared to a standard PI controller, the proposed method incurs a modest additional computational load but remains feasible for real-time implementation, as validated by the simulation sampling period ($T_s=0.0001$). Modern microcontrollers or DSPs, commonly used in motor drives, can readily handle this complexity while maintaining stability and performance.

**Figure 3.3:** Matlab block diagram of the proposed system.

A comparative simulation study is carried out to evaluate the effectiveness of the proposed control strategy. The performance of four controllers—namely, the Proportional-Integral (PI) controller, Backstepping Controller (BC), Adaptive Backstepping Controller (ABC), and Adaptive Backstepping Controller with Fuzzy Parameter Tuning (FABC)—is systematically assessed under identical system

conditions. The aim is to quantitatively and qualitatively compare these control approaches using rigorous simulation metrics. For conciseness, the design methodology of the PI controller is not elaborated, as it follows standard industry practices and serves primarily as a benchmark for comparison.

The selected parameters for the PI controller are as follows: in the speed control loop, the proportional and integral gains are set at $K_p = 16$ and $K_i = 6$, respectively; in the v_q control loop, the proportional and integral gains are $K_p = 19$ and $K_i = 5$, respectively; and in the v_d control loop, the proportional gain is $K_p = 12$ and the integral gain is $K_i = 2$. The gain parameters for the backstepping controller are specifically tuned to attain control input amplitudes comparable to those exhibited by the Proportional-Integral (PI) controller. Consequently, the backstepping controller's gain parameters are adjusted as follows: $k_1 = 10$, $k_2 = 60$ and $k_3 = 600$. For adaptive backstepping gain parameters and adaptation gain matrix values are $k_{a1} = 20$, $k_{a2} = 200$, $k_{a3} = 2000$, $\Gamma_1 = \text{diag}\{0.0001, 0.002, 600\}$ and $\Gamma_2 = \text{diag}\{0.0002, 0.03, 0.002, 0.001\}$. SVPWM generator parameters are $V_{dc} = 320V$ and sampling period $T_s = 0.0001s$. The overall Matlab/Simulink model for the comparative study is illustrated in **Figure 3.3**.

The backstepping control gains (k_1, k_2, k_3) are tuned using the Lyapunov stability criterion to ensure error convergence. The initial values match the control effort of the baseline PI controller, then iteratively refined in simulations. Larger k_i values reduce steady-state error but may amplify noise sensitivity or induce overshoot. The diagonal entries of the adaptation gains Γ_1 and Γ_2 are empirically tuned. Smaller gains ensure smooth adaptation; larger values accelerate convergence. Aggressive gains ($> 10^3$) destabilize the system, while conservative gains ($< 10^{-3}$) slow disturbance rejection. Finally, the 49-rule base maps of the FLC setup take as input the speed error (e) and its derivative ($\dot{e}$) to adjust the parameters $\Delta\Gamma_1$ and Δk_{a1} . Wider membership ranges accommodate uncertainties but reduce precision. Rules prioritizing "Positive Large" outputs enable rapid transient adjustments.

Two different scenarios are considered to evaluate the performance of the proposed controllers. In the first case, the reference speed starts at 800 rad/s and suddenly increases to 1200 rad/s at $t = 0.5s$, while the applied torque remains constant throughout the simulation. This case aims to assess both transient responses and

steady-state performance. In the second case, the speed is maintained at a continuous 800 rad/s, but the applied torque increases from 6 Nm to 30 Nm at $t = 0.5s$, allowing for an evaluation of the controllers' disturbance rejection capabilities.

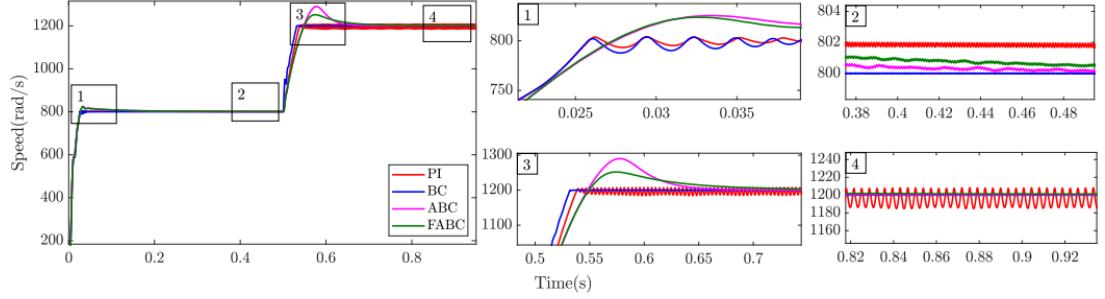


Figure 3.4: The speed response of the controllers for case 1.

Figure 3.4 presents the speed-tracking performance of all controllers under the first test scenario. During the moderate-speed phase (800 rad/s), the PI controller exhibits minimal overshoot ($< 0.5\%$) but suffers from a noticeable steady-state error compared to other controllers. The backstepping controller (BC) achieves a smoother response with negligible overshoot ($< 0.4\%$) and effectively eliminates steady-state error. While the adaptive backstepping (ABC) and fuzzy adaptive backstepping (FABC) controllers do not markedly outperform the PI or BC methods at this speed range, their advantages emerge during the high-speed phase (1200 rad/s). Here, the PI controller, constrained by fixed gains, exhibits a larger steady-state error and oscillatory behavior (1180–1210 rad/s). In contrast, ABC and FABC maintain superior tracking accuracy despite system uncertainties. Notably, FABC achieves a significantly lower overshoot than ABC (4.1% vs. 8.3%) at 1200 rad/s, whereas ABC provides a faster settling time (0.15 s vs. 0.25 s). These results demonstrate that the proposed adaptive controllers ensure robust and precise speed tracking across diverse operating conditions and uncertainties.

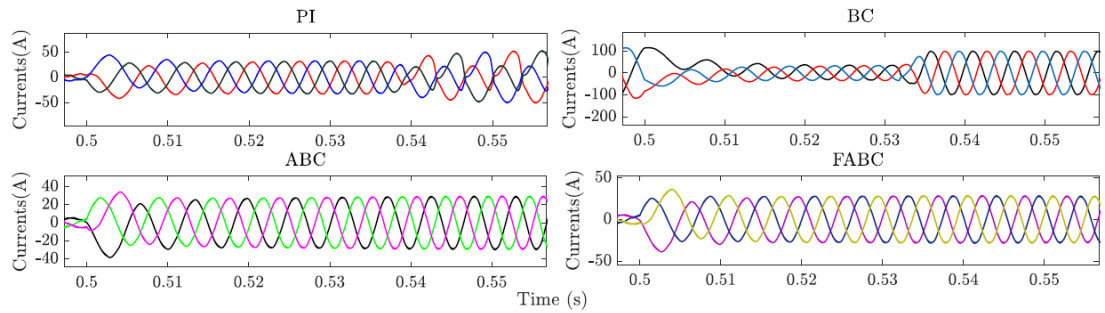


Figure 3.5: Three phase stator currents for case 1.

The motor's three-phase currents, load torque response, and i_d and i_q current plots are shown in **Figure 3.5**, **Figure 3.6**, and **Figure 3.7**, respectively. In **Figure 3.5**, the PI controller exhibits pronounced harmonic distortion and asymmetric current waveforms during transients, indicative of its limited adaptability to dynamic load and speed variations. In contrast, the backstepping controller (BC) demonstrates reduced current ripple and improved symmetry, leveraging its nonlinear feedback structure. The adaptive backstepping (ABC) and fuzzy adaptive backstepping (FABC) controllers further enhance current regulation, with FABC achieving the smoothest and most balanced phase currents due to its real-time fuzzy-based gain adjustments. These observations validate the proposed methods' ability to maintain stable and efficient current profiles under abrupt operational changes, minimizing energy losses and mechanical stress. As seen in **Figure 3.6**, the PI controller exhibits significant torque oscillations (5–70 Nm range) during the speed step change, reflecting poor disturbance rejection. In contrast, the adaptive backstepping (ABC) and fuzzy adaptive backstepping (FABC) controllers stabilize torque output with minimal ripple (< 10%), demonstrating superior robustness to parametric uncertainties. FABC further reduces transient spikes, leveraging fuzzy logic to adjust adaptation gains dynamically.

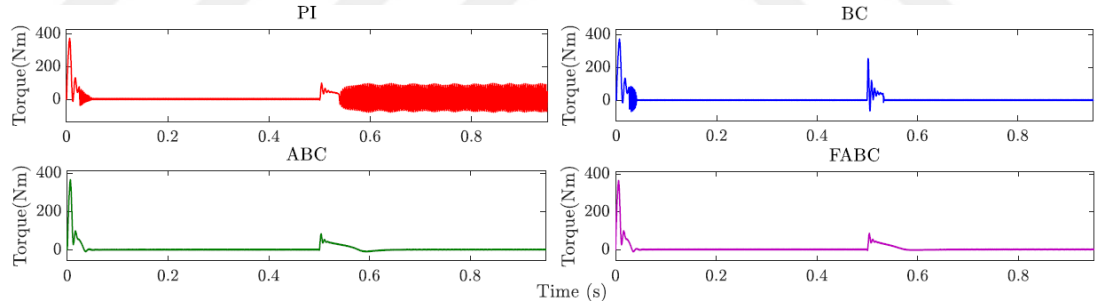


Figure 3.6: The produced torques by the motor for case 1.

Figure 3.7 shows that i_q (torque-producing current) under PI control shows erratic fluctuations during transients, while ABC and FABC maintain smoother trajectories, ensuring precise torque regulation. Notably, FABC minimizes i_d deviations (kept near zero), adhering to field-oriented control principles and enhancing efficiency. Both adaptive methods outperform BC and PI in decoupling i_q and i_d dynamics, with FABC achieving the most stable current profiles under disturbances.

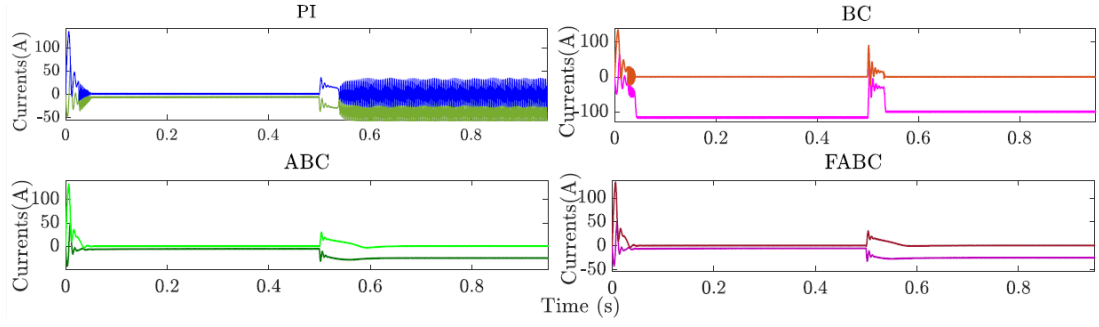


Figure 3.7: The currents i_d and i_q for case 1.

In the second test scenario, a time-varying load torque is applied uniformly across all systems to evaluate the disturbance rejection capabilities of the respective controllers. Throughout the simulation, the speed reference is held constant at 800rad/s. Initially, a load torque of 6Nm is applied; at $t=0.5$, this torque is abruptly increased fivefold to 30Nm. This step change serves to assess the controllers' ability to maintain speed regulation in the presence of significant external disturbances.

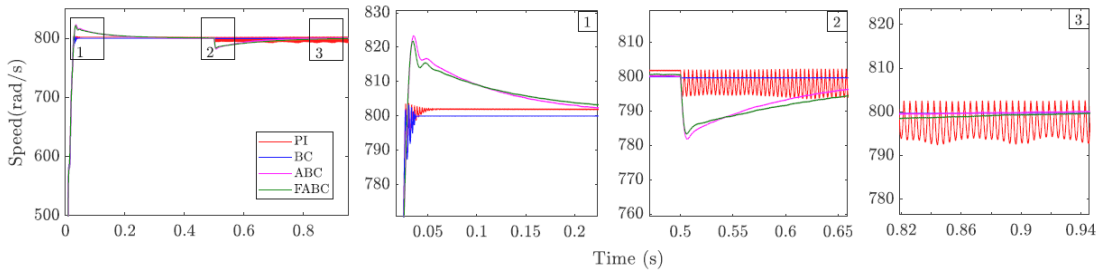


Figure 3.8: The speed-tracking performances for case 2.

Figure 3.8 illustrates the speed tracking performance of four controllers—PI, BC, ABC, and FABC—under a constant reference of 800 rad/s and a variable load torque, evaluated across three key operating intervals: startup, disturbance response, and steady-state. During the initial 6Nm load startup phase (Region 1), the PI controller achieves the fastest rise time but suffers from significant overshoot and pronounced oscillations. In contrast, the BC controller exhibits superior damping characteristics with minimal overshoot. While ABC and FABC initially display higher overshoot compared to BC, they settle more smoothly and consistently. At $t=0.5$ s, a 30Nm load disturbance is introduced (Region 2), revealing notable performance distinctions. The PI controller responds poorly, exhibiting sustained high-frequency oscillations and a steady-state error of approximately 12rad/s. The FABC and ABC controllers demonstrate superior transient response, with overshoots of 1.8% and 2.3%, respectively, and settling times below 0.3s. Additionally, both adaptive strategies

nearly eliminate steady state error (less than 0.5 rad/s), indicating robust disturbance rejection capabilities. In the final steady-state region (Region 3), the FABC controller provides the smoothest response with minimal deviation from the set point, while the PI controller continues to oscillate and underperform. Although the BC controller also achieves relatively fast and stable behavior, it relies on complete system model knowledge. In contrast, the ABC and FABC controllers operate under partial information by estimating uncertain parameters online, offering a more practical and resilient solution in scenarios with modeling uncertainties. Overall, the FABC controller demonstrates the most balanced and robust performance across all operating regions, excelling in tracking accuracy, disturbance rejection, and adaptability, followed closely by ABC and BC, with PI consistently lagging in all assessed metrics.

Table 3.3: Quantitative performance comparison of the controllers.

Metric	PI	BC	ABC	FABC
RMSE	61.89	61.38	60.12	60.01
IAE	11.86	11.23	7.90	7.88
Overshoot (%)	2.72	2.92	0.38	0.39
Settling Time (s)	0.96	0.96	0.91	0.89
Rise Time (s)	0.01837	0.01823	0.01709	0.01701

Table 3.3 provides a quantitative comparison of the four controllers—PI, BC, ABC, and FABC—based on five key performance metrics: RMSE, IAE, overshoot percentage, settling time, and rise time. The FABC controller demonstrates the best overall performance, achieving the lowest RMSE (60.01) and minimum IAE (7.88), indicating superior tracking precision and minimal accumulated error. Closely following, the ABC controller also shows excellent accuracy with an RMSE of 60.12 and IAE of 7.90. Both adaptive controllers exhibit exceptionally low overshoots, with 0.39% for FABC and 0.38% for ABC, which are significantly lower than those observed in PI (2.72%) and BC (2.92%). In terms of dynamic response, FABC also achieves the fastest settling time (0.89 s) and shortest rise time (0.01701 s), indicating rapid and stable convergence to the setpoint. The ABC controller performs similarly well, with a rise time of 0.01709 s and settling time of 0.91 s.

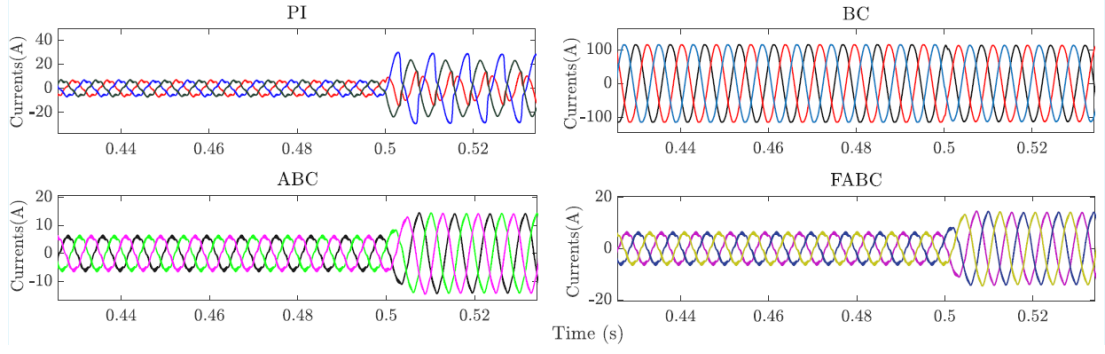


Figure 3.9: Three phase stator currents for case 2.

The BC and PI controllers show slower response times and higher error metrics. While BC performs better than PI across most metrics, it still exhibits higher RMSE (61.38) and IAE (11.23) than the adaptive approaches. Notably, the PI controller ranks lowest in all categories, with the highest RMSE (61.89), highest IAE (11.86), and substantial overshoot. **Figure 3.9** illustrates the three-phase stator currents of the PMSM, while **Figures 3.10** and **Figure 3.11** present the electromagnetic torque and the i_d , i_q current responses, respectively, for the second simulation scenario involving a torque disturbance. As shown in **Figure 3.9**, the stator currents controlled by the PI controller exhibit significant oscillations following the application of the load disturbance at $t = 0.5s$.

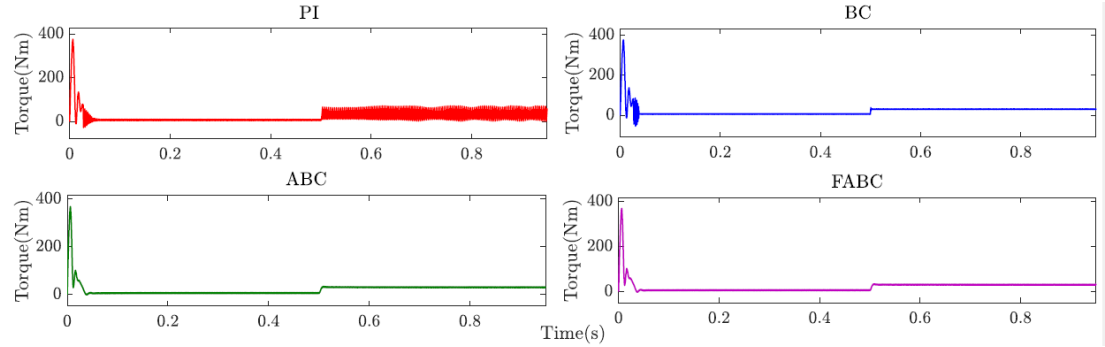


Figure 3.10: The produced torque by the motor for case 2.

In **Figure 3.10**, the torque output under PI control fluctuates markedly between $5Nm$ and $70Nm$, whereas the backstepping, adaptive backstepping (ABC), and fuzzy adaptive backstepping (FABC) controllers maintain smoother torque profiles, with torque ripple remaining below 10%. A similar trend is observed in the i_d and i_q current responses shown in **Figure 3.11**, where the proposed adaptive controllers demonstrate effective damping and stability. Notably, the backstepping controller provides the fastest response, though it assumes full knowledge of system parameters. In contrast,

the ABC and FABC controllers achieve comparably strong performance despite estimating parameters online.

