University of Blida 1

Faculty of Technology
Department of Automation and Electrical Engineering

Doctoral Thesis

in Automatics

CONTRIBUTION TO IMPROVE THE PERFORMANCE OF NEURAL NETWORK MODEL PREDICTIVE CONTROL

Defended by
Chafea STITI

Before the jury composed of:

Abdelaziz Ferdjouni	Pr, U. of Blida 1	Chairman
Abdelkader Maddi	MCA, U. of Blida 1	Examiner
Fares Bouriachi	MCA, USTHB	Examiner
Kamel Kara	Pr, U. of Blida 1	Supervisor
Mohamed Benrabah	MCB, USTHB	Co-supervisor

U. of Blida 1, 25/05/2025

بسم الله الرحمن الرحيم

قَالُوا سُبْحَانَكَ لَا عِلْمَ لَنَا إِلَّا مَا عَلَّمْتَنَا إِنَّكَ
أَنتَ الْعَلِيمُ الْحَكِيمُ

سورة البقرة , الآية 31

Dedication

*To my dear parents "**Rachid**" and "**Fatiha**", those who have guided me, in both shadow and light, who believed in me, even in doubt, who shared my joys and my setbacks, every work, word, and page of this thesis is a tribute to you, for your love, your support, your sacrifices, your encouragements that you have never ceased to give me. This thesis is the fruit of your support, a dream fulfilled, and realized thanks to you. To my dear little sister "**Ryma**" and my dear brother "**Karim**," who have always been there for me.*

*To my thesis supervisor "**Pr. KARA Kamel**", my thesis co-supervisor "**Dr. BENRABAH Mohamed**"and my colleagues as well as to all the people who have closely helped me in this work.*

*To all the brave souls who pursue their quest despite the obstacles. Finally, with self-esteem and determination, I dedicate this work to myself, "**CHAFEA**" Through the challenges and triumphs of this research journey, I have discovered my strength, resilience, degree of patience, perseverance, and limitless potential within me. This work is a tribute to the continuous quest for knowledge and self-improvement. Hoping that this thesis opens the doors to a long journey full of meaning and success in scientific research.*

Acknowledgment

As a preamble, I would like to thank Allah the Most Gracious and the Most Merciful for giving me the courage and will to complete this research challenges and develop this thesis.

This doctoral thesis, conducted at the Electrical Systems and Remote Control Laboratory (LabSET), Department of Automation and Electrical Engineering, Faculty of Technology, University of Blida 1, was made possible by the guidance of Mr. **Kamel KARA**, Professor at the University of Blida 1, whose scholarly insights and unwavering support have been instrumental in shaping this work.

I also dedicate my thanks to my thesis co-director, "**Mr. BENRABAH Mohamed**", Associate Professor at the University of Science and Technology Houari Boumediene, and a member of Electrical Systems and Remote Control Laboratory, for his availability to assist me and answer my questions throughout this thesis. His contributions have been invaluable.

I would also like to thank Doctor "**Mr. AOUICHIA Abdelhadi**" for his invaluable help during all works realized in this thesis, and I would like to thank "**Mr. OUBELAID Adel**", Associate Professor at Bejaia University, for his assistance and advice, which have been extremely helpful to me.

I also express my gratitude to my parents, my sister and my brother for their support and understanding throughout this encouragement, made sacrifices, and provided the necessary balance in my life during this demanding period, and I hope that this thesis will be worthy of our thanks.

I express my gratitude to "**Mr.FERDJOUNI Abdelaziz**", Professor at the University Blida 1, agreed to chair the jury of this thesis. My thanks also go to "**Mr. MADDI Abdelkader**", Associate Professor at the University Blida 1, and "**Mr. BOURIACHI Fares**", Associate Professor at the University of Science and Technology Houari Boumediene, who agreed to be examiners.

Finally, our heartfelt thanks also go to the jury members for agreeing to review our work and enrich it with their suggestions.

Abstract

This doctoral thesis proposes different control approaches to enhance the performance of the neural network model predictive control. For this purpose, a feed forward neural network model is suggested to build the prediction model for the model predictive control. Various metaheuristic algorithms are integrated to solve the nonlinear and non-convex optimization problem of the neural network model predictive control, such as particle swarm optimization, teaching learning-based optimization, and driving training-based optimization algorithms.

Four algorithms are introduced and tested on different systems. In the first algorithm, the teaching learning-based optimization algorithm is used to solve the neural network model predictive control optimization problem, with the implementation of a reduction weighting coefficient in the cost function.This algorithm is applied to control the permanent magnet synchronous motor speed. As for the second algorithm, a driving training-based optimization algorithm is proposed to solve the optimization problem of the neural network model predictive control. This algorithm is used to control the speed of an electric vehicle. Furthermore, a sequence of reduction weighting coefficients is integrated into the cost function of the previous algorithm. This algorithm is applied for the tracking control of a two degrees-of-freedom robot manipulator.

To ensure a closed-loop stability of the controlled system a Lyapunov function is incorporated as a constraint in the cost function of the neural network model predictive control based on the driving training optimization approach. The resulting control algorithm is applied to the speed control of an induction motor, then compared against a set of control approaches including optimized PID based on particle swarm optimization algorithm, fuzzy PID, neural network model predictive control based on different optimization algorithms, such as particle swarm optimization, teaching learning-based optimization algorithm, and driven training-based optimization algorithm.