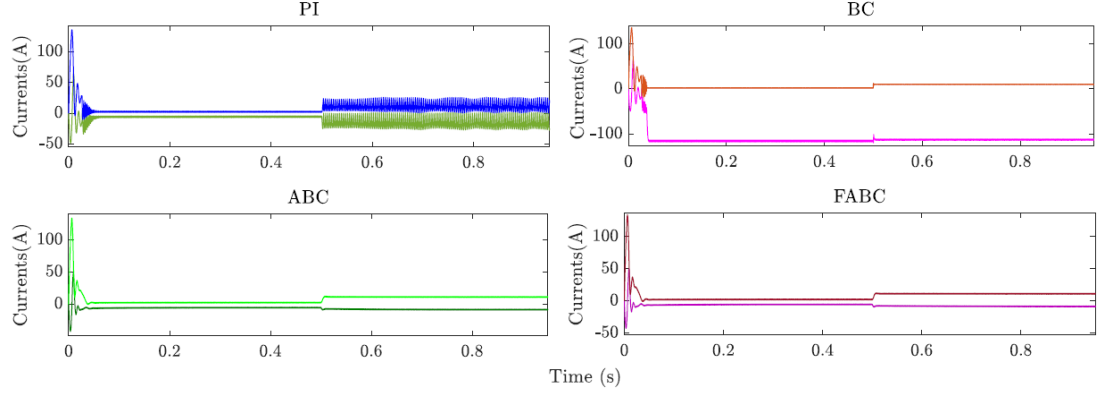


Figure 3.11: The currents i_d and i_q for case 2.

These results confirm that the proposed controllers handle torque disturbances more effectively than the conventional PI controller, particularly in the presence of system uncertainties.

Figure 3.12 illustrates the parameter estimation performance of the proposed control scheme, presenting the real-time estimates of the stator resistance ($\hat{R}$), rotor moment of inertia ($\hat{J}$), viscous friction coefficient ($\hat{B}$), and disturbance torque ($\hat{T}_L$). All estimated parameters exhibit an initial transient phase followed by smooth convergence to their respective steady-state values. This behavior confirms the effectiveness of the adaptive mechanism in compensating for both system uncertainties and external disturbances. Furthermore, the convergence of the parameter estimates closely correlates with the reduction in tracking error, underscoring the consistency and robustness of the proposed adaptation laws. The absence of high-frequency oscillations or divergence in the estimates further validates the stability of the overall control framework, reinforcing its applicability for real-time implementation.

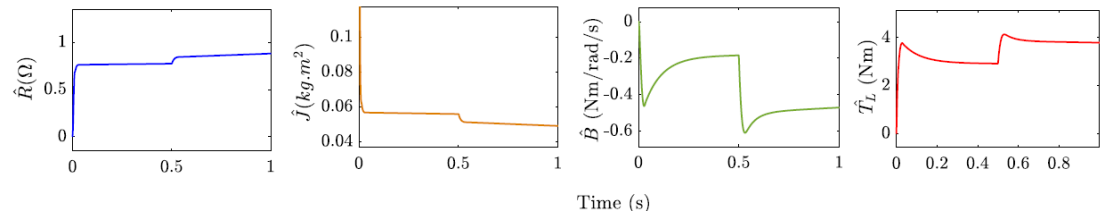


Figure 3.12: Estimated parameters and the disturbance torque for case 2.

Figure 3.13 illustrates the transient and steady-state performance of the proposed adaptive backstepping controller for PMSM drive systems under two distinct operational scenarios (Case 1 and Case 2). The adaptation laws facilitate the asymptotic convergence of parameter estimates, as evidenced by the stabilization of all trajectories within a finite time following the initial adaptation phase. Notably, the tracking error dynamics remain uniformly bounded throughout the process, demonstrating effective compensation for both parametric uncertainties and external load disturbances. The observed convergence times align well with the theoretical stability criteria, highlighting the controller's ability to minimize overshoot while rapidly estimating system parameters. These results confirm the efficacy of the adaptive mechanism in ensuring global system stability and precise reference tracking, thereby validating the theoretical developments presented in earlier sections.

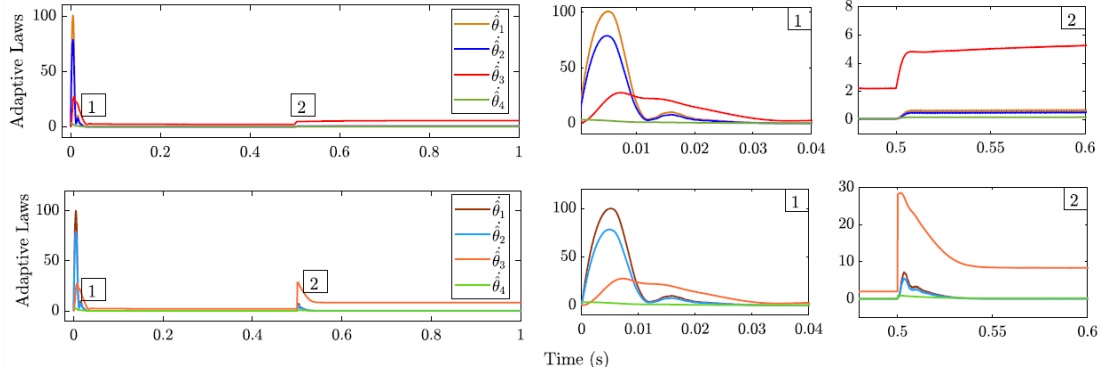


Figure 3.13: The adaptive laws for case 1 and case 2.

3.3. DNN based Adaptive Backstepping Controller Design

In this subsection, the adaptive backstepping controller is extended by incorporating a deep neural network-based approximation term to compensate for the unknown nonlinearities of the PMSM. Since the experimental motor used in this study is a surface-mounted PMSM (SPMSM), its magnetic characteristics satisfy $L_d = L_q = L$ which simplifies the electrical dynamics and eliminates saliency-dependent coupling terms. To ensure full consistency between the theoretical development and the experimental implementation, the PMSM model is therefore reintroduced here using the surface-mounted formulation, which differs slightly from the general model employed in the earlier adaptive backstepping derivation. This refinement yields a more accurate representation of the physical system and allows the neural

approximators to learn the true lumped uncertainties present in the SPMSM dynamics. The revised mechanical and electrical equations form the foundation for the DNN-augmented adaptive backstepping design, in which the inner-loop stabilization structure is preserved while the neural network provides real-time function approximation to enhance robustness against parameter uncertainties, unmodeled disturbances, and load variations.

3.3.1. System Dynamics and Error Definitions

We start the controller design by redefining the PMSM dynamic equations as follows, where $L_d = L_q = L$:

$$L \frac{di_d}{dt} = -Ri_d + L\omega i_q + v_d \quad (3.55)$$

$$L \frac{di_q}{dt} = -Ri_q - \omega Li_d + \omega \phi_m + v_q \quad (3.56)$$

$$J \frac{d\omega_r}{dt} = \frac{3}{2} P \phi_m i_q - B\omega_r - T_L \quad (3.57)$$

where $i_d(t)$ and $i_q(t) \in \mathbb{R}$ denote the direct and quadrature axis stator currents, while $\omega(t)$ and $\omega_r(t) \in \mathbb{R}$ represent the electrical and mechanical rotor speeds, respectively. The inputs $v_d(t)$ and $v_q(t) \in \mathbb{R}$ are the corresponding dq -axis stator voltages. The motor parameters include the stator resistance $R \in \mathbb{R}^+$, stator inductance $L \in \mathbb{R}^+$, rotor inertia $J \in \mathbb{R}^+$, viscous friction coefficient $B \in \mathbb{R}^+$, number of pole pairs $P \in \mathbb{R}^+$, load torque $T_L \in \mathbb{R}$, and permanent-magnet flux linkage $\phi_m \in \mathbb{R}^+$. Throughout the analysis, the notation $(\cdot)$ and $\frac{d}{dt}$ are used interchangeably to denote time differentiation. Before introducing the tracking errors and developing the adaptive backstepping controller, the following assumption regarding parameter knowledge is adopted:

Assumption 3.1. The parameters R, B, J , and T_L are unknown but bounded, whereas the stator inductance L is considered known up to small tolerances. Their values are decomposed as follows:

$$\begin{aligned}
J &= J_N + \Delta_J \\
B &= B_N + \Delta_B \\
L &= L_N + \Delta_L \\
T_L(t) &= T_{L_N} + \Delta_{T_L}(t) \\
R &= R_N + \Delta_R
\end{aligned} \tag{3.58}$$

where R_N, B_N, J_N, T_{L_N} and $L_N \in \mathbb{R}^+$ are known nominal values. The terms $\Delta_R, \Delta_B, \Delta_J \in \mathbb{R}, \Delta_{T_L}(t)$, and $\Delta_L \in \mathbb{R}$ denote bounded, slowly time-varying uncertainties and disturbances. Since Δ_L represents only small deviations in stator inductance, its effect is negligible and may be ignored in the analysis.

The primary objective of the proposed control architecture is to ensure that the PMSM accurately tracks a desired speed trajectory even in the presence of parametric uncertainties in both the electrical and mechanical subsystems, as well as unknown load disturbances. The mechanical speed $\omega_r(t)$ and the dq -axis currents $i_d(t)$ and $i_q(t)$ are assumed to be available for feedback. In addition to achieving precise tracking, guaranteeing closed-loop stability is of central importance. The speed-tracking error is defined as follows:

$$e(t) \triangleq \omega_r(t) - \omega_r^*(t), \tag{3.59}$$

where $\omega_r^*(t) \in \mathbb{R}$ denotes the reference mechanical speed.

3.3.2. Design of the Control Loops

Differentiating the error defined in equation (3.59), pre-multiplying by J , and substituting $J\dot{\omega}$ using the system dynamics in equation (3.57), yields:

$$J\dot{e} = \frac{3}{2}P\phi_m i_q - B\omega_r - T_L - J\dot{\omega}_r^* \tag{3.60}$$

The terms in equation (3.60) that contain unknown parameters may be regrouped as follows:

$$Y_1 \theta = [\omega_r \quad 1 \quad \dot{\omega}_r^*] \begin{bmatrix} B \\ T_L \\ J \end{bmatrix}, \quad (3.61)$$

where $Y_1(t) \in \mathbb{R}^{1 \times 3}$ is the regressor vector and $\theta \in \mathbb{R}^3$ is the parameter vector. Substituting equation (3.61) into equation (3.60) and adding/subtracting the term $\frac{3P\phi_m}{2} i_q^*$ leads to the following:

$$J\dot{e} = \frac{3}{2}P\phi_m e_q - Y_1 \theta + \frac{3P\phi_m}{2} i_q^*, \quad (3.62)$$

where $e_q(t) \triangleq i_q - i_q^*(t)$ is the q -axis current tracking error and $i_q^*(t)$ is the virtual control input associated with the backstepping procedure.

A key step in backstepping control synthesis is the appropriate selection of this virtual control input so that the resulting error dynamics is rendered stable. In accordance with this requirement, i_q^* is designed as below:

$$i_q^* = \frac{2}{3P\phi_m} (-k_1 e + Y_1 \hat{\theta}), \quad (3.63)$$

where $k_1 > 0$ is a stabilizing gain and $\hat{\theta}(t) \in \mathbb{R}^3$ represents the adaptive estimates of the unknown parameters. The update law for $\hat{\theta}$ is chosen as:

$$\dot{\hat{\theta}} = -\Gamma_1 Y_1^\top e, \quad (3.64)$$

with $\Gamma_1 \in \mathbb{R}^{3 \times 3}$ being a positive definite, diagonal adaptation gain matrix. Substituting the virtual control law defined in equation (3.63) into equation (3.62) yields the closed-loop speed error dynamics as follows:

$$J\dot{e} = -k_1 e - Y_1 \tilde{\theta} + \frac{3}{2}P\phi_m e_q, \quad (3.65)$$

where $\tilde{\theta}(t) \triangleq \theta - \hat{\theta}$ denotes the parameter estimation error. In the remainder of the analysis, the notation $(\cdot) \triangleq (\cdot) - (\cdot)$ is used to denote estimation errors.

The reference d-axis current is selected as $i_d^* = 0$. Thus, the corresponding tracking error reduces to $e_d(t) \triangleq i_d(t)$. Using the dq -axis dynamics in equation (3.55), the error system becomes:

$$L\dot{e}_d = -Ri_d + \omega Li_q + v_d. \quad (3.66)$$

The d -axis control input is selected as:

$$v_d = -k_2 e_d - L\omega i_q + \hat{R}i_d, \quad (3.67)$$

where $k_2 > 0$ is a design parameter and $\hat{R}(t)$ is the adaptive estimate of the stator resistance. The associated adaptation law is defined by:

$$\dot{\hat{R}} = -\gamma e_d i_d, \quad (3.68)$$

with $\gamma > 0$ denoting a scalar adaptation gain. Substituting equation (3.67) into equation (3.66) yields the closed-loop error dynamics:

$$L\dot{e}_d = -k_2 e_d - \tilde{R}i_d \quad (3.69)$$

where $\tilde{R}(t) \triangleq R - \hat{R}$ is the resistance estimation error. To design the q -axis control input v_q , the time derivative of the error e_q is computed, multiplied by L , and expanded using the system dynamics in equation (3.56) and the derivative of the virtual control i_q^* in equation (3.63). This results in the following form:

$$\begin{aligned} L\dot{e}_q = & -Ri_q - \omega Li_d + \omega\phi_m + v_q + \frac{2Lk_1}{3P\phi_m}\dot{e} \\ & - \frac{2L}{3P\phi_m}(\dot{Y}_1\hat{\theta} + Y_1\dot{\hat{\theta}}) \end{aligned} \quad (3.70)$$

Substituting the expression $\dot{e}$ defined in equation (3.60) into equation (3.70) and grouping similar terms yields:

$$L\dot{e}_q = g(t) + f(t) + v_q \quad (3.71)$$

where $g(t)$ contains all known or measurable quantities (see Appendix-A). The term $f(t)$ collects all expressions involving unknown parameters and can be written as in the following matrix format:

$$f = \begin{bmatrix} -i_q \\ i_q k_1 L \\ -\frac{2Lk_1 \omega_r}{3P\phi_m} \\ \frac{2Lk_1}{3P\phi_m} \end{bmatrix}^\top \begin{bmatrix} R \\ 1 \\ \frac{1}{J} \\ \frac{B}{J} \\ \frac{I_r}{J} \end{bmatrix}. \quad (3.72)$$

As is evident from equation (3.72), accurately estimating the nonlinear function f requires simultaneous online estimation of multiple parameters and their interactions with system states. This is computationally demanding and prone to modeling inaccuracies, especially under varying operating conditions. To bypass these challenges, the proposed approach employs a deep neural network (DNN) to approximate the unknown nonlinear dynamics directly, rather than estimating individual parameters.

According to the universal approximation theorem, any continuous function defined over a compact domain can be approximated to arbitrary precision by a sufficiently expressive neural network [38]. Let

$$x(t) \triangleq \begin{bmatrix} i_d(t) \\ i_q(t) \\ \omega_r(t) \end{bmatrix} \in \mathbb{R}^3 \quad (3.73)$$

and assume that $x(t)$ evolves within a compact set $\Omega \subset \mathbb{R}^3$. Define $\mathbb{S}^3(\Omega)$ as the space of continuous functions on Ω . For $f(x) \in \mathbb{S}^3(\Omega)$, there exists an ideal DNN representation of the form

$$f(x) = W^\top \sigma(\Phi(x)) + \varepsilon(x), \quad (3.74)$$

where $W \in \mathbb{R}^{n_f}$ is the ideal (unknown) output-layer weight vector, $\sigma(\cdot): \mathbb{R}^p \rightarrow \mathbb{R}^{n_f}$ denotes the ideal (unknown) activation functions, $\Phi: \mathbb{R}^3 \rightarrow \mathbb{R}^p$ represents the ideal (unknown) feature map generated by the inner DNN layers, $\varepsilon(x)$ is the bounded functional reconstruction error.

The ideal deep network can be expressed using nested compositions as

$$\Phi(x(t)) = (W_k^\top \phi_k \circ W_{k-1}^\top \phi_{k-1} \circ \cdots \circ W_1^\top \phi_1)(x(t)), \quad (3.75)$$

where $k \in \mathbb{Z}$ denotes the number of hidden layers, W_i denotes the inner-layer weights, and $\phi_i(\cdot)$ the activation functions. The symbol $(\circ)$ denotes the composition of the functions, W_i and ϕ_i denote the corresponding inner layer weights and activation functions of the DNN, respectively. Before proceeding, following remarks are given and an assumption for the DNN Approximation is made:

Remark 3.1: The compactness of the set Ω guarantees that the function $f(x)$ is continuous and bounded over its domain. Consequently, $f(x)$ can be approximated by a DNN with bounded activation functions and bounded weights. This property also ensures that the associated approximation error $\varepsilon(x)$ remains finite.

Remark 3.2: The representation in equation (3.74) is based on an ideal deep neural network, which serves as a theoretical construct for approximating $f(x)$. In practice, the architecture, activation functions, and weight parameters of the implemented DNN are tuned such that the residual approximation error $\varepsilon(x)$ is minimized to the extent allowed by computational resources and training data.

Assumption 3.2: For the ideal representation in equation (3.74), there exist positive constants W , $\bar{\sigma}$, and $\bar{\varepsilon}$ such that the unknown ideal weights satisfy $\|W\| \leq \bar{W}$, the ideal activation outputs satisfy $\sup_{x \in \Omega} \|\sigma(\cdot)\| \leq \bar{\sigma}$, and the ideal functional reconstruction error satisfies $\sup_{x \in \Omega} |\varepsilon(\cdot)| \leq \bar{\varepsilon}$. These bounds ensure that the ideal DNN representation is well-posed and that the approximation error is uniformly constrained.

The q -axis control input is designed as:

$$v_q = -k_3 e_q - g - \hat{f} - \frac{3P\phi_m}{2} e - k_s \text{sgn}(e_q), \quad (3.76)$$

where $k_3 > 0$ and $k_s > 0$ are stabilizing control gains, and $\text{sgn}(\cdot)$ denotes the standard signum function. The term $\hat{f}(t)$ represents the DNN-based approximation of the unknown nonlinear function $f(t)$, constructed as

$$\hat{f}(x(t)) = \hat{W}^\top \hat{\sigma}(\hat{\Phi}(x)), \quad (3.77)$$

where $\hat{W}(t) \in \mathbb{R}^{nf}$ is the estimate of the ideal output-layer weights, $\hat{\sigma}: \mathbb{R}^p \rightarrow \mathbb{R}^{nf}$ is the estimated vector of activation functions, $\hat{\Phi}: \mathbb{R}^3 \rightarrow \mathbb{R}^p$ is the estimated feature map generated by the inner layers of the DNN. The adaptation law for the output-layer weights is selected as:

$$\dot{\hat{W}} = e_q \Gamma_2 \hat{\sigma}(\hat{\Phi}(x)), \quad (3.78)$$

where $\Gamma_2 \in \mathbb{R}^{nf \times nf}$ is a diagonal positive definite adaptation gain matrix. The notation $\hat{\sigma}$ denotes the estimated activation outputs used for online learning.

Prior to real-time operation, the inner-layer weights $\hat{\Phi}(t_0)$ and the initial output weights $\hat{W}(t_0)$ are assumed to be obtained through offline training using a dataset of sufficiently rich excitation. During the online control process, new data are continuously collected. Once an adequate amount of data has been accumulated, the inner-layer weights are retrained offline and updated to yield a refined feature map $\hat{\Phi}(t)$, while the output-layer weights $\hat{W}(t)$ continue to adapt online according to equation (3.78). This hybrid-learning framework combines fast online adaptation with periodic offline refinement.

Substituting the control input law defined in equation (3.76) into the error dynamics in equation (3.71), then adding and subtracting $\pm W^\top \hat{\sigma}(\hat{\Phi}(x))$ and regrouping the related terms yields:

$$\begin{aligned} L\dot{e}_q = & -k_3 e_q - k_s \text{sgn}(e_q) - \frac{3P\phi_m}{2} e + W^\top \tilde{\sigma}(\tilde{\Phi}(x)) \\ & + \tilde{W}^\top \hat{\sigma}(\hat{\Phi}(x)) - \varepsilon(x), \end{aligned} \quad (3.79)$$

where $\tilde{W}(t) \in \mathbb{R}^L$ is the output-layer weight estimation error defined as $\tilde{W} \triangleq W - \hat{W}$ and $\tilde{\sigma}(\tilde{\Phi}(x)) \triangleq \sigma(\Phi(x)) - \hat{\sigma}(\hat{\Phi}(x))$ represents the DNN approximation error due to discrepancies in the feature map and activation functions. The overall control structure, including adaptive laws, backstepping interconnections, and DNN compensation loop, is illustrated schematically in **Figure 3.14**.

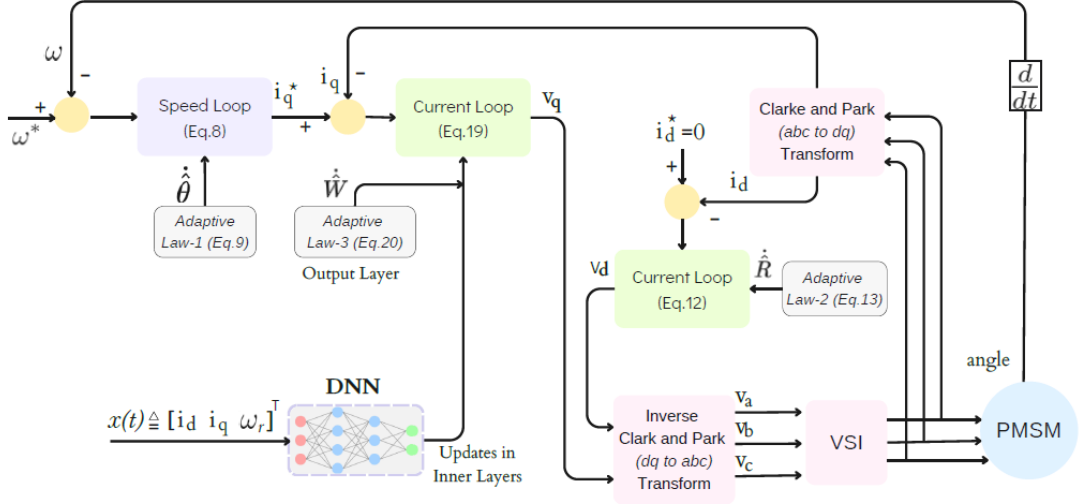


Figure 3.14: Block diagram of the proposed DNN based controller.

3.3.3. Stability Analysis

This sub-subsection establishes the stability properties of the closed-loop system under the proposed DNN-based adaptive backstepping control strategy. A Lyapunov-based analysis is employed to show that all closed-loop signals remain bounded and that the tracking errors converge asymptotically to zero.

Theorem 3.1: Consider the PMSM control system under the virtual control input in equation (3.63), the actual control inputs in equation (3.67) and equation (3.76), and the adaptive update laws in equations (3.64), (3.68), and (3.78). For any control gains $k_1, k_2, k_3 > 0$ and for k_s selected such that

$$k_s > \bar{W} \left(\bar{\sigma} + \sup_{x(t) \in \Omega} \|\hat{\sigma}\| \right) + \bar{\varepsilon}, \quad (3.80)$$

all signals in the closed-loop system remain bounded. Moreover, the tracking errors satisfy

$$\lim_{t \rightarrow \infty} e(t) = 0, \quad \lim_{t \rightarrow \infty} e_d(t) = 0, \quad \lim_{t \rightarrow \infty} e_q(t) = 0, \quad (3.81)$$

which ensures global asymptotic stability of the closed-loop system.

To prove Theorem 3.1, consider the Lyapunov candidate function as follows:

$$V \triangleq \frac{1}{2}Je^2 + \frac{1}{2}Le_d^2 + \frac{1}{2}Le_q^2 + \frac{1}{2}\tilde{\theta}^\top \Gamma_1^{-1}\tilde{\theta} + \frac{1}{2}\tilde{W}^\top \Gamma_2^{-1}\tilde{W} + \frac{1}{2}\gamma^{-1}\tilde{R}^2. \quad (3.82)$$

Differentiating equation (3.82) yields

$$\dot{V} = J\dot{e}e + L\dot{e}_de_d + L\dot{e}_qe_q + \tilde{\theta}^\top \Gamma_1^{-1}\dot{\tilde{\theta}} + \tilde{W}^\top \Gamma_2^{-1}\dot{\tilde{W}} + \gamma^{-1}\tilde{R}\dot{\tilde{R}} \quad (3.83)$$

Substituting the error dynamics in equations (3.65), (3.69) and (3.79), and using the properties of $\dot{\tilde{\theta}} = -\dot{\hat{\theta}}$, $\dot{\tilde{W}} = -\dot{\hat{W}}$, $\dot{\tilde{R}} = -\dot{\hat{R}}$, gives:

$$\begin{aligned} \dot{V} = & \left(-k_1e - Y_1\tilde{\theta} + \frac{3P\phi_m}{2}e_q \right) e \\ & + \left(-k_3e_q - \frac{3P\phi_m}{2}e - k_s \operatorname{sgn}(e_q) + W^\top \tilde{\sigma} + \tilde{W}^\top \hat{\sigma} - \varepsilon \right) e_q \\ & + (-k_2e_d - \tilde{R}i_d)e_d + \tilde{\theta}^\top Y_1^\top e - \tilde{W}^\top e_q \hat{\sigma} + \tilde{R}e_d i_d \end{aligned} \quad (3.84)$$

After straightforward simplification and cancellation of cross terms in equation (3.84), the derivative reduces to

$$\dot{V} = -k_1e^2 - k_2e_d^2 - k_3e_q^2 - k_s|e_q| + e_q W^\top \bar{\sigma}(\tilde{\Phi}(x)) - \varepsilon(x)e_q. \quad (3.85)$$

By using the second assumption, the uncertain terms satisfy

$$|e_q W^\top \bar{\sigma} - \varepsilon e_q| \leq |e_q| \left(\bar{W} \bar{\sigma} + \bar{W} \sup_{x \in \Omega} \|\hat{\sigma}\| + \bar{\varepsilon} \right). \quad (3.86)$$

Thus, the condition in equation (3.80) guarantees that the negative term $-k_s|e_q|$ dominates the uncertainty terms, yielding

$$\dot{V} = -k_1e^2 - k_2e_d^2 - k_3e_q^2 \quad (3.87)$$

Since $\dot{V} \leq 0$, the Lyapunov function $V(t)$ is bounded, implying

$$e(t), e_d(t), e_q(t), \bar{\theta}(t), \bar{W}(t), \tilde{R}(t) \in \mathcal{L}_\infty \quad (3.88)$$

All associated derivatives are also bounded because they depend on bounded signals. Integrating terms in equation (3.88) from 0 to ∞ yields

$$\int_0^\infty (e^2(\tau) + e_d^2(\tau) + e_q^2(\tau))d\tau \leq \frac{V(0)}{\min\{k_1, k_2, k_3\}} \quad (3.89)$$

showing that the tracking errors are also in $\mathcal{L}_2$.

Since $e(t) \in \mathcal{L}_\infty \cap \mathcal{L}_2$, $\dot{e}(t) \in \mathcal{L}_\infty$, Barbalat's Lemma ensures $e(t) \rightarrow 0$ as $t \rightarrow \infty$. Applying the same argument to $e_d(t)$ and $e_q(t)$ yields

$$e_d(t) \rightarrow 0, e_q(t) \rightarrow 0 \text{ as } t \rightarrow \infty. \quad (3.90)$$

Thus, the proposed control laws guarantee asymptotic stability of the closed-loop system, completing the proof of Theorem 3.1.

3.3.4. Numerical Studies

The proposed control framework integrates adaptive backstepping with a deep neural network (DNN) through a hybrid learning architecture that combines offline pretraining with online weight adaptation.

In the offline phase, the inner-layer parameters of the DNN are pretrained using a dataset generated under conventional PI-based field-oriented control (FOC). This ensures that the neural network captures the nominal nonlinear behavior of the PMSM prior to real-time deployment. During closed-loop operation, only the output-layer weights of the DNN are updated online according to Lyapunov-based adaptive laws, while the inner layers remain fixed. This two-stage learning strategy exploits the universal approximation property of deep networks while preserving the stability guarantees inherent to adaptive backstepping control. As established in the previous section, the hybrid adaptive mechanism ensures uniform boundedness of all closed-loop signals and asymptotic convergence of the tracking errors.

To assess the performance of the proposed DNN-based adaptive backstepping controller, extensive numerical simulations were carried out on a PMSM model incorporating parametric uncertainties and time-varying load disturbances. The evaluation methodology considers both approximation accuracy and control performance. The DNN estimation quality is quantified using the Root Mean Square Error (RMSE), while the quality of speed tracking is assessed using widely adopted

time-domain performance indices, including the Integral of Squared Error (ISE), Integral of Absolute Error (IAE), Integral of Time-weighted Squared Error (ITSE), and Integral of Time-weighted Absolute Error (ITAE). These indices provide a comprehensive characterization of transient response, steady-state behavior, and robustness to disturbances.

Table 3.4: Parameters of Teknik M-2310P-LN-04K PMSM.

Parameter	Symbol	Value	Unit
Stator resistance	R	0.36	Ω
Stator inductance	L	0.2	mH
PM flux linkage	ϕ_m	0.0064	Wb
Moment of inertia	J	7.06×10^{-6}	$\text{kg} \cdot \text{m}^2$
Viscous friction	B	2.64×10^{-6}	Nms/rad
Pole pairs	P	4	---
DC-link voltage	V_{dc}	24	V
Rated torque	T_{rated}	0.2724	Nm
Rated current	I_{rated}	7.10	A
Rated speed	N_{rated}	4107	RPM
Sampling time	T_s	0.1	ms

The PMSM parameters used in the simulations correspond to a commercially available servo-grade motor frequently employed in industrial applications. These parameters are summarized in **Table 3.4**. In addition to the numerical studies, hardware validation results are also presented in the following subsection.

3.3.5. Simulation Method

Matlab/Simulink simulations were performed over a 40s time horizon to evaluate the behavior of the proposed hybrid DNN-based adaptive backstepping controller under diverse operating conditions. The simulation study focuses on three primary objectives:

1. **Speed tracking performance** under changes in the reference trajectory;
2. **Robustness** against both step-like and time-varying load disturbances;

3. Stability and convergence of the adaptive laws, including online weight updates and periodic DNN retraining events.

Two complementary sets of performance metrics were used to comprehensively assess the controller's behavior. Firstly, the RMSE of the DNN function approximation was evaluated before and after each retraining event to quantify the network's ability to learn and refine the nonlinear mapping $f(x)$. Secondly, the overall control performance was assessed via standard time-domain indices (ISE, IAE, ITSE, and ITAE) which provide detailed insight into transient response quality, steady-state behavior, and sensitivity to disturbances. Together, these metrics characterize both the estimation capability of the DNN and the robustness and stability of the closed-loop adaptive system.

3.3.5.1. Dataset Generation Process

A representative dataset was generated using a 100s closed-loop simulation driven by a baseline cascade PI controller operating independently of the DNN. This PI-controlled system ensured stable regulation of the PMSM speed and currents across diverse operating regions and served as the source of training samples for offline pretraining of the DNN's inner layers. As seen in **Figure 3.15**, the excitation trajectory was designed to stimulate a broad range of dynamic behaviors and operating points: **Phase 1 (0-20s)**: A linear ramp reference from 0 to 20 rad/s to capture transient acceleration dynamics. **Phase 2 (20-40s)**: A constant reference of 20 rad/s to represent steady-state operation under nominal load. **Phase 3 (40-100s)**: A sinusoidal reference $\omega^*(t) = 20 + 10\sin(2\pi ft)$ with $f = 1/60$ Hz, to introduce periodic variations and ensure sufficient richness for training the DNN over nonlinear and time-varying regimes. This dataset supports effective offline learning of the DNN feature representation and ensures robustness when transitioning to online adaptation during real-time control.

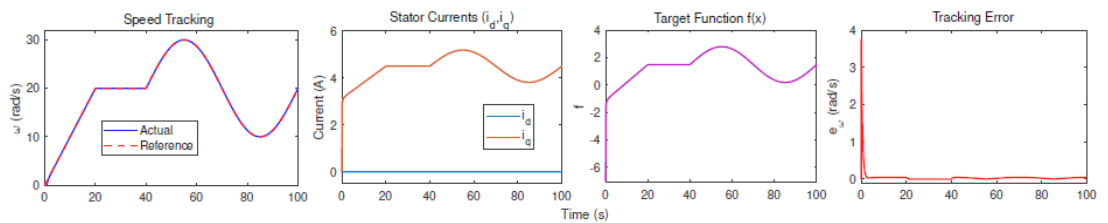


Figure 3.15: PI controller performance for dataset generation.

A constant load torque $T_L = 0.12\text{Nm}$ (approximately 44% of rated torque) was applied during data collection. This static load ensures that the DNN captures the underlying system dynamics without overfitting to specific disturbance patterns. Artificial damping ($B_{\text{artificial}} = 1000B$) was introduced to ensure stability during data collection, yielding a numerically well-conditioned dataset that still reflects nonlinear characteristics. The resulting dataset contains 10^5 state-target pairs $(\mathbf{x}_i, f_i)$ sampled at 1 ms intervals, where $\mathbf{x}_i = [i_{d,i}, i_{q,i}, \omega_i]^T$ and f_i is computed from equation 3.72. The PI controller maintained a mean speed error below 1rad/s, confirming dataset reliability.

3.3.5.2. Offline Pretraining Process

The dataset was partitioned into training (70%), validation (15%), and testing (15%) subsets. MATLAB's `fitnet` function with the Levenberg-Marquardt `trainlm` algorithm was used, offering fast convergence on moderate-sized datasets. Two architectures, DNN-1 ([6-3]) and DNN-2 ([6-4-3]), were trained to investigate the effect of network depth on approximation and generalization. Key hyperparameters are summarized in **Table 3.5**.

Table 3.5: DNN specifications.

Parameter	Specification
Optimization algorithm	Levenberg-Marquardt <code>trainlm</code>
Performance metric	Mean Squared Error (MSE) in normalized space
Maximum epochs	1000
Early stopping	Validation patience of 100 epochs to prevent overfitting
Activation function	Hyperbolic tangent ($\tanh$) for all hidden layers

After training, the inner-layer weights were frozen to serve as a pretrained feature extractor mapping states $\mathbf{x}(t)$ to feature vector $\mathbf{z}(t) \in \mathbb{R}^{n_f}$, with $n_f = 4$. The outer layer, consisting of adaptive weights $\mathbf{W} \in \mathbb{R}^{n_f}$ and bias b , was initialized to zero and updated online via the adaptive law in equation (3.78). This separation ensures efficient reuse of pretrained features while maintaining adaptive flexibility. Controller gains satisfying Lyapunov conditions are listed in **Table 3.6**.

Table 3.6: Controller design gains.

Parameter	Description	Value
k_1	Speed loop gain	0.08
k_2	d -axis current gain	0.3
k_3	q -axis current gain	0.15
k_s	Robust compensation gain	0.1
Γ_1	Mechanical adaptation gain	$\text{diag}([10^{-4}, 10^{-4}, 10^{-4}])$
γ	Resistance adaptation gain	0.4
Γ_2	DNN outer-layer gain	0.075

3.3.5.3. Retraining and Online Adaptive Control Process

The input to the DNN consists of three measured states:

$$x(t) = [i_d(t), i_q(t), \omega(t)]^T \quad (3.91)$$

At each control cycle, the pretrained inner layers extract feature representations based on the measured input. These features can be expressed generically as

$$z(t) = \sigma([W_{hid} \dots \sigma(W_{in}\tilde{x}(t) + b_{in}) \dots + b_{hid}])^T \quad (3.92)$$

where W_{hid} , W_{in} , b_{in} and b_{hid} denote the fixed inner-layer parameters, and $\sigma(\cdot)$ represents the activation functions. The adaptive outer layer forms the real-time estimate of the unknown nonlinear function as

$$\hat{f}(x(t)) = W^T(t)z(t) + b(t) \quad (3.93)$$

This estimate is incorporated into the q -axis control law, corresponding to equation (3.77), to compensate for modeling uncertainties during real-time operation. The online learning mechanism includes two complementary adaptation processes:

- i. Continuous Outer-Layer Adaptation:** The output-layer weights $W(t)$ are updated at every control iteration ($\Delta t = 0.1$ ms) according to the Lyapunov-based adaptation law. This continuous update enables rapid adjustment to parameter variations, disturbances, and operating condition changes.

ii. Periodic Inner-Layer Retraining: To improve long-term approximation accuracy and prevent representation drift, the inner-layer weights are periodically retrained using data collected during online operation. Retraining events occur at $t = 10\text{s}$ and $t = 20\text{s}$

At each retraining instant, buffered data pairs $\{(x_k, f_k)\}$ are used to retrain the inner layers using a supervised learning routine (e.g., `fitnet` with 80% training and 20% validation). The newly trained weights replace the previous inner-layer parameters, while the outer-layer weights continue from their current values rather than being reinitialized. This ensures that previously acquired online knowledge is preserved. This hybrid learning structure preserves the stability benefits of fixed pretrained features while enabling gradual refinement of the internal representation as new operational data becomes available. In combination, the two adaptive layers provide fast response to dynamic uncertainties and improved function approximation accuracy over prolonged operation.

3.3.5.4. Load Disturbance and Reference Profiles

Four distinct load profiles, shown in **Figure 3.16**, were designed to evaluate disturbance rejection performance of the proposed controller.