Résumé

Cette thèse de doctorat introduit différentes approches de commande pour améliorer les performances de la commande prédictive à modèle neuronal. A cette fin, un réseau de neuronestatique est suggéré pour construire le modèle de prédiction de la commande prédictive à modèle neuronal. Divers algorithmes d'optimisation métaheuristique sont intégrés pour résoudre le problème d'optimisation non linéaire et non convexe de la commande prédictive à modèle neuronal, tels que l'optimisation par essaims de particules, l'optimisation basée sur l'apprentissage et l'algorithme d'optimisation basé sur l'entraînement.

Quatre algorithmes sont développés et testés sur différents systèmes. Dans lapremièrestratégie, l'algorithme d'optimisation basé sur l'apprentissage est utilisé pour résoudre le problème d'optimisation de la commande prédictive à modèle neuronal, avec l'introduction d'un coefficient de pondération dans la fonction de coût. Cet algorithme est utilisé pour commander la vitesse d'un moteur synchrone à aimants permanent. En ce qui concerne la deuxième stratégie, un algorithme d'optimisation basé sur l'entraînement est proposé pour résoudre le problème d'optimisation de la commande prédictive à modèle neuronal. Cet algorithme est utilisé pour la commande de la vitesse d'un véhicule électrique. En outre, une séquence de coefficients de pondération est intégrée dans la fonction de coût de l'algorithme précédent. L'algorithme de commande obtenu est appliqué à la commande d'un bras manipulateur à deux degrés de liberté.

Le dernier algorithme est conçu en ajoutant une fonction de Lyapunov comme contrainte dans la fonction de coût de la commande prédictive à modèle neuronal utilisant l'approche d'optimisation basée surl'entraînement.Cette approche permet de garantir la stabilité en boucle fermée du système commandé. L'algorithme développé est appliqué à lacommande de la vitesse d'un moteur à induction, puis comparé à un ensemble d'approches de commande comprenant le PID optimisé basé sur l'algorithme d'optimisation par essaims de particules, le PID flou, la commande prédictive a modèle neuronal basé sur différents algorithmes d'optimisation, tels que l'optimisation par essaims de particules, l'algorithme d'optimisation basé sur l'apprentissage et l'algorithme d'optimisation basé sur l'entraînement.

ملخص

تقدم رسالة الدكتوراه هذه خوارزميات تحكم مختلفة لتحسين أداء التحكم التنبؤي ذو النموذج العصبي. ولتحقيق هذه الغاية، تم اعتماد نموذج الشبكات العصبية متعددة الطبقات كنموذج تنبؤي للتحكم التنبؤي ذو النموذج العصبي. تم دمج عدة خوارزميات التحسين لحل مشكلة التحسين غير الخطي وغير المحدب للتحكم التنبؤي ذو النموذج العصبي، مثل خوارزميات التحسين العملي السربي، والتحسين القائم على التعلم، وخوارزميات التحسين القائم على التدريب

يتم تقديم أربع خوارزميات واختبارها على أنظمة مختلفة. في الخوارزمية الأولى، تُستخدم خوارزمية التحسين القائم على التعلم لحل مشكلة تحسين التحكم التنبؤي ذو النموذج العصبي مع اضافة معامل الترجيح في دالة التكلفة. يتم استعمال هذه الخوارزمية للتحكم في سرعة المحرك المتزامن المغناطيسي الدائم. فيما يتعلق بالخوارزمية الثانية، تُقترح خوارزمية تحسين قائمة على التدريب لحل مشكلة تحسين التحكم التنبؤي ذو النموذج العصبي، مع تطبيقها على التحكم في سرعة سيارة كهربائية. بالإضافة إلى ذلك، يتم دمج سلسلة من معاملات ترجيح في دالة التكلفة لخوارزمية التحكم السابقة. تُطبَّق هذه الخوارزمية على التحكم في رُوبوت ثنائي درجة الحرية.

تم تصميم الخوارزمية الأخيرة عن طريق إضافة دالة Lyapunov كقيد في دالة التكلفة للتحكم التنبؤي ذو النموذج العصبي مع إستعمال خوارزمية تحسين التدريب. تمكن هذه الخوارزمية من ضمان استقرار النظام المتحكم به. يتم تطبيق الخوارزمية المطورة على التحكم في سرعة محرك تحريضي ومقارنتها بمجموعة من خوارزميات التحكم بما في ذلك خوارزمية التحكم PID، خوارزمية التحكم PID الضبابية، خوارزمية التحكم التنبؤي ذو النموذج العصبي القائمة على خوارزميات تحسين مختلفة، مثل خوارزمية تحسين سرب الجسيمات، خوارزمية التحسين القائم على التعلم وخوارزمية التحسين القائم على التدريب.

Contents

List of Figures

List of Tables

List of Algorithms

Nomenclature

Acronyms

ANN	Artificial Neural Network
NN	Neural Network
MPC	Model Predictive Control
NMPC	Nonlinear MPC
NNMPC	Neural Network MPC
AOA	Archimedes Optimization Algorithm
PSO	Particle Swarm Optimization
TLBO	Teaching-Learning-Based Optimization
ADRC	Active Disturbance Rejection Control
DoF	Degrees of Freedom
RMSE	Root Mean Squared Error
MAE	Mean Absolute Error
MSE	Mean Squared Error
NNMPC-AOA	NNMPC based on AOA
NNMPC-PSO	NNMPC based on PSO
NNMPC-TLBO	NNMPC based on TLBO
NNMPC-DTBO	NNMPC based on DTBO
CNNMPC-DTBO	Constrained NNMPC based on DTBO
ENNMPC-TLBO	Enhanced NNMPC based on TLBO
INNMPC-DTBO	Improved NNMPC based on DTBO
LNNMPC-DTBO	Lyapunov based NNMPC using DTBO
FLC-TLBO	Fuzzy Logic Controler based on TLBO

For models formulations

q, q_i	joint angles, ith joint angle
$\dot{q}$, $\ddot{q}$	dot notation indicates first and second order derivatives
J_m	Jacobian matrix
$M(q)$, M_{ij}	mass and inertia matrix, element of line i and column j of $M(q)$
$V(q)$	centrifugal and Coriolis matrix
$B(q)$	Coriolis matrix

$D(q)$	centrifugal matrix
$\dot{q}\dot{q}$	vector of velocity products
$G(q)$, G_i	gravity vector, element i of $G(q)$
$F(q)$, F_i	friction vector, element i of $F(q)$
τ, τ_i	input vector of torques, element i of τ
g	gravitational acceleration
u	control signals
y, y_i	system outputs, ith system output
$f_{NN}(.)$	neural network function
χ	input vector of neural network
σ_1, σ_2	hidden layer activation function, output layer activation function
W_1, W_2	weight matrices
b_1, b_2	bias vectors
x_i	state variable
T_s	sampling time
t	continuous time
k	discrete time index
t_0, t_f	initial time, final time

For controllers formulations

N_1	prediction horizon
N_2	prediction horizon
N_u	control horizon
N_s	constraint horizon
$J(.)$	cost function
Q, R	positive semi-definite matrix, positive definite matrix
Γ_y, $\Gamma_{y_i}(0)$	output-dependent weight matrix, constant minimum value of Γ_y
ref	reference traectories
$\hat{y}(k+i \mid k)$	predicted outputs $k+i$ at time k
Δu	control increment
$e(.)$	output error
k_p, k_i, k_d	PID parameters
λ	weighting factor
R^2	coefficient of determination

For optimization algorithms

c_1, c_2, w, w_d	PSO parameters

Introduction

Industrial systems are becoming increasingly complex, which explains their need for more sophisticated control strategies capable of managing non-linearities, uncertainties, multi variable interactions, and operational constraints. In this regard, Model Predictive Control (MPC) has emerged as a very effective approach, as it uses explicit process models to predict the behavior of the system to be controlled and optimizes the applied control actions, over a finite horizon. However, the performance of traditional model predictive control is highly dependent on the accuracy of the underlying process model. Linear models or simplified non-linear models utilized in MPC techniques like Dynamic Matrix Control (DMC), Model Algorithmic Control (MAC), and Generalized Predictive Control (GPC) are inadequate when working with highly non-linear or dynamically complicated systems, resulting in less-than-ideal control performance.

Neural Network Model Predictive Control (NNMPC) has emerged as a powerful method for addressing these challenges, combining the predictive ability of model predictive control with the nonlinear modeling ability of neural networks. Neural networks, thanks to their ability to approximate complex non-linear functions, have become a more efficient method of modeling complex dynamic systems than traditional linear or simplified non-linear modeling methods. The integration of neural networks into the predictive control framework makes NNMPC an approach with universal approximation capability, learning,adaptability, and generalization to complex systems. Even though NNMPC offers a lot of benefits for controlling nonlinear systems, each control step's success is mostly dependent on resolving a nonlinear optimization problem, in addition, ensuring the stability of the system to be controlled in the presence of model inaccuracies and perturbations remains an essential concern. Real-time implementation of this optimization problem is difficult due to its non-convexity, high-dimensional, and computationally demanding, especially for systems with quick dynamics or demanding performance requirements. Solving this type of optimization problem using conventional deterministic optimization methods such as Sequential Quadratic Programming (SQP) or interior point algorithms proves insufficient, as they are subject to numerous problems such as sensitivity to initial conditions and convergence to a local optimum.

To overcome these limitations, metaheuristic optimization algorithms provide a promising alternative. These algorithms have proven their effectiveness in solving complex, non-convex optimization problems. Particle Swarm Optimization (PSO), Teaching Learning Based Optimization (TLBO), Archimedes Optimization Algorithm (AOA) and Driving Training Based Optimization algorithm (DTBO) are examples of techniques that are natively robust, adaptable, and able to manage constraints and nonlinearities without the need for derivative information of the problem to be optimized. These algorithms offer more dependable and effective solutions to the underlying optimization problem, which can greatly improve the performance of the NNMPC technique.

- **Motivations and objectives:**
  This thesis aims to improve the performance of neural network model predictive control so that it can control various non-linear systems, guaranteeing good control performance, namely compensation of parametric variations, robustness against modeling errors and external disturbances, stability, and fast response time. This has motivated us to develop different NNMPC-type control strategies based on the integration of new variants of metaheuristic optimization algorithms with high convergence speed and precision, the improvement of neural network modeling of the systems to be controlled, and the use of new techniques that can guarantee the closed-loop stability and robustness of the NNMPC under variable operating conditions.The objective of this thesis is to study the performance of the proposed improved NNMPC algorithms through simulations on different reference systems, such as the asynchronous machine, the permanent magnet synchronous motor, the electric vehicle, and the manipulator robot.