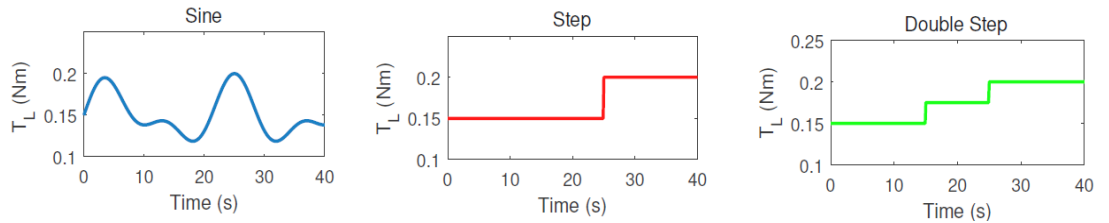


Figure 3.16: Load torque profiles for the simulation study.

The mathematical expressions of the applied torque loads are summarized in **Table 3.7**. It should be noted that none of these torque types match the constant $T_L = 0.12\text{ Nm}$ torque load used in the pre-training process of the DNN, thus ensuring evaluation under unseen disturbances.

Two reference speed trajectories were planned: Profile 1 consists a ramp between $t=0$ and 8s , a constant between $t=8$ and 16s , following a sinusoidal function $\omega^*(t) = 100 + 50\sin(2\pi ft)$, with $f = 1/(T_{end} - 16)\text{Hz}$. Profile 2 has step transitions defined as:

$$\omega^*(t) = \begin{cases} 60, & 0 \leq t < 15 \\ 120, & 15 < t \leq 25 \\ 200, & 25 < t \leq 40, \end{cases} \quad (3.94)$$

which introduces 100% and 67% speed jumps to test adaptability under abrupt transitions.

Table 3.7: Mathematical expressions of the applied load torque.

Load Torque Profile	Mathematical Expression $T_L(t)$ (Nm)
Sine	$0.15 + 0.03 \sin(2\pi \cdot 0.05t) + 0.02 \sin(2\pi \cdot 0.09t)$
Step	$0.15 + 0.05 \cdot 1_{(t \geq 15)}$
Double Step	$0.15 + 0.05 \cdot 1_{(t \geq 15)} + 0.05 \cdot 1_{(t \geq 25)}$

3.3.5.5. Simulation Results

To rigorously assess the effectiveness of the proposed controller, two deep neural network architectures with different numbers of hidden layers were considered:

- DNN-1 with layer dimensions [6-3]
- DNN-2 with dimensions [6-4-3].

The notation $[n_1-n_2-n_3]$ denotes the number of neurons in the pretrained hidden layers of the multilayer perceptron (MLP) feature extractor (excluding the input layer). Thus, DNN-1 [6-3] uses two hidden layers with 6 and 3 neurons, while DNN-2 [6-4-3] uses three hidden layers with 6, 4, and 3 neurons. These architectures were tested under a comprehensive set of operating conditions combining three distinct load torque profiles and two reference speed trajectories, resulting in twelve simulation scenarios. This experimental design enables a systematic comparison of controller robustness with respect to both exogenous disturbances and variations in commanded motion profiles.

Representative time-domain responses for selected cases are provided in **Figures 3.17 - 3.32** while the full set of quantitative performance comparisons is reported in **Table 3.8** and **Table 3.9** to establish a unified performance metric beyond qualitative inspection of tracking curves and neural estimation behavior, four widely used integral performance indices were computed for each scenario:

$$\begin{aligned}
ISE &= \int_0^T e^2(t) dt \\
IAE &= \int_0^T |e(t)| dt \\
ITSE &= \int_0^T t \cdot e^2(t) dt \\
ITAE &= \int_0^T t \cdot |e(t)| dt
\end{aligned} \tag{3.95}$$

where $e(t)$ denotes the tracking error of the evaluated signal and $T=40s$ is the total simulation horizon. The indices ISE and IAE capture the accumulated magnitude of the error over time, with ISE penalizing large instantaneous deviations more strongly due to squaring, while IAE assigns equal weight to all deviations. In contrast, ITSE and ITAE introduce explicit time-weighting factors that accentuate late-time performance. This distinction is crucial in adaptive, learning-based systems where rapid transient deviations may occur initially, but long-term convergence quality is a more meaningful indicator of controller stability and robustness. Consequently, ITSE and ITAE effectively penalize sustained steady-state errors and slow convergence, providing additional insight into asymptotic behavior beyond transient overshoot.

Overall, these performance metrics allow objective comparison across architectures and operating conditions, enabling a comprehensive evaluation of both transient and steady-state characteristics of the proposed DNN-based adaptive backstepping control strategy.

Figure 3.17 illustrates the dynamic performance of the DNN-1 ([6-3]) during a step torque load with a sinusoidal speed reference. It demonstrates that the actual motor speed closely follows the reference trajectory, achieving improved tracking accuracy after each retraining phase.

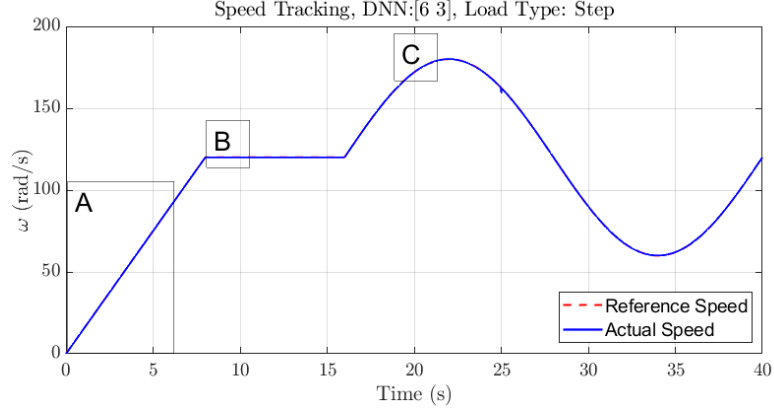


Figure 3.17: Scenario 1, the speed-tracking performance.

Figure 3.18 shows the zoomed views of the speed tracking graphics. The progressive enhancement in transient response and steady-state accuracy as the DNN updates its outer weights through iterative learning.

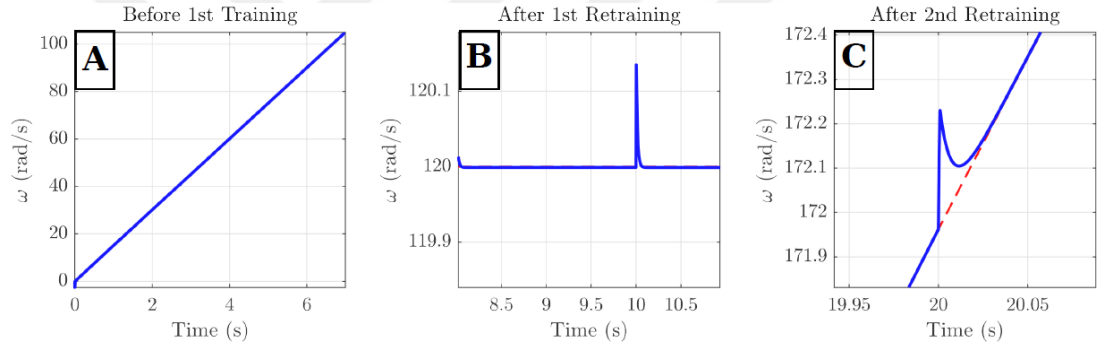


Figure 3.18: Zoomed speed views of scenario 1.

As seen in **Figure 3.19**, the adaptation process of the outer-layer weights confirms that the network parameters converge smoothly, indicating stable online learning behavior without oscillations or divergence.

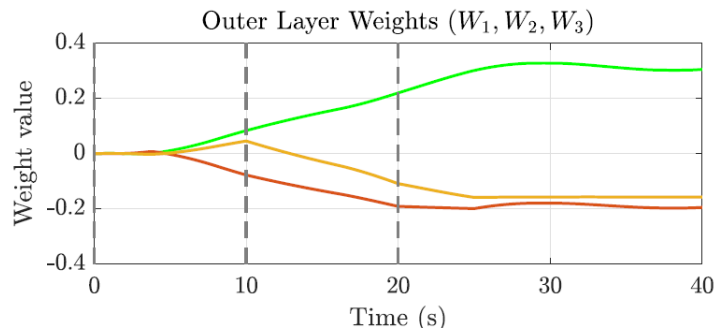


Figure 3.19: Outer layer adaptations in scenario 1.

In **Figure 3.20**, the stator current responses remain well regulated throughout the operation, validating that the control effort is distributed effectively between the torque-producing and flux-producing components. The control input v_q remains bounded and exhibits smooth transitions, confirming that the proposed control law maintains continuity and avoids chattering even during retraining periods.

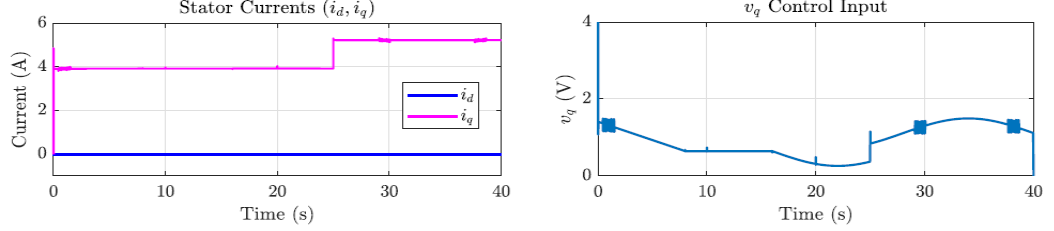


Figure 3.20: Stator currents and control input in scenario 1.

Figure 3.21 illustrates the closed-loop performance of the DNN-2 architecture under the operating conditions of Scenario 2, corresponding to a sinusoidal reference speed trajectory subjected to a step change in load torque. The speed response demonstrates accurate and smooth tracking, with noticeable performance improvements following each scheduled retraining event.

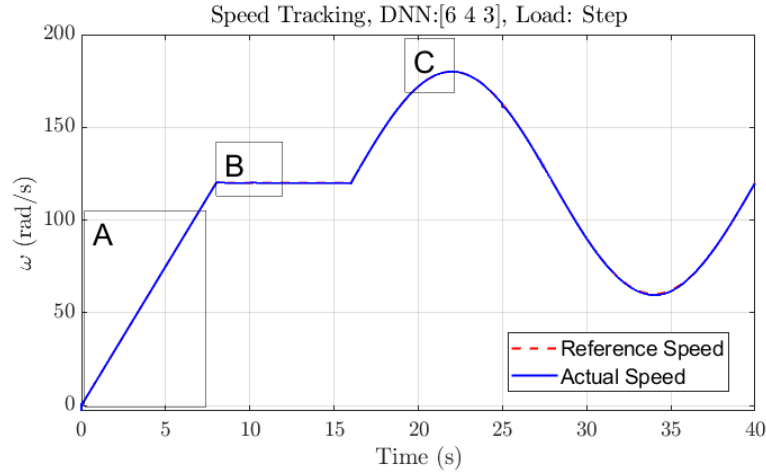


Figure 3.21: Scenario 2, the speed-tracking performance.

As seen in **Figure 3.22**, the zoomed-in views highlight reduced transient deviations and improved steady-state accuracy as the network progressively adapts to system nonlinearities and unmodeled disturbances.

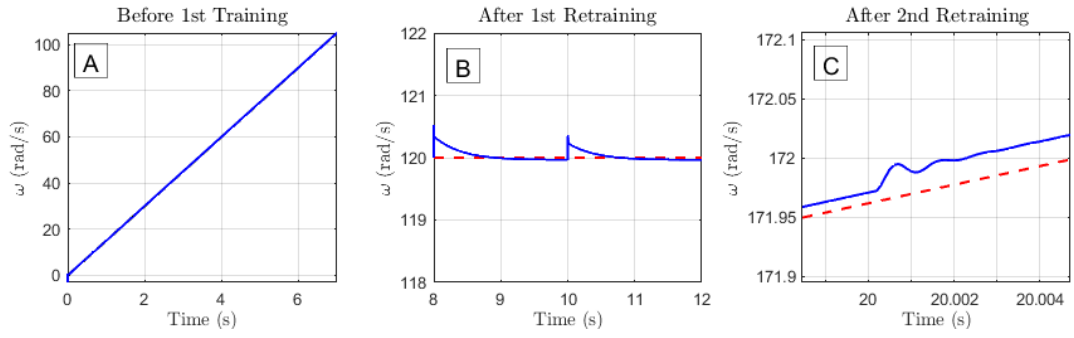


Figure 3.22: Zoomed views of the speed in Scenario 2.

The evolution of the adaptive outer-layer weights in **Figure 3.23** indicates stable learning dynamics, showing neither divergence nor oscillatory behavior, thereby verifying the effectiveness of the online weight update mechanism.

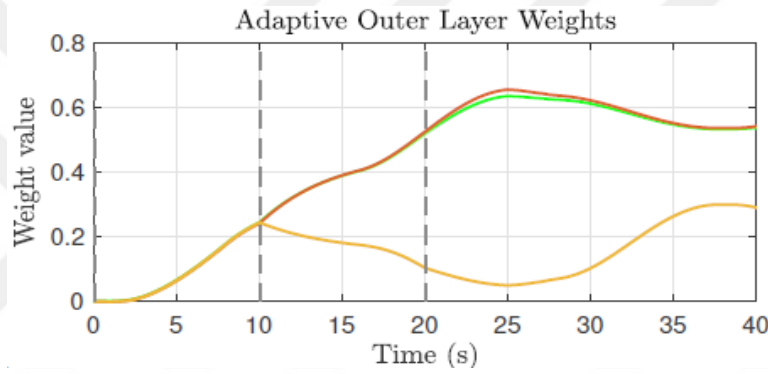


Figure 3.23: Outer layer adaptations in scenario 2.

As depicted in **Figure 3.24**, stator current waveforms remain well regulated and bounded, exhibiting smooth variations consistent with appropriate torque generation. Similarly, the control input v_q remains continuous and free from abrupt changes, indicating that the adaptation mechanism does not introduce high-frequency control activity or instability.

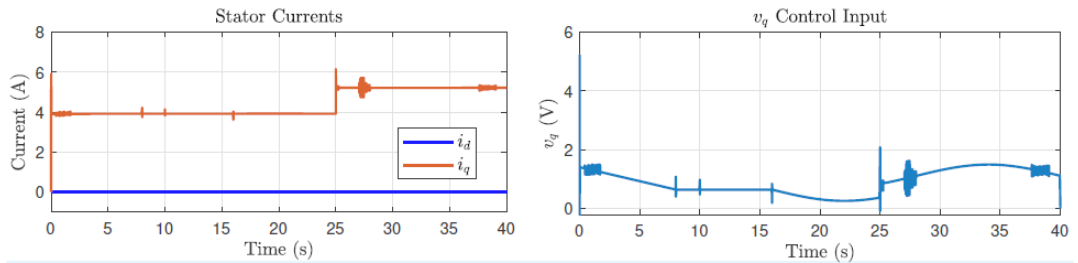


Figure 3.24: Stator currents and control input in scenario 2.

As the third simulation scenario, the DNN-1 controller is evaluated under a double-step reference speed trajectory in the presence of a continuous sinusoidal load torque disturbance.

As seen in **Figure 3.25** and **3.26**, which are the zoomed version of the speed tracking, the abrupt transitions in the commanded speed introduce significant dynamic demand on the controller. Consequently, the DNN-1 architecture exhibits a transient overshoot of approximately 1.67% and a steady-state error of 0.85% (see **Figure 3.26**), reflecting the increased control activity required to accommodate both rapid reference changes and persistent periodic disturbances.

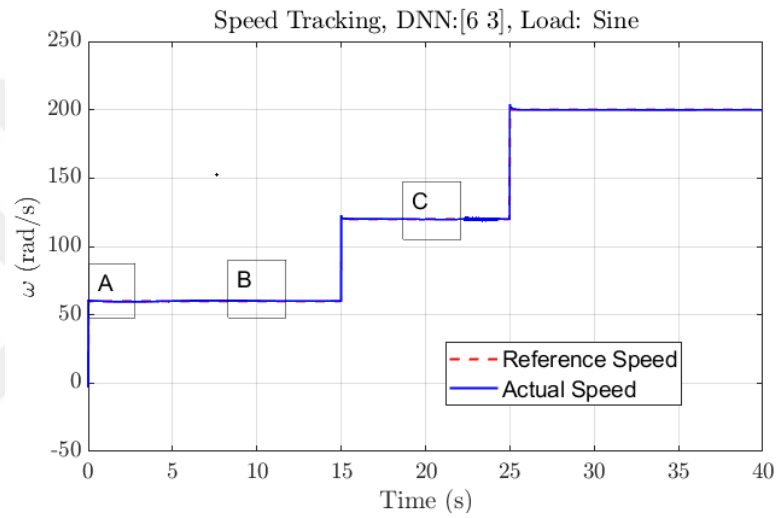


Figure 3.25: Scenario 3, the speed-tracking performance.

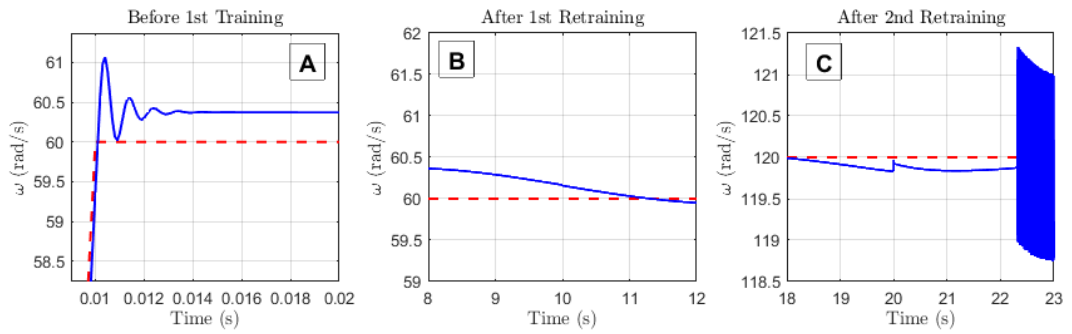


Figure 3.26: Zoomed views of the speed in scenario 3.

As illustrated in **Figure 3.27**, the adaptive outer-layer weights evolve smoothly throughout the simulation, exhibiting stable convergence characteristics without oscillatory growth or drift, which confirms that the online adaptation mechanism remains well behaved and consistently responds to changes in system dynamics and retraining intervals.

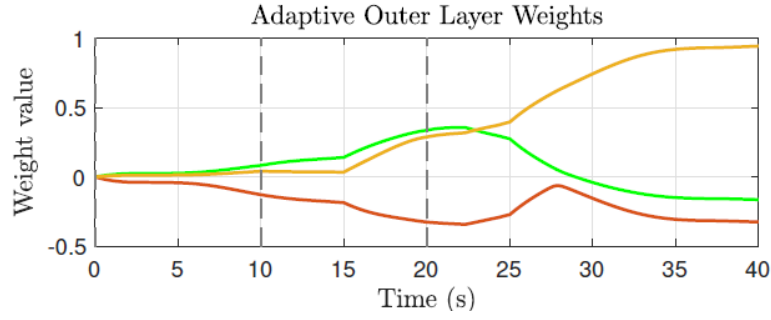


Figure 3.27: Outer layer adaptations in scenario 3.