- **Manuscript organization:**
  This thesis is organized into four chapters. Chapter 1 presents the state of the arts of model predictive control techniques, including the linear model predictive control, nonlinear model predictive control, and the neural network model predictive control.Chapter 2 introduces the main concepts of the population-based metaheuristic optimization algorithms, and their different application fields. Then it defines the PSO, TLBO, and DTBO algorithms in more detail.Chapter 3 gives the principle of the NNMPC based on metaheuristic optimization approach subject to inputs and outputs constraints. Then it presents the three proposed NNMPC control algorithms, namely, enhanced NNMPC based on TLBO algorithm, DTBO-based constrained NNMPC, and improved NNMPC based on DTBO algorithm. The effectiveness of each proposed control method in terms of accuracy, robustness, and computing time

is approved by a comparative study of the obtained simulation results of each proposed control technique compared with the results of other ones. Chapter 4 introduces a new technique to control constrained nonlinear systems, named Lyapunov-based neural network model predictive control using a metaheuristic optimization approach. This controller utilizes a feedforward neural network model as a prediction model and it employs the DTBO algorithm to resolve the related constrained optimization problem, and the Lyapunov function to ensure the closed-loop stability.

Chapter 1

Model Predictive Control Literature Review

1.1 Introduction

This chapter presents an overview of model predictive control, including a brief review of the used prediction model types, cost functions, and different constraints. It then gives an overview of the linear model predictive control, nonlinear model predictive control, and neural network model predictive control, and it presents the different optimization techniques used in the literature to solve the model predictive control optimization problem. Additionally, this chapter provides a comprehensive literature review, offering context for the developed work presented in the subsequent chapters.

1.2 Model predictive control methodology

Model predictive control, also known as Model-Based Predictive Control (MBPC), or Receding Horizon Control (RHC)originated in the late 1970s [2]. More specifically after the works of Richalet and al in 1978 [3]. In 1979 engineers from Shell (Cutler, Ramaker, Prett, and Gillette) introduced another MPC technique named, Dynamic Matrix Control (DMC) [4]. The MPC control strategy does not specify a unique control approach but refers to a variety of control techniques that utilize a model to predict the system's future outputs [5–8]. This control strategy is based on two parts, the first one consists of obtaining an accurate model of the system to be controlled, and the second part consists of minimizing a performance criterion-usually the quadratic error between the predicted response and the desired reference trajectory over a finite horizon (receding horizon) [9].The MPC technique has been widely adopted by the academic and industrial world in various sectors [10, 11],

such as chemical and petroleum industries [12], robotics [13], etc. The popularity of this type of controller is due to its ability to control different kind of systems, like multivariable or monovariable, with long delay times, unstable or non-minimal phases, and those with simple or complex dynamics [14]. All MPC methods rely on the following points [15]:

- Obtaining the system model to predict its future behavior.
- Calculation of a sequence of future controls in such a way as to minimize a finite-horizon quadratic cost function . Typically, this cost function relates to the error between the reference trajectory, the predicted output.
- Application of the first element of the calculated sequence.
- The previous steps are repeated at the next sampling time, using the receding horizon principle.

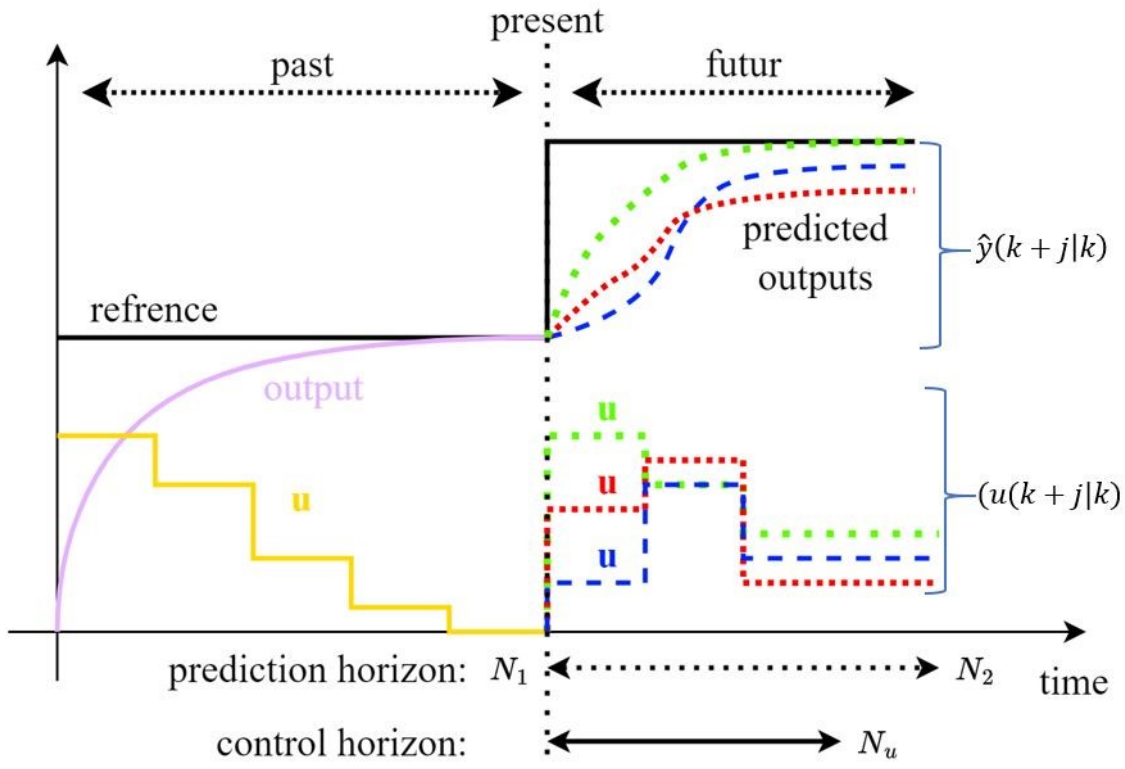

Figure 1.1 – MPC strategy

Figure 1.1 illustrates this approach, where $(\hat{y}(k+j|k), (j = N_1, ..., N_2))$ represent the predicted outputs at time k along a finite prediction horizon $N_p = [N_1, N_2]$. $(u(k+j|k), (j = 0, N_u - 1))$ represent the control signals at the same time k, calculated by optimizing a cost function over a finite control horizon N_u such that $(N_u \leq N_p)$ and $(u(k+j|k) = u(k+N_u-1|k)$ for $N_u \leq j \leq N_p)$.