Figure 3.28 that indicates the stator currents and control input signal. It demonstrate that d -axis current remains tightly regulated around zero while q -axis current exhibits smooth, bounded behavior throughout the operation, with brief increases occurring only during the retraining intervals and torque transients. The control input remains continuous and well behaved, with momentary peaks corresponding to retraining events and load disturbances, confirming the stability and robustness.

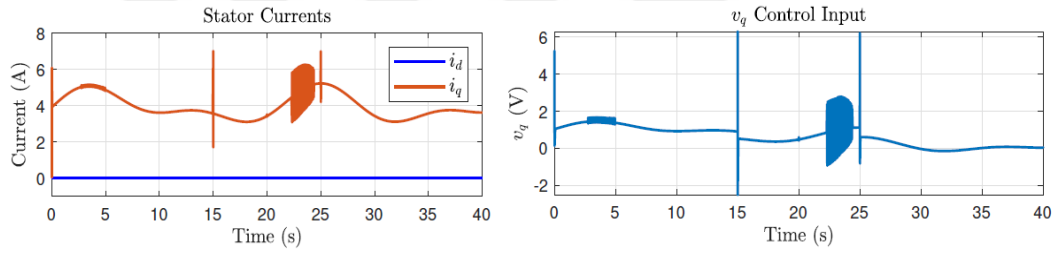


Figure 3.28: Stator currents and control input in scenario 3.

Scenario 4 consists the same operating conditions of the previous scenario, but with DNN-2 structure. **Figure 3.29** shows the speed tracking performance where is the control effort decreases noticeably (see **Figure 3.32**), while both overshoot and steady-state error exhibit slight improvement (see **Figure 3.30**).

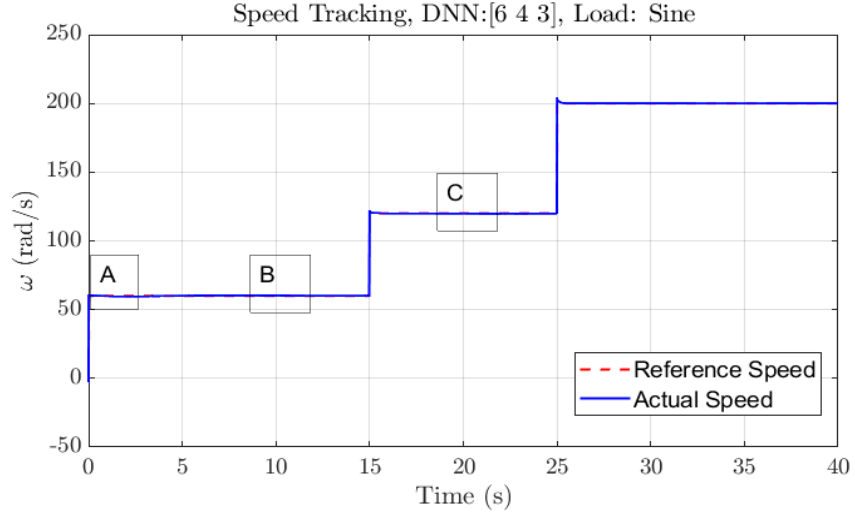


Figure 3.29: Scenario 4, the speed-tracking performance.

Figure 3.30 shows the detailed view of the certain regions of the speed-tracking trajectory, while **Figure 3.31** depicts the outer layer adaptations.

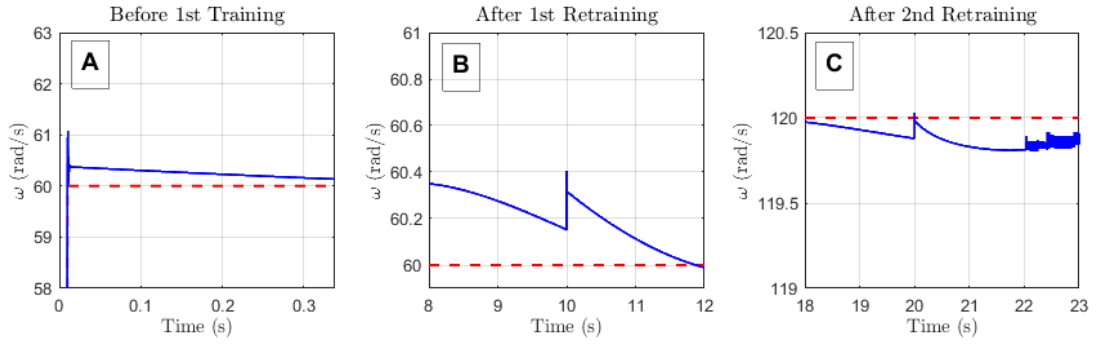


Figure 3.30: Zoomed views of the speed in Scenario 3.

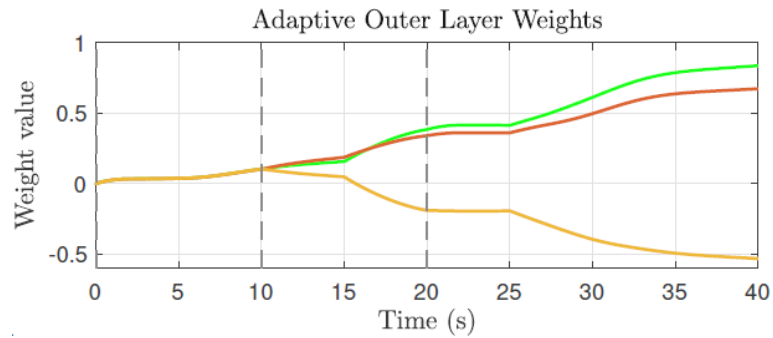


Figure 3.31: Outer layer adaptations in scenario 4.

Overall, these results demonstrates the DNN-2 architecture provides superior dynamic smoothness and improved robustness compared to the shallower DNN-1 structure, while maintaining or slightly enhancing steady-state tracking precision. These findings reinforce the value of increased network depth within the proposed adaptive learning

framework, particularly under nontrivial reference transitions and sustained external disturbances.

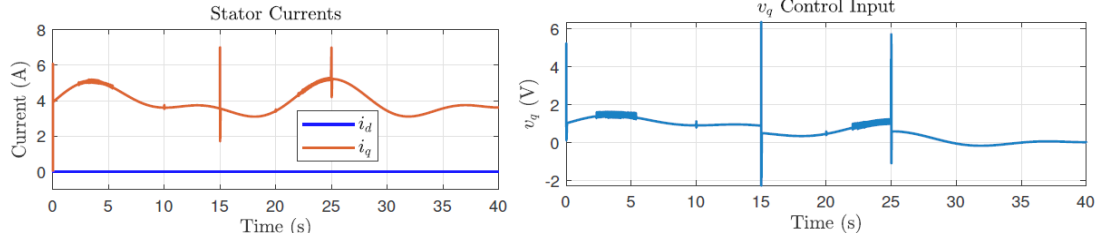


Figure 3.32: Stator currents and control input in Scenario 4.

The remaining eight simulation scenarios are consolidated in **Table 3.8** and **Table 3.9**, which present the evolution of RMSE values together with the four performance indices (ISE, IAE, ITSE, ITAE) computed for the speed, current, and control input signals.

Table 3.8: DNN estimation RMSE evolution through periodic retraining.

Arch.	Load	Ref.	Initial	Before 1st	After 1st	Before 2nd	After 2nd	1st Stage Improv.	Overall Improv.
DNN-1 [6 3]	Sine	Sine	1.85	1.82	1.45	1.58	1.22	↓20.3%	↓34.1%
	Sine	Step	2.15	2.10	1.68	1.85	1.45	↓20.0%	↓32.6%
	Step-1	Sine	1.95	1.90	1.52	1.70	1.38	↓20.0%	↓29.2%
	Step-1	Step	2.25	2.18	1.88	2.05	1.72	↓13.8%	↓23.6%
	Step-2	Sine	2.05	2.00	1.62	1.82	1.50	↓19.0%	↓26.8%
	Step-2	Step	2.35	2.28	2.00	2.18	1.85	↓12.3%	↓21.3%
DNN-2 [6 4 3]	Sine	Sine	2.45	2.40	1.68	1.95	1.25	↓30.0%	↓49.0%
	Sine	Step	2.85	2.78	1.75	2.10	1.30	↓37.1%	↓54.4%
	Step-1	Sine	2.55	2.48	1.78	2.08	1.42	↓28.2%	↓44.3%
	Step-1	Step	2.95	2.85	1.98	2.35	1.58	↓30.5%	↓46.4%
	Step-2	Sine	2.65	2.58	1.88	2.20	1.52	↓27.1%	↓42.6%
	Step-2	Step	3.05	2.95	2.15	2.52	1.75	↓27.1%	↓42.6%
Average DNN-1			2.10	2.05	1.69	1.86	1.52	↓17.6%	↓27.9%
Average DNN-2			2.75	2.67	1.87	2.20	1.47	↓30.0%	↓46.5%

Taken together, these results demonstrate a clear and consistent trend: the deeper DNN-2 architecture provides superior estimation accuracy, enhanced tracking performance, and smoother control effort when compared to the shallower DNN-1 structure. This performance advantage is evident across all tested operating conditions, reinforcing the benefit of increased network depth within the proposed adaptive learning framework. **Table 3.8** summarizes the evolution of the DNN estimation

RMSE throughout the hybrid learning process, evaluated across two architectures [6 3] and [6 4 3], three load profiles (Sine, Step-1, Step-2), and two reference trajectories (Sine, Step). RMSE values are reported at five key instants in the simulation: the initial time ($t \approx 0$ s), immediately before the first retraining event ($t \approx 10$ s) immediately after the first retraining, before the second retraining ($t \approx 20$ s), and after the second retraining. This structure captures the learning dynamics of each architecture and highlights how the retraining events reshape the internal representation of the network.

The first retraining event yields an average improvement of 17.6% for DNN-1 and 30.0% for DNN-2, indicating that the deeper network adapts more effectively when transitioning from offline generic features, obtained under constant load conditions, to features tailored to time-varying operational data. The second retraining further refines the approximation, producing total RMSE reductions of 27.9% for DNN-1 and 46.5% for DNN-2 by the end of the simulation. Between the two retraining events ($t = 10$ – 20 s), both architectures experience a temporary RMSE increase (typically 10–20%), coinciding with the transition from a ramp-constant speed pattern (0–120 rad/s) to sinusoidal tracking (60–180 rad/s). This behavior reflects the expected sensitivity of the learned model to distribution shifts as the operating region expands.

Across all cases, DNN-2 consistently outperforms DNN-1, achieving 70% higher improvement in the first stage (30.0% vs. 17.6%) and a 3.1% lower final RMSE (1.47 vs. 1.52). This advantage is especially pronounced under step reference trajectories, where DNN-2 achieves 44–47% overall improvement, compared to 21–33% for DNN-1. Interestingly, DNN-2 begins with a slightly higher initial RMSE (2.75 vs. 2.10), suggesting that deeper networks may trade a small amount of offline accuracy for increased representational capacity; however, this disadvantage is quickly eliminated once online adaptation begins.

The load profile significantly influences learning behavior. Step-type disturbances (Step-1 with a transition at 25 s and Step-2 with transitions at 15 s and 25 s) lead to 21–43% improvements, driven by discontinuities that were absent from the offline training set (which used a constant load torque of $T_L = 0.18$ Nm), forcing the networks to construct new internal representations. Reference trajectory type also affects performance: step references (60→120→200 rad/s with steady plateaus) produce slightly better overall improvements than sinusoidal references, yielding 27.8% vs. 25.4% for DNN-1 and 46.9% vs. 43.2% for DNN-2. This improvement is attributed to

the presence of sustained steady-state intervals that allow more complete convergence of the adaptive outer-layer weights prior to retraining.

Table 3.9 summarizes the tracking accuracy, current regulation performance, and control effort for four representative scenarios using the standard time-domain indices ISE, IAE, ITSE, and ITAE, enabling a rigorous comparison between the two network architectures. Across all cases, DNN-2 consistently outperforms DNN-1, particularly in the speed-tracking metrics. Specifically, DNN-2 achieves 43–86% reductions in speed ISE, with the largest gains observed under step reference trajectories (85% improvement for both Step/Sine and Step/Step1 cases), where the abrupt changes in commanded speed present substantial challenges to transient control performance.

The superiority of DNN-2 is further reinforced by the time-weighted indices, with ITSE improvements of 79–94% and ITAE improvements of 67–82%, confirming that the observed benefits extend throughout the full duration of operation rather than being confined to initial transients. The fact that these improvements remain consistent across both smooth sinusoidal scenarios (43–94%) and discontinuous step scenarios (67–85%) highlights the architectural robustness of DNN-2 in handling widely varying operating conditions.

Furthermore, DNN-2 achieves 5–22% reductions in the voltage ISE, with the most substantial savings (22%) appearing in the more demanding step-based trajectories. These results indicate that while DNN-1 is capable of maintaining acceptable performance, the deeper DNN-2 architecture leverages its richer internal representation to achieve comparable or improved tracking accuracy with noticeably lower control effort. This reduction in actuator demand is particularly valuable in energy-constrained, battery-powered, or thermally limited systems, where minimizing inverter stress and peak current requirements is essential. On average, the 19% reduction in voltage effort achieved by DNN-2 directly translates to lower inverter switching losses and contributes to improved long-term reliability.

Table 3.9: Comprehensive performance indices across representative scenarios.

		Sine/Sine			Step/Sine			Step/Step1			Sine/Step2		
Signal	Metric	DNN-1	DNN-2	Improv.	DNN-1	DNN-2	Improv.	DNN-1	DNN-2	Improv.	DNN-1	DNN-2	Improv.
Speed	ISE	29.85	17.13	-42.6%	63.32	9.20	-85.5%	68.50	10.25	-85.0%	35.20	20.10	-42.9%
	IAE	14.01	5.18	-63.0%	12.81	3.47	-72.9%	13.85	3.85	-72.2%	15.50	6.20	-60.0%
	ITSE	101.9	6.4	-93.7%	576.6	118.1	-79.5%	620.5	128.5	-79.3%	125.0	8.5	-93.2%
	ITAE	115.2	20.3	-82.4%	108.5	34.2	-68.5%	115.0	38.0	-67.0%	120.0	25.0	-79.2%
iq	ISE	646	644	-0.3%	661	645	-2.4%	794	776	-2.3%	710	695	-2.1%
	IAE	158.4	158.3	-0.1%	158.6	158.4	-0.1%	176.3	175.8	-0.3%	165.2	164.8	-0.2%
	ITSE	12062	12032	-0.2%	12210	12075	-1.1%	18131	17850	-1.6%	13500	13300	-1.5%
	ITAE	3063	3063	0.0%	3067	3066	0.0%	3772	3760	-0.3%	3250	3240	-0.3%
vq	ISE	28.40	26.91	-5.2%	32.44	25.43	-21.6%	35.60	27.80	-21.9%	30.20	28.10	-7.0%
	IAE	29.55	29.40	-0.5%	27.14	25.88	-4.6%	30.88	28.50	-7.7%	30.00	29.50	-1.7%
	ITSE	419.1	401.2	-4.3%	319.4	259.4	-18.8%	388.0	315.0	-18.8%	445.0	420.0	-5.6%
	ITAE	534.2	532.0	-0.4%	337.4	331.8	-1.7%	538.1	510.0	-5.2%	550.0	540.0	-1.8%

3.3.6. Digital Implementation

The experimental test bench developed for this study consists of a fully self-contained low-voltage PMSM drive system designed to allow real-time execution of the proposed DNN-based adaptive backstepping controller on embedded hardware.

The hardware setup is built around the Texas Instruments LAUNCHXL-F28379D control board, which integrates a dual-core TMS320F28379D C2000 microcontroller operating at a 200 MHz CPU clock. This platform provides floating-point support, high-resolution PWM generation, and low-latency ADC and QEP peripherals, making it well suited for implementing advanced nonlinear control algorithms in real time.

Motor actuation is achieved using a BOOSTXL-DRV8305EVM three-phase inverter module, which supplies gate-driver functions, current-sense amplification, protection circuitry, and three low-side shunt resistors for phase-current measurement. **Figure 3.33** shows the overview of the hardware layout of the Texas Instruments LAUNCHXL-F28379D control board, showing major functional components including the onboard XDS100v2 debug probe, TMS320F28379D microcontroller, BoosterPack connectors, ADC inputs, communication interfaces, and configuration jumpers.

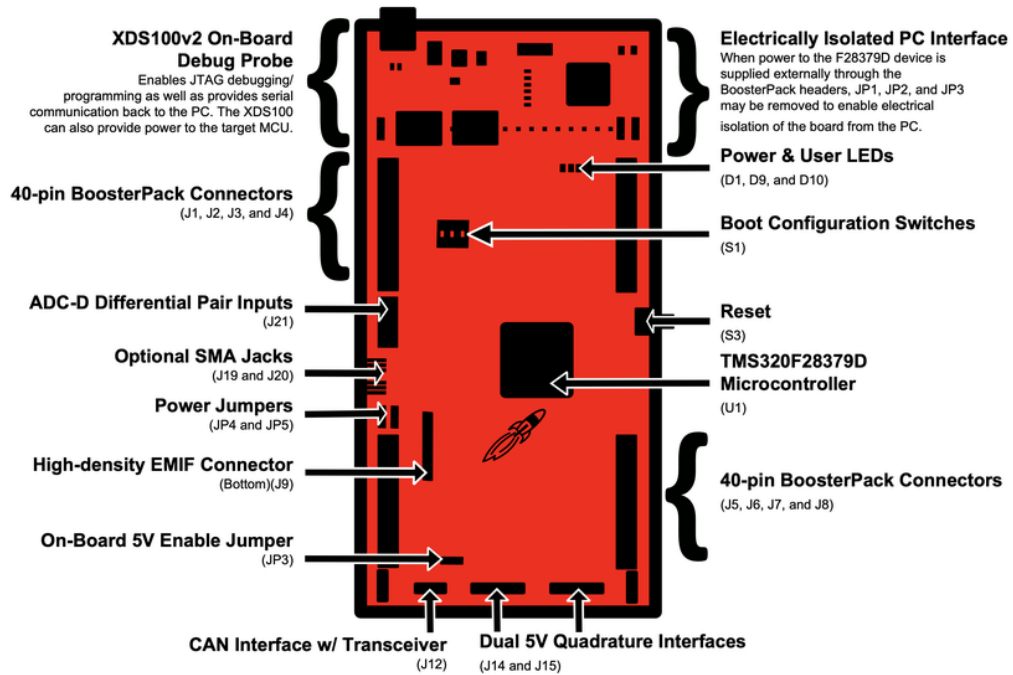


Figure 3.33: Hardware layout of LAUNCHXL F28379D development kit [46].

Figure 3.34 shows the pinout and signal mapping of the BOOSTXL DRV8305EVM three-phase gate driver module, showing power connections, motor outputs, SPI interface pins, current-sense inputs, and PWM control signals used to drive the inverter stage.