1.2.1 Prediction models

The first and most important step in MPC designing is to obtain a prediction model. This model is used to predict the system's output over a given prediction horizon. It should accurately represent the behavior of the real system, in particular the relationship between outputs, current measured variables, and the current /future inputs. The performance, accuracy, and viability of MPC all depend on the used prediction model. For this issue, different prediction models were proposed in the literature, such as:

- Linear models including state space models [16], transfer function models, Auto-Regressive with Exogenous Input (ARX) models [17] and Linear Parameter-Varying (LPV) models [18]. These models are generally known for their computational speed, simplicity and stability when modeling linear or approximately linear systems, but unfortunately this is no longer the case for highly non-linear systems or systems with variable dynamics.

- Nonlinear models, such as state-space nonlinear models [19], and Nonlinear ARX (NARX) models.These models give more efficiency in terms of accuracy, but they require complex solvers for optimization, and they tack more computation time.

- Data-driven models: these types, don't depend on specific equations to explain the dynamics of the system. Rather, they use input-output data to infer the behavior of the system. Among these models, we can mention Hammerstein and Wiener models, fuzzy models [20, 21], neural network models [22], and Gaussian Processes models (GPs).These models are efficient for systems whose dynamics are unknown or extremely complex, but they may have significant computational requirements.

- Hybrid models: these models combine the data-driven models with others, like neural ordinary differential equation models [23], which consist of a hybridization between neural networks and ordinary differential equations.

Additional methods for linear and nonlinear modeling are provided in [24]. If the used prediction model has a linear form, the optimization problem is represented by a convex quadratic function with a unique and global minimum, for which a solution could be obtained analytically, particularly if no constraints are imposed. When constraints are present, Quadratic Programming (QP)techniques can be employed. However, when the prediction model is nonlinear, the optimization problem becomes nonlinear and non-convex, which requires numerical optimization methods.

1.2.2 Cost function

Cost function is the most important component in the MPC, it is used to guide the MPC algorithm to minimize the difference between the predicted outputs and a desired reference trajectory over a prediction horizon, ensuring optimal control actions. The cost function can include a number of performance indicators like control effort and tracking error. The cost function is essential for defining control goals, such as, maintaining stability, tracking accuracy, robustness, and enhancing system performance. Several cost functions were proposed for the MPC formulation in the literature, including a quadratic cost function [8]. This type of cost function is the most widely used one due to its convexity, simplicity,and flexibility [25]. The mathematical form of this cost function for single input-single output systems is given as follows:

$$\text{minj}\, J_{u(k)}(\Delta u(k), \hat{y}(k), r(k)) = \sum_{j=N_1}^{N_2} (r(k+j) - \hat{y}(k+j/k))^2 + \sum_{j=1}^{N_u} (\Delta u(k+j-1))^2 \tag{1.1}$$

where $\Delta u(k) = u(k) - u(k-1)$ is the control increment, $\hat{y}(k)$ is the predicted outputs, r is the desired reference trajectories, N_1 and N_2 are respectively the minimum and the maximum prediction horizons generally satisfy this constraint $(1 \leq N_1 \leq N_2)$ and N_u is the control horizon with $(0 \leq N_u \leq N_2)$.

Economic cost function is another type that was introducedin 2012 [25]. The purpose of this type of cost function is not to just follow reference trajectories, but to directly optimize economic performance. This cost function enables the MPC controller to make choices that improve efficiency, coordinating control measures with financial goals.The mathematical form of this cost function is as follows:

$$\text{minj}_{u(k)}(u(k), \hat{y}(k)) = \sum_{j=N_1}^{N_2} \mathcal{L}(u(k+j-1), \hat{y}(k+j/k)) \tag{1.2}$$

where the function $\mathcal{L}(u(k+j-1), \hat{y}(k+j/k))$ symbolizes the system's operational cost or profit at each time step.The review [26] gives the different applications and more details of the MPC using this type of cost function. Weighted cost function, which is initially appeared implicitly in the DMC [4] and the Model Predictive Heuristic Control (MPHC) [3] frameworks.It is given for multiple input-multiple output systems by:

$$\begin{aligned}
&\text{minj}_{u(k)}(\Delta u(k), \hat{y}(k), r(k)) \\
&= \sum_{j=N_1}^{N_2} \left[(r(k+j) - \hat{y}(k+j/k))^T Q(r(k+j) - \hat{y}(k+j/k))\right] \\
&+ \sum_{j=1}^{N_u} \left[(\Delta u(k+j-1))^T R(\Delta u(k+j-1))\right]
\end{aligned} \tag{1.3}$$

where R and Q represent the weights matrices, which, even in early implementations, allowed for adjustments that prioritized accuracy or control effort. In MPC, weighted cost functions are essential because they provide the flexibility to customize control strategies to meet certain goals [27]. Even though their design and tuning have significantly advanced, issues including scaling, trade-offs, and processing requirements still exist. Developments in adaptive, optimization-based, and machine learning-assisted weight selection techniques are continuously pushing the limits of MPC performance. The research paper [28] presents a review of state-of-the-art methods in a weighting factor design of the MPC.

Robust cost function type was firstly integrated into MPC by Campo and Morari in 1987 [29]. Their work laid the foundation for robust MPC by proposing methods to directly handle uncertainties within the cost function given by equation (1.4), ensuring the controller's performance remained effective under a wide range of conditions.

$$\begin{aligned}
J &= \sum_{j=N_1}^{N_2} \left[(r(k+j) - \hat{y}(k+j/k))^T Q(r(k+j) - \hat{y}(k+j/k))\right] \\
&+ \sum_{j=1}^{N_u} \left[(\Delta u(k+j-1))^T R(\Delta u(k+j-1))\right] + \sum_{j=N_1}^{N_2} \left[W w_k(k+j)\right]^2 \\
&+ \Big[\left(r\left(k+j/N_2\right) - \hat{y}\left(k+j/N_2\right)\right)^T P \left(r\left(k+j/N_2\right) - \hat{y}\left(k+j/N_2\right)\right)
\end{aligned} \tag{1.4}$$

where w_k represents the disturbance, W represents weighting matrix that penalizes uncertainty effects, and P is a positive definite weighting matrix.

Terminal cost function type was proposed by the works of D.Q. Mayne and H. Michalska in 1990 [30] and K.R. Muske and J.B. Rawlings in 1993 [31]. This cost function type is calculated using the equation (1.5) given below:

$$\begin{aligned}
&\min j\, J_{u(k)}(\Delta u(k), \hat{y}(k), r(k)) = \\
&\sum_{j=N_1}^{N_2} \left[(r(k+j) - \hat{y}(k+j/k))^T Q(r(k+j) - \hat{y}(k+j/k))\right] \\
&+ \sum_{j=1}^{N_u} \left[(\Delta u(k+j-1))^T R(\Delta u(k+j-1))\right] + V_f\left(y\left(k+j/N_2\right)\right)
\end{aligned} \tag{1.5}$$

where $V_f(y(k+j/N_2))$ is the terminal cost function evaluated at the terminal state.

Several MPC works have been completed since the introduction of the terminal cost function type. These include the works described in [32] and [33], which are designed to ensure the stability and performance of the MPC framework by designing the terminal cost function and terminal regions.

1.2.3 Constraints handling in model predictive control

The MPC is a widely used control method that handles system constraints during the optimization process. It is perfect for applications in complex and safety-critical systems, since it may explicitly take constraints into account at every stage of the prediction horizon. Various researches are conducted to discuss different types of constraints, and how they are handled in the MPC framework. Beginning with the fundamental works, such as those by Bemporad and Morari [34], who suggest handling state, input, and robust constraints for the MPC framework in uncertain environments. Since the publication of these foundational works, several studies have been proposed in this area, including the survey conducted by Schwenzer and Ay in 2021 [25], where they emphasized practical approaches to handling input, output, state, and soft constraints with an emphasis on real-world implementation.

1.3 Linear model predictive control

Based on the prediction model, several techniques of Linear MPC (LMPC), for which a linear prediction model is used, have been developed. A brief review and some applications of well known LMPC strategiesare given in the following.

1.3.1 Model algorithmic control

Model Algorithmic Control (MAC), also called Model Predictive Heuristic Control(MPHC), is the earliest linear predictive control strategy, proposed by Richalet and all in 1978 [3].This control technique has been implemented in several industrial processes, like paper machines [35] and chemical process [8]. The success of the MAC

method for complex multivariable processes can be attributed to two factors: first, it uses an impulse response representation as a prediction model, where the state-space representation is challenging to obtain; second, it uses heuristic algorithms to solve the MAC optimization problem, which makes it appropriate for real-time applications that require fast computations.

1.3.2 Dynamic matrix control

Dynamic Matrix Control (DMC) is one of the well-known MPC techniques that uses the step response as prediction model [2]. In DMC framework, especially in the enhanced DMC framework named QDM [36] a quadratic programming (QP) problem is solved over a control horizon to obtain the optimal control actions. The cost function that is minimized in DMC typically balances tracking performance and control effort, while adhering to system constraints. DMC has been widely adopted in various fields such as chemical and petrochemical industries [37].The extensive use of DMC in these sectors demonstrates how reliable and adaptable it is at handling intricate, multivariable control problems.

1.3.3 Generalized predictive control

Generalized Predictive Control (GPC)is proposed in the late 1980s, is the most popular predictive control technique. It uses a Controlled Autoregressive Integrated Moving Average (CARIMA) [38] model as a prediction model due to its ability to handle disturbances and provide accurate predictions for the system output [38]. The optimal control actions are obtained by optimizing a typically quadratic cost function that strikes a balance between tracking performance and control effort. Since its appearance and due to their robustness and adaptability, the GPC technique has been widely adopted in academic research and various industrial applications such as chemical industries [39] and robotics [40].

1.4 Nonlinear model predictive control

Although, linear MPC techniques give good control performance in many practical applications, in the case of highly nonlinear systems, severe degradations in the control performance can be observed [41]. Therefore, to ensure good control performance, Nonlinear MPC (NMPC) methods that use a nonlinear prediction model should be investigated [42]. In fact, a lot of attention was given to the NMPC techniques, and several control algorithms were proposed. The main difficulties in designing any NMPC algorithm are to obtain an accurate nonlinear model for the system to be controlled and to calculate its optimal control actions by solving the

related nonlinear optimization problem. Actuality, there is not a given modeling strategy that works perfectly for representing nonlinear systems,so that various nonlinear models were used as prediction models, such as:

- Volterra series models, which are introduced by Vito Volterra's work on the class of integral equations used to model non-linear dynamic systems [43]. These models have been very successful in their use as prediction models in the NMPC, due to their ability to capture nonlinear dynamics while remaining computationally feasible, and its capacity to handle memory effects.