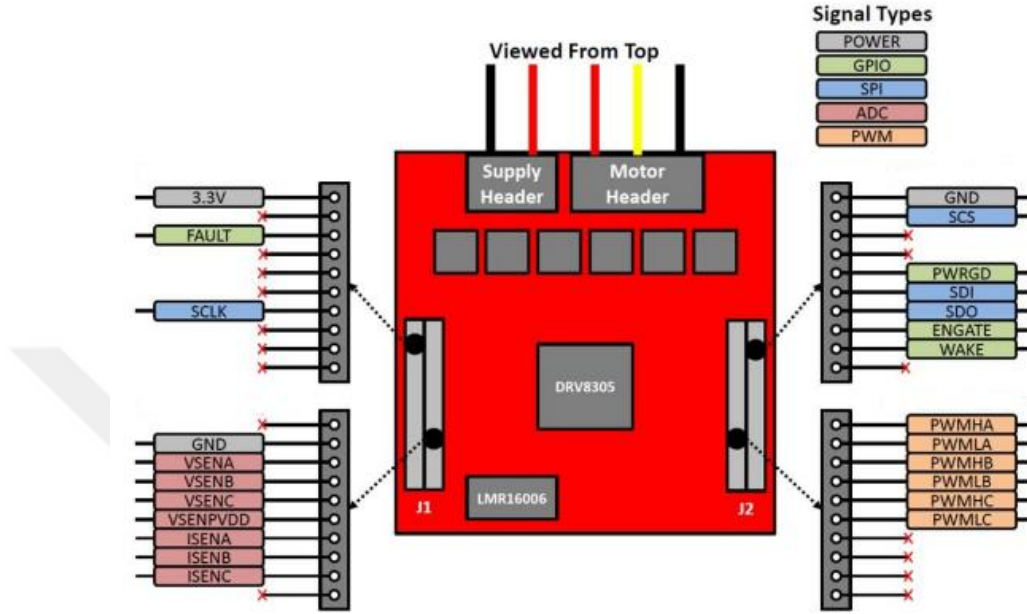


Figure 3.34: Pinout diagram of BOOSTXL DRV8305EVM motor driver [47].

The proposed control algorithm is executed on CPU1 using a fixed-step architecture tailored for deterministic timing. The current control loop is implemented inside an EPWM-triggered interrupt service routine (ISR) running at 20 kHz, which includes Clarke/Park transformations, current feedback processing, voltage command computation, and the adaptive terms of the DNN-augmented backstepping law. A slower 10ms background task handles supervisory functions such as data logging, parameter monitoring, and reference trajectory generation. The DRV8305's analog front-end conditions the phase-current signals, which are subsequently, sampled using the MCU's 12-bit ADCs precisely at the center of the PWM period to reduce switching-induced measurement noise. Rotor electrical position and mechanical speed are obtained from the Teknic M-2310P-LN-04K motor's incremental encoder, which provides 1000 slits (4000 counts per mechanical revolution in 4× quadrature decoding mode). The on-chip enhanced quadrature encoder pulse (EQEP) module, enabling high-accuracy angle reconstruction and real-time mechanical speed estimation, captures the encoder signals.

The load torque applied to the PMSM shaft is measured using a Burster 8661-4500-V0200 torque transducer, whose ± 10 V analog output is level shifted and scaled to the 0–5 V input range of the Arduino Nano ADC by the signal-conditioning circuit shown in **Figure 3.35**.

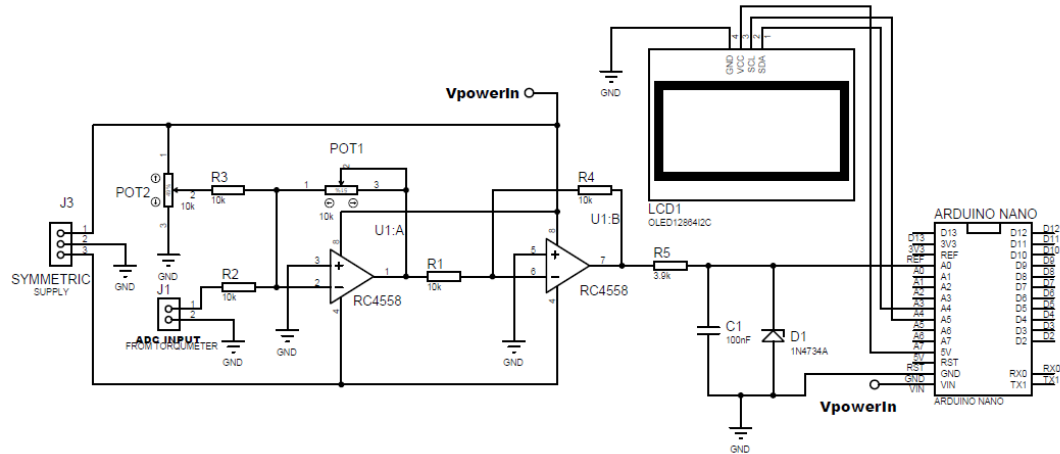


Figure 3.35: Designed circuit scheme for the load torque display.

The transducer output is connected to J1 and first processed by a level-shifter stage based on an op-amp adder (U1:A, RC4558). In this stage, the inverting input sums the sensor signal with a constant +10V reference adjusted by POT2, such that -10 V, 0 V, and $+10$ V at the sensor are translated to 0 V, 10 V, and 20 V, respectively, at the stage output. The second stage is an attenuation stage implemented as an inverting amplifier (U1:B) with a gain of approximately $1/4$, tuned by POT1, which compresses the 0 – 20 V range to 0 – 5 V. As a result, -10 V, 0 V, and $+10$ V at the transducer correspond to 0 V, 2.5 V, and 5 V at the Arduino input; therefore, 0 – 2.5 V represents negative torque and 2.5 – 5 V represents positive torque. The scaled signal is fed through R5 and a first-order low-pass network (R5-C1) to attenuate high-frequency noise; while a 5.1 V zener diode D1 (1N4734A) limits the maximum voltage to protect the microcontroller input. Using the known torque constant of the Burster sensor, the Arduino converts the conditioned voltage to torque and displays the measured value on a 128×64 I²C OLED module (LCD1). Printed Circuit Board (PCB) of the designed circuit is shown in **Figure 3.36**. For the C-based Arduino code of the torque measurement circuit, please see the Appendix-B.

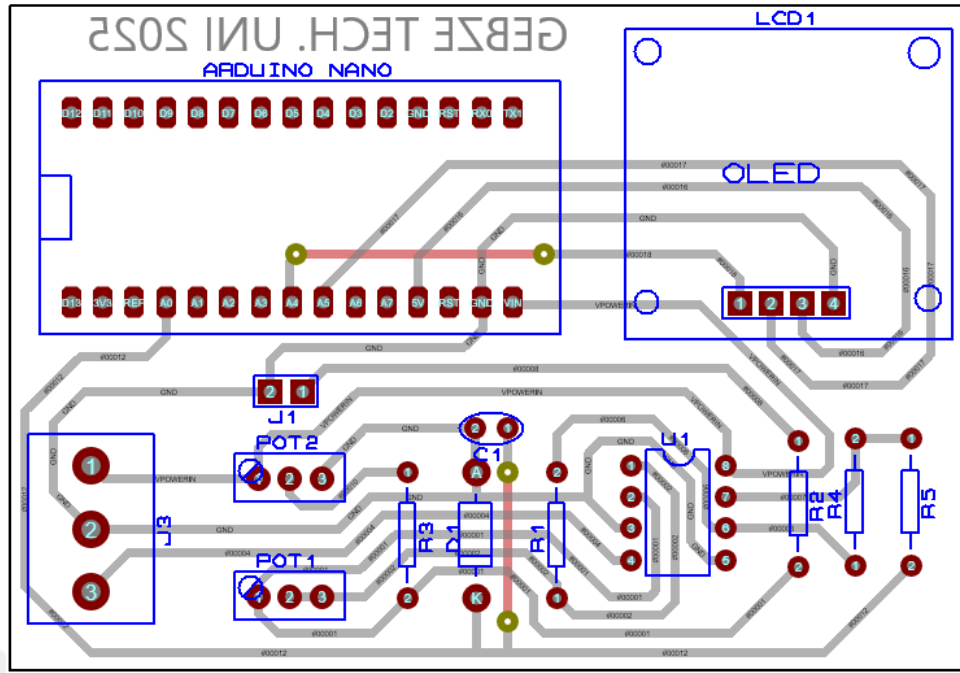


Figure 3.36: Bottom PCB view of the load torque display circuit.

To evaluate the controller under varying load conditions, the motor shaft is mechanically coupled to a torque transducer and a DC generator-potentiometer assembly. **Table 3.10** shows the specifications of DC generator. This setup allows controlled application of load torque while simultaneously measuring the electromagnetic torque produced. A regulated 24 V DC laboratory power supply energizes the inverter DC bus, ensuring stable operation during all tests. All experiments are conducted under sensed field-oriented control, with the full DNN-based adaptive backstepping strategy, comprising the backstepping loops, adaptive laws, robustifying terms, and neural approximation, executed directly on the F28379D without reliance on any external computational resources. A photograph of the complete experimental setup is provided in **Fig. 3.37**.

Table 3.10: Specifications of DC generator [48].

Parameters	Values
Operating Voltage (V)	3–9V
Free-run Speed (RPM)	646
Free-run Current (mA)	80
Stall Current (A)	1
Stall Torque (kg.cm)	1.5
Gear Ratio	1:9.6

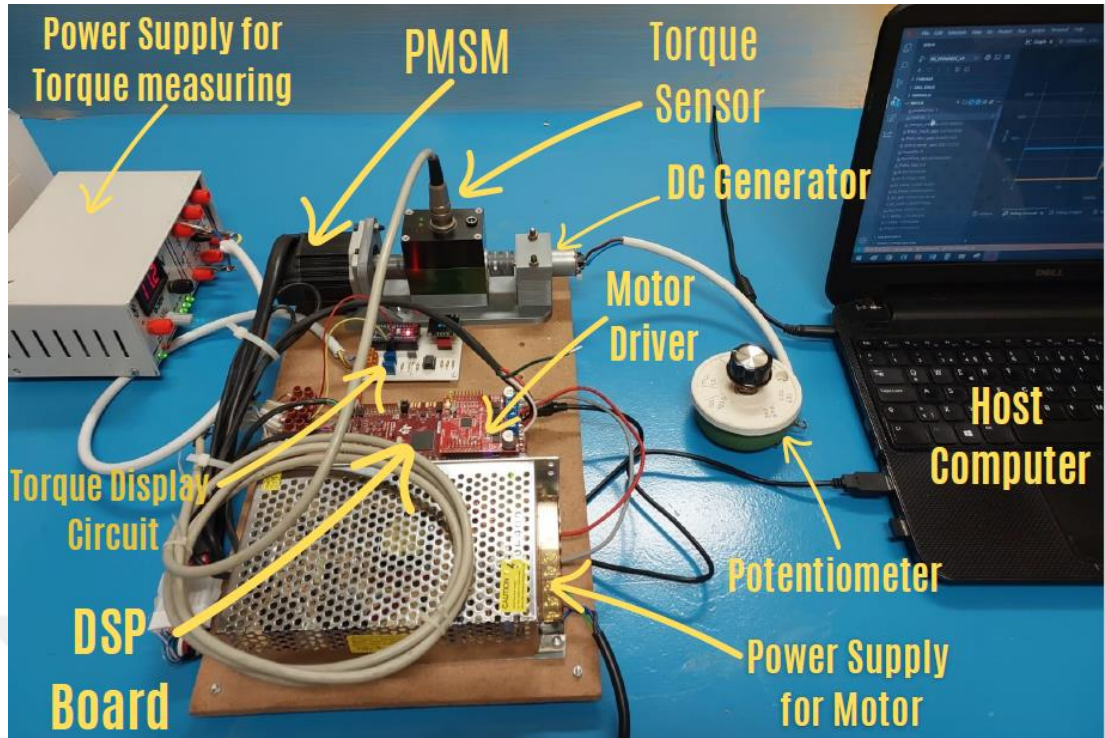


Figure 3.37: Experimental setup.

Table 3.11 shows the materials and their definitions used during building experimental setup. A platform is also designed in 3D for mounting and coupling PMSM, torque sensor and DC generator. See the Appendix-C for the designed 3D view of the mounting platform.

Table 3.11: Components of the experimental setup.

Component Name	Definition
PMSM	Teknic M-2310P-LN-04K
DSP Board	LAUNCHXL F28379D
Motor Driver Module	BOOSTXL DRV8305EVM
DC Generator	JGA320-25 620RPM
Potentiometer	100W, 100 Ω
Power Supply for Motor	24V 10A 240W SMPS, Metal Case
Power Supply for Torque Display	$\pm 12V$ 2A Symmetric, Lab. Type
Torque Transducer	Burster 8661 4500 v0200

In contrast to the MATLAB simulations, which incorporate both offline pretraining and periodic online retraining of the DNN's inner layers, the embedded implementation is subject to stringent timing, memory, and computational constraints. Real-time training of multilayer neural networks, particularly using second-order methods such as Levenberg-Marquardt, is not feasible on the F28379D, given its limited RAM, the absence of dedicated matrix-acceleration hardware, and the

requirement to meet a 20 kHz control deadline with deterministic timing guarantees. For this reason, the experimental system employs a frozen-feature architecture: the nonlinear feature mapping generated by the DNN's inner layers is trained offline in MATLAB using the pre-collected PMSM dataset and then transferred to the microcontroller as fixed-weight matrices and bias vectors.

During operation, these inner-layer parameters remain constant, and only the output-layer weights and bias are adapted online according to the update law. This preserves the expressive power of the DNN in the operating region—since the learned nonlinear basis functions span the relevant state space—while ensuring that the online computational burden remains compatible with the microcontroller's real-time constraints. The outer-layer adaptation requires only vector-scalar multiplications and is therefore easily executed at 20 kHz within the ISR. As a result, the embedded implementation realizes the adaptive backstepping structure and online learning mechanism described in the theoretical development, while adhering to the practical limitations of the digital control hardware. In MATLAB, periodic inner-layer retraining was applied as an additional study to investigate the effect of feature refinement; however, the stability analysis and embedded implementation both correspond to the frozen-feature case described below.

To assess the closed-loop tracking performance in a manner consistent with the MATLAB simulations, the experiments employ reference-speed profiles that are practical counterparts of the two trajectories used in the simulation study. In simulation, *Profile 1* consists of a ramp (0–8s), constant-speed (8–16s), and sinusoidal phase (16–40s) defined in rad/s as $\omega^*(t) = 100 + 50\sin(2\pi ft)$ with $f = 1/(T_{\text{end}} - 16)\text{Hz}$, while *Profile 2* is a multi-step reference with step transitions between 60, 120, and 200rad/s to test adaptability under abrupt changes. For the real-time experiments, two analogous profiles are implemented in RPM over a longer 0–120s horizon. In the first scenario, a four-step mechanical-speed trajectory (400–600–800–1000RPM) mirrors the multi-step structure of *Profile 2* is used to evaluate the transient response under abrupt setpoint changes. In the second scenario, a composite ramp-constant-sinusoidal profile over the 0–120s interval mirrors the structure of *Profile 1*, comprising a ramp from standstill to the nominal speed, a constant-speed segment, and a low-frequency sinusoidal variation around the nominal operating point. The corresponding experimental results for these two scenarios are presented in the following figures.

All experimental variables were recorded using a dedicated logging buffer of length $N_{\log} = 1000$ with a logging period of $T_{\log} = 0.12$ s (i.e., $f_{\log} = 8.33$ Hz). The adaptive backstepping and DNN control loops execute at 20 kHz, so the relatively low logging rate does not affect real-time performance while still providing sufficient temporal resolution to visualize the 120s experiments.

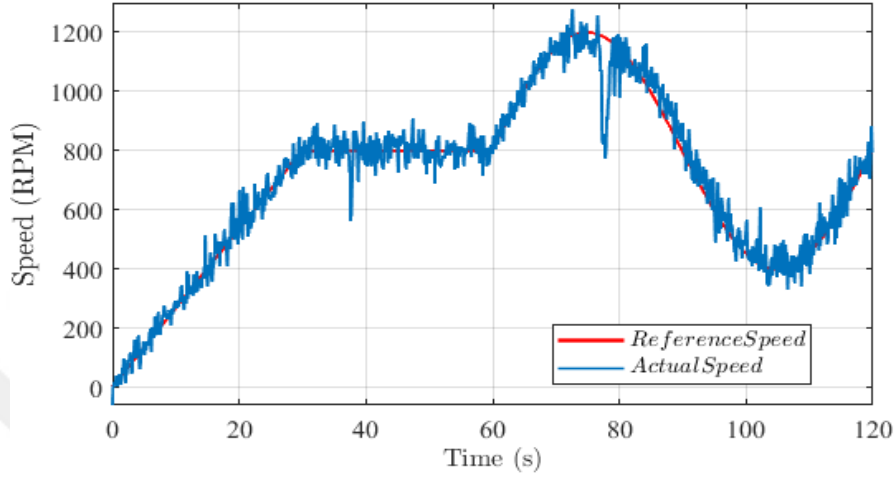


Figure 3.38: Speed-tracking performance of Scenario 1.

Figure 3.38 presents the measured mechanical-speed tracking performance of the embedded DNN-based adaptive backstepping controller for the reference profile shown. A constant load torque of 0.1Nm is applied at the beginning of the test to represent a nonzero operating load. To assess disturbance rejection, the load torque is then increased abruptly to 0.15Nm (50% step) during 5 seconds at $t=35$ s and again at $t=75$ s, emulating short torque pulses. In both events, the controller exhibits a brief speed droop and a short-lived oscillatory transient at the disturbance edges, consistent with the sudden increase in required electromagnetic torque. However, the speed recovers rapidly to the reference and remains well regulated throughout the subsequent segments, with no sustained offset or drift after the pulses are removed. The temporary downshoots remain bounded and localized in time, indicating effective rejection of abrupt load-torque variations under real-time implementation constraints.

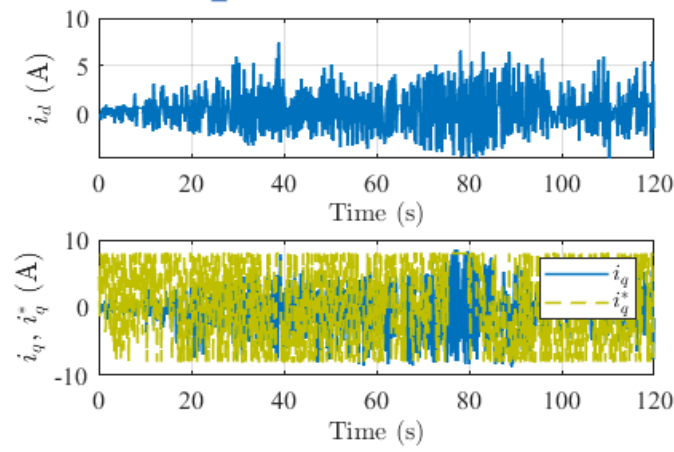


Figure 3.39: Currents for scenario 1.