- Fuzzy logic models, these models are introduced in the MPC framework by the work of Bin Huang and Jinfeng Lou [21], where they proposed a Fuzzy Model Predictive Control (FMPC) approach for designing control systems to handle complex nonlinearities in control applications.

- Neural network models have been applied as prediction models in the MPC scheme since the works of Andreas Draeger and al,where they proposed a neural network-based MPC technique to regulate PH in a laboratory-scale system [44]. The neural network model becomes a powerful modeling tool for the NMPC design, due to their capability to model complex nonlinearities, adaptability, and reduction of model development time [45].

In addition to the selection of an accurate prediction model, the choice of the optimization technique to solve the nonlinear NMPC optimization problem proves to be very difficult to guarantee an efficient optimal solution, for this, several techniques have been proposed in the literature, such as:

- **Sequential quadratic programming:** Sequential Quadratic Programming (SQP) is an optimization technique that efficiently manages constraints by iteratively solving a sequence of quadratic approximations to nonlinear problems [46]. SQP has several benefits when it comes to managing constraints and nonlinear dynamics in NMPC [47], but it also has drawbacks in terms of implementation complexity, sensitivity to initial circumstances, processing demands, and convergence guarantees.

- **Direct multiple shooting:** Direct Multiple Shooting (DMS) is a numerical optimization technique that divides the time horizon into multiple segments (shooting intervals), where for each segment, the system dynamics are explicitly integrated using numerical methods like Runge-Kutta [48]. This technique is used to solve optimal control problems by enabling the management of complex constraints and concurrent integration of system dynamics. This method enables the efficient computation of control inputs by solving the

NMPC nonlinear optimization problem, which takes into account constraints, system dynamics, and the required performance criteria. Using DMS in NMPC has several benefits, such as better convergence qualities, increased numerical stability, and the capacity to manage intricate constraints. But it also brings with its problems like more complicated computations and the requirement for advanced initialization techniques to guarantee convergence to the optimal solution.

- **Direct collocation:** Direct collocation is a numerical optimization method that discretizes the state and control variables at a set of collocation points over the entire time horizon, where it uses polynomial interpolation like Lagrange polynomials to approximate the system dynamics between these points [49]. This method converts continuous-time optimal control problems into nonlinear programming problems to make the computation of control inputs efficient. This method is used to solve the NMPC optimization problem using piecewise polynomials, due to their efficiency.

- **Variable discretization methods:** Variable discretization modifies time steps or spatial intervals according to system dynamics, error estimates, or particular performance requirements, in contrast to fixed discretization, which keeps these intervals constant. These techniques are used in NMPC to modify the discretization grid during the optimization phase. Numerous research publications have examined the idea of variable discretization techniques in NMPC [50].

- **Neural network-based optimization:** A various methods and frameworks for generating approximate NMPC controllers using Neural Network-Based Optimization (NNBO) have been given in [51]. Based on the guarantees they offer these techniques are divided into three categories [51].

  - Methods without guarantees include an imitation learning technique, where a neural network is trained to mimic NMPC controllers by regression, minimizing error between outputs [51]. The works presented in [52–55] demonstrated the feasibility of this method for NMPC approximation. These techniques are efficient, but they don't guarantee stability or constraint satisfaction, and they frequently call for parallel safety measures.

  - Methods with probabilistic guarantees include statistical learning techniques [56]. These techniquesensure constraints and sub-optimality bounds by satisfying a given probability and providing statistical assurances

on constraint satisfaction and stability, such as the works presented in [57, 58].

- Methods with strict guarantees aimed at preserving theoretical properties of NMPC [56], such as stability and constraint satisfaction, these techniques have been proposed by the works presented in [59], where they employed a trained neural network to approximate the NMPC cost function, while ensuring stability and constraint satisfaction.

- **Metaheuristic optimization algorithms:** Since their appearance, metaheuristic algorithms have been very successful in both industry and academics [60].These algorithms can handle both discrete and continuous optimization problems and generate excellent best solutions in a reasonable time, and they are not impacted by initial conditions due to their heuristic and random features [61]. Because of their capacity to identify nearly optimal solutions in situations where traditional optimization techniques would not be successful, metaheuristic algorithms have become extremely effective tools for resolving the NMPC optimization problem. Among the metaheuristic algorithms that have been applied for the NMPC optimization problem, we cite: Genetic Algorithm (GA) [62], Differential Evolution (DE) [63], Particle Swarm Optimization (PSO) [64], Artificial Bee Colony (ABC) [65], Teaching Learning Based Optimization Algorithm (TLBO) [66], Archimedes optimization algorithm (AOA) [67]and Driving Training Based Optimization Algorithm (DTBO) [68]. Despite their effectiveness, metaheuristic algorithms are not flawless methods of optimization. Indeed, some disadvantages could be listed, such as the need to adjust a number of the algorithm's parameters before using it. The efficacy of this type of algorithm is greatly influenced by its parameters, the ideal values of which are not always known. Although metaheuristic-based algorithms can typically produce excellent results in a fair time, neither this nor the fact that the produced solutions are close to the global optimum are guaranteed by them [61].