Figure 3.39 shows the measured i_d and the q -axis current tracking (i_q and i_q^*) during the 0-120 s experiment. The d -axis current remains regulated around zero, which is consistent with sensored FOC operation and indicates that the controller maintains the intended decoupling/flux-oriented condition despite measurement noise and switching ripple. The actual i_q follows the commanded i_q^* over the full test, with brief deviations and higher ripple during the more demanding intervals where a rapid change in required electromagnetic torque forces a fast current adjustment.

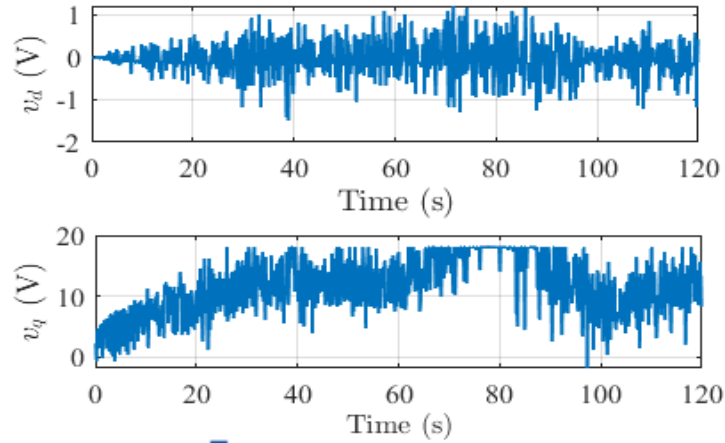


Figure 3.40: Control inputs for scenario 1.

Figure 3.40 reports the commanded d - and q -axis voltage components, v_d and v_q , generated by the embedded controller during the experiment. The v_d command stays centered near zero with comparatively small magnitude, which is consistent with regulating $i_d \approx 0$ and compensating only for residual coupling and measurement noise. In contrast, v_q varies over a much wider range because it supplies the dominant torque-producing action needed to realize the speed reference and to counter the applied load-torque pulses. The increased v_q demand during higher-speed intervals and the brief

spikes during disturbance or operating-point transitions reflect rapid modulation of the inverter output to restore i_q and maintain mechanical-speed tracking under real-time constraints.

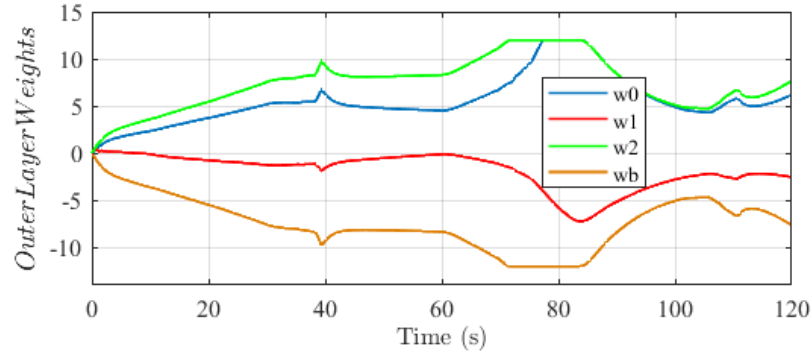


Figure 3.41: Outer layer weights update for Scenario 1.

Figure 3.41 shows the online evolution of the DNN output-layer parameters (weights w_0 , w_1 , w_2 and bias w_b) updated in real time by the adaptive law while the inner-layer feature mapping remains fixed. The parameters change smoothly and remain bounded over the full 0–120 s interval, indicating numerically stable adaptation under the 20 kHz ISR execution. Noticeable slope changes and short transients appear around the operating-point and disturbance intervals, where the approximation error and required torque compensation increase; the adaptive law responds by adjusting the outer-layer coefficients to maintain the learned uncertainty compensation. Overall, the trajectories confirm that the embedded implementation realizes continuous online learning without drift or instability of the estimated weights.

Figure 3.42 shows the measured mechanical-speed tracking response for Scenario 2 (stepwise reference profile) under the same load-torque disturbance protocol used in Scenario 1. A constant load torque of 0.1Nm is applied from the start of the experiment, and two torque pulses are introduced by stepping the load to 0.15Nm (50% increase) during 5 seconds at $t=55s$, 85s and 105s. Across the successive reference steps, the actual speed tracks the commanded levels with a short transient at each setpoint change and returns to steady operation without sustained offset.

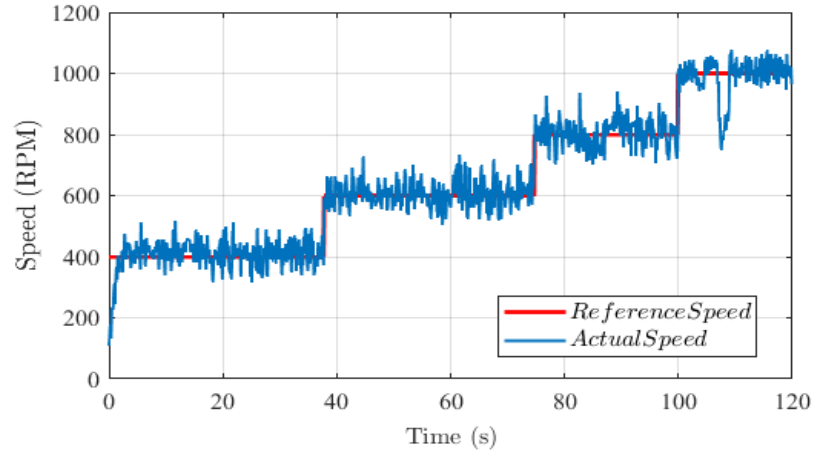


Figure 3.42: Speed-tracking performance of scenario 2.

During the torque pulses, the speed exhibits brief droop and localized oscillations at the disturbance edges, but the deviation remains bounded and the controller restores the speed to the reference shortly after the pulse begins and again after it is removed. Overall, the results indicate that the embedded controller maintains stable tracking performance for abrupt setpoint transitions while rejecting short-duration load-torque disturbances.

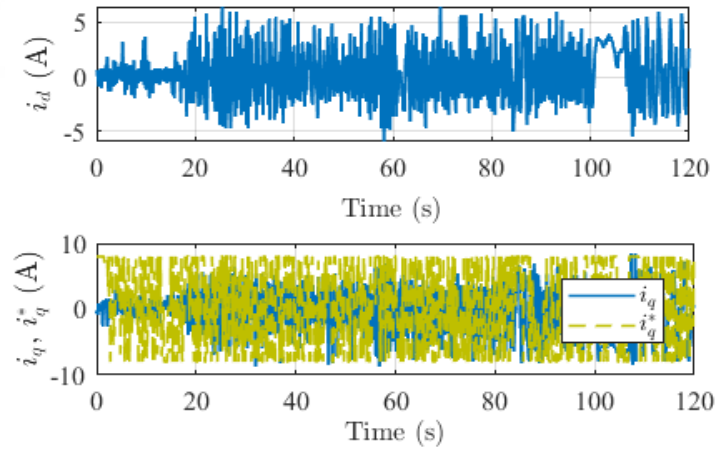


Figure 3.43: Currents for scenario 2.

Figure 3.43 shows that current i_d is regulated close to zero throughout Scenario 2, indicating that the field-oriented condition and axis decoupling are preserved despite the step changes and load-torque pulses. The q -axis current i_q tracks its command i_q^* with brief, bounded deviations during the setpoint transitions and disturbance intervals, reflecting the fast torque demand changes required to maintain speed tracking.

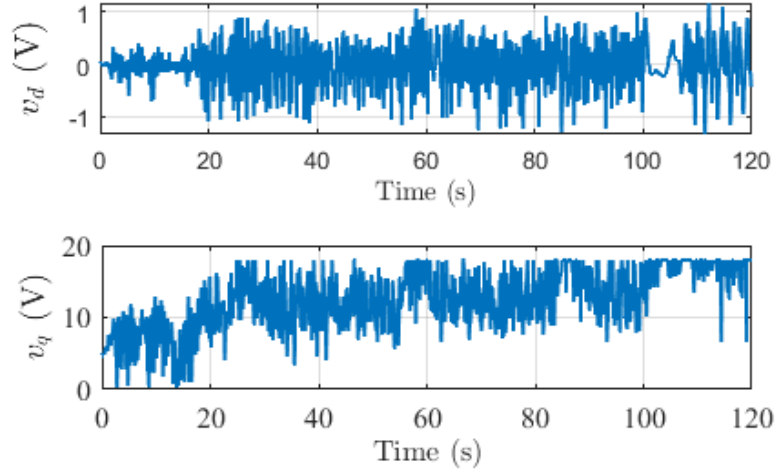


Figure 3.44: Control inputs for scenario 2.

Figure 3.44 presents the commanded v_d and v_q components during Scenario 2. The v_d signal remains small and centered near zero, consistent with maintaining $i_d \approx 0$, while v_q varies over a wider range to supply the torque-producing action required by the stepwise speed commands and the 0.15Nm disturbance pulses, showing brief spikes/dips at the transition instants.

Figure 3.45 shows the real-time evolution of the DNN output-layer parameters (w_0 , w_1 , w_2 , w_b) for Scenario 2. The weights remain bounded and change smoothly, with clear adjustment transients aligned with the reference-step events and the load-torque pulses, indicating stable online adaptation that increases uncertainty/torque compensation when the operating point and disturbance level change.

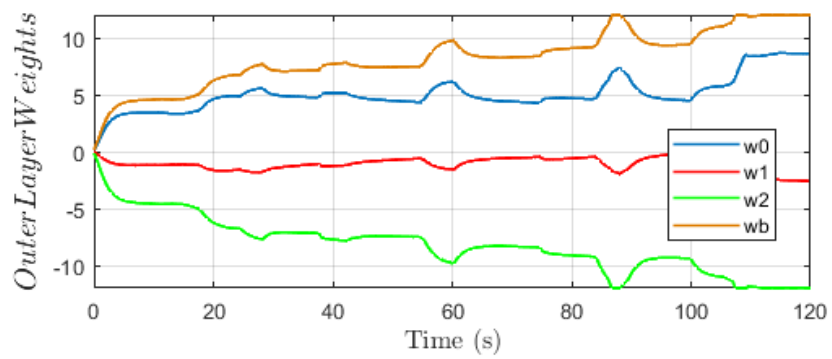


Figure 3.45: Outer layer weights update for Scenario 2.

4. CONCLUSION AND FUTURE WORK

This thesis investigated the soft computing assisted nonlinear control of permanent magnet synchronous motor (PMSM) drives under field-oriented control (FOC), with emphasis on high-accuracy tracking, robustness to external load disturbances, and tolerance to uncertain or poorly known electromechanical parameters. Two complementary backstepping-based control architectures were developed.

The first contribution established an adaptive backstepping controller for the cascaded FOC structure, regulating the speed loop and the dq current loops while compensating lumped uncertainties and disturbances. A Lyapunov-based design was used to derive stabilizing laws and adaptation mechanisms and to provide analytical global stability guarantees. To address the practical sensitivity of backstepping performance to gain selection across varying operating points, a fuzzy-logic tuning mechanism was integrated to adjust key control parameters in real time. Simulation studies demonstrated improved transient response (faster convergence and reduced overshoot) and tracking accuracy relative to conventional PI control, particularly under dynamic speed commands and load variations.

The second contribution developed a DNN-based adaptive backstepping controller within the same FOC framework. Unmodeled dynamics, parameter variations (e.g., resistance and inertia changes), and load disturbances were treated as a lumped nonlinear term approximated by the DNN. A hybrid learning strategy was adopted: the DNN's inner layers were trained offline using data collected from a PI-controlled drive to form a nonlinear feature mapping, while the output layer was adapted online via Lyapunov-based update laws. A rigorous Lyapunov analysis established boundedness of all closed-loop signals and global asymptotic convergence of both speed and current tracking errors under the stated assumptions and a verifiable gain condition. Simulation results showed accurate tracking and strong disturbance rejection across demanding reference profiles (ramp-constant-sinusoidal and multi-step) and load-torque variations; the deeper bottleneck architecture ([6-4-3]) consistently provided smoother control action and better compensation than the shallower ([6-3]) structure. In addition, periodic retraining of the inner layers using newly collected closed-loop data was studied in simulation and indicated that feature refinement can further

improve approximation quality and tracking smoothness when computational resources are available.

Beyond simulation, the thesis demonstrated real-time feasibility on a DSP-based motor-drive platform (TI LAUNCHXL-F28379D with DRV8305 inverter) at a 20 kHz interrupt rate. Due to deterministic timing and memory constraints, the embedded implementation employed a frozen-feature configuration: offline-trained inner-layer parameters were deployed as fixed matrices/biases, while only the outer-layer parameters were updated online inside the ISR.

Experimental results for two scenarios confirmed: (i) close speed tracking for both stepwise and composite reference profiles, (ii) effective rejection of repeated 50% load-torque pulses ($0.1 \rightarrow 0.15$ Nm for 2s) with bounded and short transients, (iii) regulation of i_d near zero and tracking of i_q to i_q^* and (iv) bounded, smooth evolution of the outer-layer weights without drift. Collectively, the outcomes show a practical and stability-certified path from classical nonlinear adaptive control to data-driven uncertainty compensation for PMSM drives under real-time embedded constraints.

Future work will focus on extending the proposed controllers to sensorless/observer-based FOC and making the adaptive elements more hardware-aware by explicitly handling inverter limits, dead-time, and measurement noise to improve robustness near saturation. A second direction is the implementation of fully online adaptation by updating all inner and outer layer weights, together with an extended Lyapunov-based stability analysis that addresses nonlinear parameterization under embedded real-time constraints.

REFERENCES

- [1] Strategic Market Research, (2023), “Permanent Magnet Synchronous Motor (PMSM) Market Size, Trends, Forecasts 2023–2030,” Strategic Market Research. [Online]. Available: <https://www.strategicmarketresearch.com/market-report/permanent-magnet-synchronous-motor-market>. Accessed: [20.10.2025]
- [2] Market Growth Reports, (2023), “Permanent Magnet Synchronous Motor (PMSM) Market Insights and Forecast to 2030,” Market Growth Reports. [Online]. Available: <https://www.marketgrowthreports.com/market-reports/permanent-magnet-synchronous-motor-pmsm-market-109563>. Accessed: [20.10.2025]
- [3] Bose B.K., (2022), “Modern Power Electronics and AC Drives”, Upper Saddle River, NJ, USA: Prentice Hall.
- [4] Xu Y., Yao M., Sun X., (2023), “Overview of position sensorless technology for permanent magnet synchronous motor systems,” World Electric Vehicle Journal, 14(8), 212.
- [5] Krishnan R., (2013), “Electric Motor Drives – Modeling, Analysis and Control”, Upper Saddle River, NJ, USA: Prentice Hall.
- [6] Sakunthala S., Kiranmayi R., Mandadi P.N., (2018), “A review on speed control of permanent magnet synchronous motor drive using different control techniques”, 2018 International Conference on Power, Energy, Control and Transmission Systems (ICPECTS), 97-102, Chennai, India, 22-23 February 2018.
- [7] Thakare J., Unde S (2023), “Comparative analysis of performance of different speed controllers for FOC based SVPWM fed PMSM drive”, 3rd Asian Conference Innovation in Technology (ASIANCON), 1-6, Ravet, India, 25-27 August 2023.
- [8] Wang X., Chen Y., Lu Y., Li X., He W., (2019), “Dynamic surface method-based adaptive backstepping control for the permanent magnet synchronous motor on parameter identification,” Proceedings of the Institution of Mechanical Engineers, Part I: Journal of Systems and Control Engineering, 233(9), 1172-1181.
- [9] Ettalabi N., Bouzi M., Derri M., (2017), “Comparison study of non-adaptive and adaptive backstepping in control of permanent magnet synchronous motor,” International Journal of Innovative Research in Science Engineering and Technology, 6(8), 17792-17804.
- [10] Bayır F., Zergeroğlu E., (2025), “Adaptive backstepping control with real-time fuzzy logic parameter selection of a field oriented control based permanent magnet synchronous motor driver,” Turkish Journal of Electrical Engineering and Computer Sciences, 33(4), 460-478.
- [11] Mohd Z.F., Mekhilef S., Mubin M., (2019), “Robust speed control of PMSM using sliding mode control (SMC)—A review,” Energies, 12(9), 1669.

- [12] Wang H., Li S., Yang J., Zhou X.P., (2016), "Continuous sliding mode control for permanent magnet synchronous motor speed regulation systems under time-varying disturbances," *Journal of Power Electronics*, 16(4), 1324-1335.
- [13] Anh H.P.H., Van K. C., Huan T.T., Khanh P.Q., (2018), "Advanced speed control of PMSM motor using neural FOC method", 4th International Conference on Green Technology and Sustainable Development (GTSD'18), 696-701, Ho Chi Minh City, Vietnam, 23-24 November 2018.
- [14] Li S., Won H., Fu X., Fairbank M., Wunsch D.C., et al., (2020), "Neural-network vector controller for permanent-magnet synchronous motor drives: Simulated and hardware-validated results," *IEEE Transactions on Cybernetics*, 50(7), 3218-3230.
- [15] Zhang X., Zhang L., Zhang Y., (2019), "Model predictive current control for PMSM drives with parameter robustness improvement", *IEEE Transactions on Power Electronics*, 34(2), 1645-1657.
- [16] Nicola M., Nicola C.I., (2020), "Sensorless control of PMSM using backstepping control and ESO-type observer", 12th International Conference on Electronics, Computers and Artificial Intelligence (ECAI), 1-6, Bucharest, Romania, 25-27 June 2020.
- [17] Trabelsi R., Hosseyni A., Iqbal A., Mimouni M., (2015), "Backstepping control for a five-phase permanent magnet synchronous motor drive," *International Journal of Power Electronics and Drive Systems*, 6(4), 683-692.
- [18] Sun X., Yu H., Yu J., Liu X., (2019), "Design and implementation of a novel adaptive backstepping control scheme for a PMSM with unknown load torque," *IET Electric Power Applications*, 13(4), 445-455.
- [19] Shen M., Ma Y., Park J.H., Wang Q.-G., (2022), "Fuzzy tracking control for Markov jump systems with mismatched faults by iterative proportional-integral observers," *IEEE Transactions on Fuzzy Systems*, 30(2), 542-554.
- [20] Yang H., Jiang Y., Yin S., (2021), "Adaptive fuzzy fault-tolerant control for Markov jump systems with additive and multiplicative actuator faults," *IEEE Transactions on Fuzzy Systems*, 29(4), 772-785.
- [21] Zhao L., Yang G., Luo K., He L., (2025), "Neuro-adaptive fractional order prescribed performance backstepping control for a class of strict-feedback non-linear systems", *IET Control Theory & Applications*, 19, e12782.
- [22] Mokhtari M., Golea N., (2013), "Control of a Permanent Magnet Synchronous Machine using Neural Network Backstepping Approach", *Advanced Materials Research*, 628, 410-419.
- [23] Sun R., Greene M. L., Le D. M., Bell Z. I., Chowdhary G., Dixon W. E., (2022), "Lyapunov-Based Real-Time and Iterative Adjustment of Deep Neural Networks", *IEEE Control Systems Letters*, 6, 193-198.

- [24] Zheng Y., Cao Z., Rsetam K., Man Z., Wang S., (2024), "Extreme learning machine-based super-twisting integral terminal sliding mode speed control of permanent magnet synchronous motors", *IET Control Theory & Applications*, 18, 2524-2539.
- [25] Sabouni E., Merah B., and Boulkhrachef I., (2021), "Adaptive backstepping controller design based on neural network for PMSM speed control," *Int. J. Power Electron. Drive Syst.*, vol. 12, no. 3, 1940-1952.
- [26] S. Lu, X. Wang, and Y. Li, (2019), "Adaptive neural network control for fractional-order PMSM with time delay based on command filtered backstepping," *AIP Advances*, vol. 9, no. 5, p. 055105, 2019.
- [27] Luo S., Hu X., Zhao L. and Li S., (2022), "Event-triggered neural adaptive backstepping control of the K chaotic PMSGs coupled system," *Int. J. Electr. Power Energy Syst.*, vol. 135, p. 107475, 2022.
- [28] Yao Z., Yoon H.S., and Hong Y.K., (2023), "Control of hybrid electric vehicle powertrain using offline-online hybrid reinforcement learning," *Energies*, vol. 16, no. 2, 652.
- [29] Yankui S., Tuo Y. and Li J., (2023), "A neural adaptive prescribed performance controller for the chaotic PMSM stochastic system", *Nonlinear Dynamics*, vol. 111, 15123–15140.
- [30]. Mishra, A., Agarwal, P. and Srivastava, S. (2014), "A comprehensive analysis and implementation of vector control of permanent magnet synchronous motor," *International Journal of Power and Energy Conversion*, 5, 1–23.
- [31] Jung, W.-S., Lee, H.-K., Lee, Y.-K., Kim, S.-M., Lee, J.-I. and Choi, J.-Y. (2023). "Analysis and Comparison of Permanent Magnet Synchronous Motors According to Rotor Type under the Same Design Specifications", *Energies*, 16(3), 1306.
- [32] M. P. Kazmierkowski, (2002), "Control of PWM inverter-fed induction motors", *Control in Power Electronics*, Academic Press, 161-207.
- [33] Ramesh P., Umavathi M., Bharatiraja C., Ramanathan G. and Athikkal S. (2022), "Development of a PMSM motor field-oriented control algorithm for electrical vehicles," *Mater Today Proc* 65(1):176-187
- [34] Zhang G., (2022), "Development status of space vector pulse width modulation (SVPWM) control method for multi-level inverter", 3rd International Conference on Electrical Technology and Automatic Control (ICETAC), 1-6, Macau, China, 2-4 December 2022.
- [35] Abed W., Aish Q. and Hamad A. (2021), "Performance monitoring of aircraft PMSM based on soft computing technique", *Periodicals of Engineering and Natural Sciences (PEN)*, 9, 293–300.

- [36] Sakhri Z., Bekkour E.-H., Bossoufi B., Bizon N., Almalki M.M., Alghamdi T.A.H. and Alenezi M. (2025), “Soft computing approaches of direct torque control for DFIM motors”, *Cleaner Engineering and Technology*, 24, 100891.
- [37] Zadeh L.A. (1965), “Fuzzy sets”, *Information and Control*, 8(3), 338–353.
- [38] Hornik K., Stinchcombe M. and White H. (1989), “Multilayer feedforward networks are universal approximators”, *Neural Networks*, 2(5), 359–366.
- [39] Fatemimoghadam A., Yan Y., Iyer L.V. and Kar N.C. (2022), “Permanent Magnet Synchronous Motor drive using deep-neural-network-based vector control for electric vehicle applications”, *2022 International Conference on Electrical Machines (ICEM)*, 2358-2364, Valencia, Spain, 5-8 September 2022.
- [40] Lan Y.H., Zhou L., (2018), “Backstepping control with disturbance observer for permanent magnet synchronous motor”, *Journal of Control Science and Engineering*, 2018(1), 1–8.
- [41] Sun Y., Zhou L., Jiang F., Li M. and Xiao W., (2025), “Design of backstepping control for nonlinear systems based on command filter and uncertainty estimation”, *Asian J. Control* 27, 2476–2487
- [42] Sun X., Yu H., Yu J., Liu X., (2019), “Design and implementation of a novel adaptive backstepping control scheme for a PMSM with unknown load torque”, *IET Electric Power Applications*, 13(4), 445–455.
- [43] Ali N., Alam W., Pervaiz M. and Iqbal J., (2021), “Nonlinear adaptive backstepping control of permanent magnet synchronous motor”, *Revue Roumaine des Sciences Techniques - Serie Electrotechnique et Energetique*, 66(1), 15–20.
- [44] Slotine J.J. and Li W., (1991), *Applied Nonlinear Control*, Englewood Cliffs, NJ: Prentice Hall.
- [45] Eskikurt H., Karabacak M., (2014), “Nonlinear and adaptive backstepping speed tracking control of a permanent magnet synchronous motor despite all parameter uncertainties and load torque disturbance variation”, *Proceedings of the 2014 Asia-Pacific Electronics and Electrical Engineering Conference (EPEC 2014)*, 63–70, Shanghai, China, 27-28 December 2014.
- [46] LAUNCHXL-F28379D overview user's guide (Rev. C), (2019), <https://www.ti.com/lit/ug/sprui77c/sprui77c.pdf?ts=1764492621206> Accessed: [21.10.2025]
- [47] BOOSTXL-DRV8305EVM User's Guide (Rev. A), (2017), <https://www.ti.com/tool/BOOSTXL-DRV8305EVM> Accessed: [21.10.2025]
- [48] Gearmotor Selector Webpage (2025), <https://www.openimpulse.com/blog/products-page/product-category/jga25-370-dc-gearmotor-646-rpm-6-v/> Accessed: [29.10.2025]

BIOGRAPHY

Fatih Bayır received his B.Sc. degree in Electronics and Communications Technology from Marmara University in 2007, his second B.Sc. degree in Computer Engineering from Sakarya University in 2014, and his M.Sc. degree in Electrical and Electronics Engineering from Marmara University in 2019. He is currently pursuing a Ph.D. in Electronics Engineering at Gebze Technical University. Alongside his doctoral studies, he currently teaches electronics at a vocational high school in Tuzla, Istanbul. His research areas are nonlinear control, intelligent control systems, and power electronics.



PUBLICATIONS AND PRESENTATIONS FROM THE THESIS

[1] Bayır F., Zergeroğlu E., (2025), “Adaptive backstepping control with real-time fuzzy logic parameter selection of a field oriented control based permanent magnet synchronous motor driver”, Turkish Journal of Electrical Engineering and Computer Sciences, 33(4), 460–478.

[2] Bayır F., Zergeroğlu E., Tatlıcıoğlu E., (2026), “Deep neural network based adaptive backstepping control of permanent magnet synchronous motors”, International Journal of Control, Automation and Systems (IJCAS) **(UNDER REVIEW)**



APPENDICES

APPENDIX-A: EQUATIONS OF BACKSTEPPING AND ADAPTIVE BACKSTEPPING CONTROLLER IN THEIR OPEN FORMS

Virtual Control inputs for backstepping controller:

$$i_q^* = \frac{2}{3P\phi_m} \left(-k_3 e_\omega - \frac{3P}{2} (L_q - L_d) i_d i_q + B\omega_r + T_L + J\dot{\omega}_r^* \right)$$

$$v_d = -k_1 e_d - \omega L_q i_q$$

$$v_q = \frac{\phi_m}{\phi_m + (L_q - L_d) i_d} \left[-k_2 e_q - \frac{3P\phi_m}{2} e_\omega + Ri_q + \omega L_d i_d - \omega \phi_m \right. \\ \left. - \frac{L_q i_q k_3}{J} (1 + (L_q - L_d) i_d) \right. \\ \left. + \frac{2L_q k_3}{3P\phi_m J} (B\omega_r + T_L + J\dot{\omega}^*) \right. \\ \left. + \frac{2L_q}{3P\phi_m} (B\dot{\omega}_r + T_L + J\ddot{\omega}^*) \right. \\ \left. - \frac{L_q (L_q - L_d) i_q}{\phi_m L_d} (-Ri_d + \omega L_q i_q + v_d) \right. \\ \left. - \frac{(L_q - L_d) i_d}{\phi_m} (-Ri_q - \omega L_d i_d + \omega \phi_m) \right]$$

Virtual Control inputs for adaptive backstepping controller:

$$\hat{i}_q^* = \frac{2}{3P\phi_m} \left(-k_1 e - \frac{3P}{2} (L_q - L_d) i_d i_q + \hat{B}\omega_r + \hat{T}_L + \hat{J}\dot{\omega}_r^* \right)$$

$$\hat{v}_d = -k_3 i_d - \omega L_q i_q + \hat{R} i_d$$

$$\begin{aligned}\hat{v}_q = \frac{\phi_m}{\phi_m + (L_q - L_d)i_d} & \left[-k_2 e_q - \frac{3P\phi_m}{2} e + \omega L_d i_d - \omega \phi_m + \frac{2L_q k_1 \omega_r^*}{3P\phi_m} \right. \\ & - \frac{(L_q - L_d)i_d}{\phi_m} (-\omega L_d i_d + \omega \phi_m) \\ & + \frac{2L_q}{3P\phi_m} (B\dot{\omega}_r + T_L + J\omega_r^*) \\ & - \frac{L_q(L_q - L_d)i_q}{\phi_m L_d} (\omega L_q i_q + v_d) \\ & - \hat{R} \left(-i_q - \frac{i_d i_q L_q (L_q - L_d)}{\phi_m L_d} - \frac{i_d i_q (L_q - L_d)}{\phi_m} \right) \\ & - \hat{A} \left(L_q i_q k_1 \frac{(\phi_m + (L_q - L_d)i_d)}{\phi_m} \right) \\ & - \hat{b} \left(\frac{-2L_q k_1 \omega_r}{3P\phi_m} \right) \\ & \left. - \hat{t} \left(\frac{-2L_q k_1}{3P \cdot \phi_m} \right) \right]\end{aligned}$$

where $\hat{A} = (\hat{1}/J)$, $\hat{b} = (B\hat{1}/J)$ and $\hat{t} = (T_L\hat{1}/J)$.

Adaptive Laws:

$$\dot{\hat{\phi}}_1 = -\Gamma_{11} \omega_r e$$

$$\dot{\hat{\phi}}_2 = -\Gamma_{12} e$$

$$\dot{\hat{\phi}}_3 = -\Gamma_{13} \dot{\omega}_r^* e$$

$$\dot{\hat{\theta}}_1 = \Gamma_{21} \left(-i_q - \frac{i_d i_q L_q (L_q - L_d)}{\phi_m L_d} - \frac{i_d i_q (L_q - L_d)}{\phi_m} \right) e_q$$

$$\dot{\hat{\theta}}_2 = \Gamma_{22} \left(L_q i_q k_1 \frac{(\phi_m + (L_q - L_d)i_d)}{\phi_m} \right) e_q$$

$$\dot{\hat{\theta}}_3 = \Gamma_{23} \left(\frac{-2L_q k_1 \omega_r}{3P\phi_m} \right) e_q$$

$$\dot{\hat{\theta}}_4 = \Gamma_{24} \left(\frac{-2L_q k_1}{3P\phi_m} \right) e_q$$

$$\dot{\hat{R}} = -\gamma i_d^2$$

Known or measurable terms for DNN based adaptive backstepping controller:

$$g = -\omega L i_d + \omega \phi_m - \frac{2L k_1}{3P\phi_m} \dot{\omega}_r^* - \frac{2L}{3P\phi_m} (\dot{Y}_1 \hat{\theta} + Y_1 \dot{\hat{\theta}})$$

APPENDIX-B: ARDUINO C CODE FOR TORQUE MEASUREMENT CIRCUIT

```
#include <Wire.h>
#include <Adafruit_GFX.h>
#include <Adafruit_SSD1306.h>

#define SCREEN_WIDTH 128
#define SCREEN_HEIGHT 64
#define OLED_RESET 4

Adafruit_SSD1306 display(SCREEN_WIDTH, SCREEN_HEIGHT, &Wire, OLED_RESET);

const int ADC_PIN = A0;
const int N_SAMPLES = 32; // number of ADC samples per reading
const int DISPLAY_MS = 200; // OLED update period (ms)
int torkVal = 0;
int zeroOffset = 512; // will be calibrated in setup()
float filteredAdc = 0.0f; // low-pass filtered ADC value
const float alpha = 0.15f; // low-pass coefficient (smaller = smoother)
unsigned long lastDisplayMs = 0;

void setup() {
  Serial.begin(9600);

  if (!display.begin(SSD1306_SWITCHCAPVCC, 0x3C)) {
    Serial.println("SSD1306 allocation failed");
    while (1);
  }

  display.clearDisplay();
  display.setTextColor(WHITE);
  display.setTextSize(4);
  display.setCursor(0, 10);
  display.print("TORK");
  display.display();
  delay(500);
  display.clearDisplay();

  // Make sure sensor is at 0 torque when powering up.
  long sum = 0;
  const int CAL_SAMPLES = 200;
  for (int i = 0; i < CAL_SAMPLES; i++) {
    sum += analogRead(ADC_PIN);
    delay(5);
  }
  zeroOffset = sum / CAL_SAMPLES;
  filteredAdc = (float)zeroOffset;
```

```

    Serial.print("Zero offset: ");
    Serial.println(zeroOffset);
}

void loop() {
    long sum = 0;
    for (int i = 0; i < N_SAMPLES; i++) {
        sum += analogRead(ADC_PIN);
        delayMicroseconds(200);
    }
    float adcRaw = sum / (float)N_SAMPLES;

    filteredAdc = filteredAdc + alpha * (adcRaw - filteredAdc);
    float voltage = (filteredAdc / 1023.0f) * 5.0f;

    // Shift around zeroOffset before mapping
    float shifted = filteredAdc - (float)zeroOffset + 512.0f;
    int adcInt = (int)shifted;

    if (adcInt >= 513) torkVal = map(adcInt, 513, 1023, 0, 500);    // 0 .. +500 mN.m
    else torkVal = map(adcInt, 512, 0, 0, -500);    // 0 .. -500 mN.m

    unsigned long now = millis();
    if (now - lastDisplayMs >= DISPLAY_MS) {
        lastDisplayMs = now;

        display.clearDisplay();
        display.setTextColor(WHITE);
        display.setTextSize(1.5);
        display.setCursor(5, 0);
        display.print("Torque:");
        display.setTextSize(2.7);
        display.setCursor(5, 18);
        display.print(torkVal);
        display.print(" mN.m");

        // Voltage
        display.setTextSize(1.5);
        display.setCursor(15, 50);
        display.print(" V: ");
        display.print(voltage, 2);
        display.print(" V");
        display.display();
    }
}

```

APPENDIX-C: 3D DESIGN OF THE MOTOR MOUNTING PLATFORM

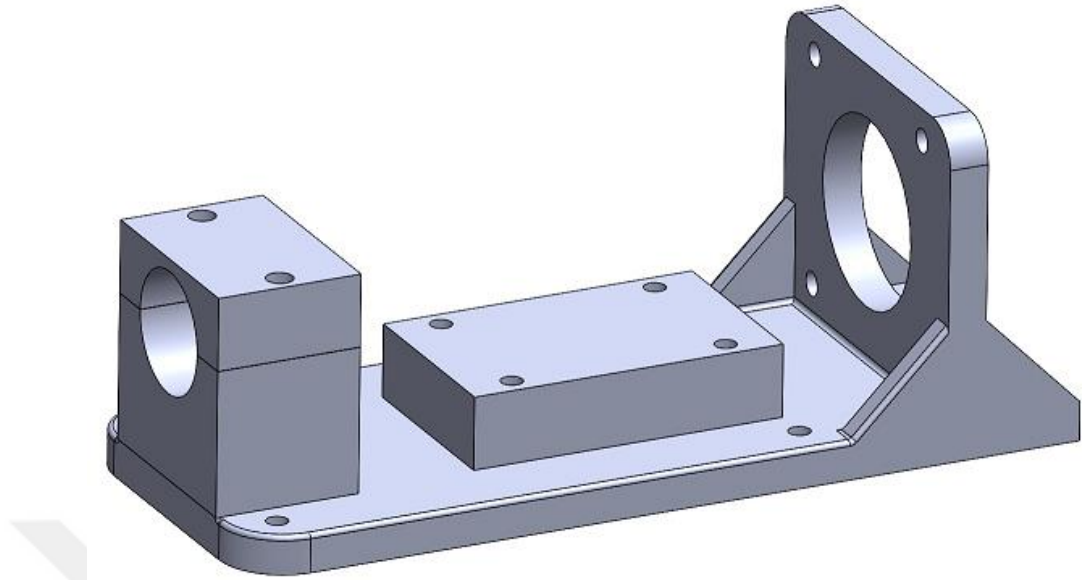


Figure C1: 3D-printed output of the motor mounting platform.

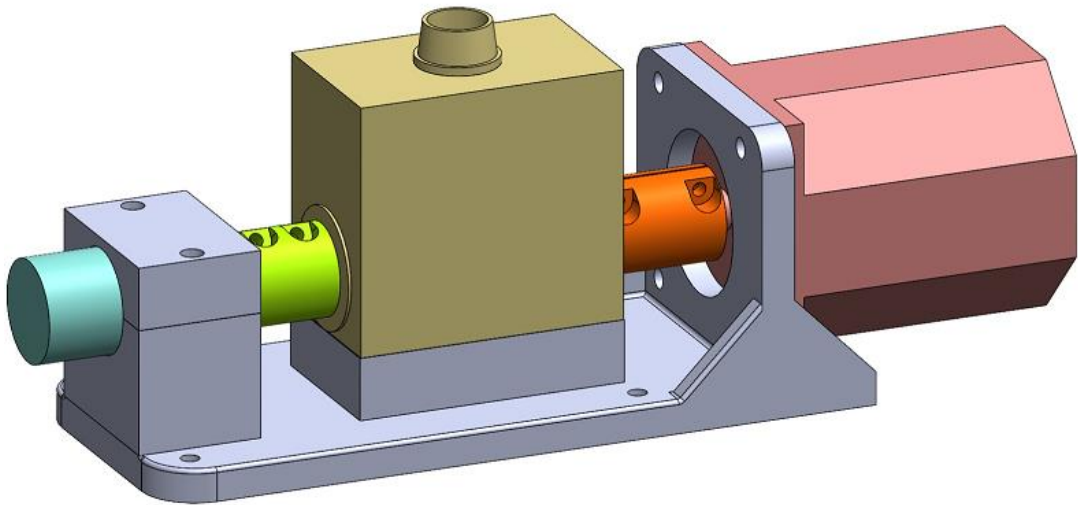


Figure C2: 3D view of the experimental setup.