  To enhance the control performances of the NMPC framework, different strategies have been proposed based on the various prediction models and optimization technique cited above, one of these techniques which is more recent and has been very successful is Neural Network based MPC (NNMPC) [69].

1.5 Neural network model predictive control

By the late 1980s and early 1990s, Narendra and Parthasarathy (1989) utilized Feedforward Neural Networks (FNNs) to approximate the nonlinear functions gov-

erning the system dynamics [70]. They demonstrated that Neural Networks (NNs) could learn the input-output relationships of complex systems through training on historical data [71]. Their work showed that (NNs) could effectively capture the behavior of nonlinear systems and handle their complexities, providing a foundation for using NNs as a viable tool for system identification and control applications. Their contributions influenced a wide range of applications and spurred further developments in NN-based control strategies [72]. The 2000s saw a surge in interest as computational power increased. During this period, NNs were increasingly applied to chemical process control, where they helped manage complex, nonlinear dynamics more effectively than traditional methods [73]. Several neural network architectures have been used as MPC prediction models in the literature, such as Multilayer Perceptron (MLP) [74], Convolutional Neural Networks (CNN) [75], Recurrent Neural Networks (RNN) [76], Long Short-Term Memory Networks (LSTM) [77], and Gated Recurrent Units (GRU) [77].To obtain such models, a training step of the neural network weights is required so several training algorithms have been proposed, such as Quasi-Newton methods Levenberg-Marquardt algorithm (LM),and Back propagation algorithm [78].

The use of neural networks in NMPC has evolved from basic implementations to sophisticated, real-time control systems. Ongoing research continues to push the boundaries, focusing on enhancing robustness, efficiency, and applicability to a wider range of complex, nonlinear systems, like robotics [79], chemical processes [80], power electronics [81] and smart grid [82]. For this purpose, many NN based NMPC approaches have been proposed, such as:

- **Model-based neural network nonlinear model predictive control:** This control technique uses neural networks as a prediction model in the NMPC framework, to approximate the system's nonlinear dynamics based on historical input-output data. In this method the neural network learns the system behavior and provides state predictions over the control horizon, instead of relying on first principles or physics-based models. The trained neural network is then integrated in the NMPC to enhance its performances, by improving model accuracy for complex systems where traditional modeling is difficult.This method has made the NMPC more applicable to real-world nonlinear systems, such as robotics [67], and energy Systems [83].

- **Model-free neural network nonlinear model predictive control:** This method is directly learned by the neural network without the need for an explicit system dynamics model [84,85]. Usually, methods from Reinforcement Learning (RL) are used to do this. Through interaction with the environment, RL algorithms allow the neural network to learn optimal control rules. Re-

wards and punishments are sent to the network as input, directing its learning process. When it is difficult to derive an explicit model for a system, this method may handle extremely complex and nonlinear systems. The neural network in this type approach is directly maps states to control actions. With time, it gains the ability to maximize cumulative rewards. Model-free neural network propagation is used in many fields, such as robotics, where it is used for tasks like navigation, manipulation, and interaction with dynamic environments. The neural network learns control policies that enable the robot to perform complex tasks without explicit modeling of its dynamics [86]. Another application of this control approach is energy management, where it is used to optimize energy distribution and storage of smart grids and energy storage systems where the neural network learns to manage energy flows efficiently [87], adapting to varying demand and supply conditions. Finally, autonomous vehicles use this control technique for real-time decision-making and control, and the neural network learns driving policies that ensure safe and efficient navigation in a variety of traffic scenarios.

- **Hybrid neural network nonlinear model predictive control:** This method integrates neural networks with physical models [88]. The physical model handles the well-understood dynamics of the system, while the neural network represents its complicated and nonlinear aspects. By merging the outputs from the physical model with the neural network, the hybrid model predicts future states. The NMPC then uses these predictions to optimize control inputs, balancing the advantages of both models to produce reliable and effective control. This method is integrated in many applications like Hybrid Electric Vehicles (HEVs) where it is used for optimal power management to minimize pollution and maximize fuel efficiency, considering that the physical model takes care of the well-understood engine dynamics, while the neural network models the nonlinear battery dynamics. In chemical processes, this control approach is also utilized [89, 90]. Here, physical models of the process equipment are combined with neural networks to capture these nonlinear interactions, guaranteeing exact control and optimization. This control technique is used by smart grids to regulate the distribution of energy, with the physical model representing the grid infrastructure and the neural network modeling nonlinear demand patterns to optimize energy flow and stability.

  The nonlinear and nonconvex optimization problem of the model-based Neural Network Nonlinear Model Predictive Control (NNMPC)cannot be solved using the same techniques with nonlinear model predictive control, like SQP, interior point methods [91], and Newton-type methods [92].Due to the disadvantages

of these classical optimization techniques, such as their sensitivity to initial conditions, and their difficulty in handling no convexity issues are difficult, their need for the gradient information for complex models , their problems with local minima and convergence, their problems with scalability where the performance decrease when the optimization problem size increases, and their lack of flexibility [93]. Because of these drawbacks, academics have turned their attention to metaheuristic algorithms, which have several benefits including the capacity to explore the entire space, which makes them useful for locating global optima in nonlinear, non convex environments, the capability to solve a variety of problems like those with discontinuous or non-differentiable cost functions, because they do not require gradient information, the robustness against local minima, the usefulness for real-time applications due to its simple implementation and parallelize, the capacity to offer improved accuracy, speed, and robustness in control performance due to its efficiency in the systems with complicated constraints and fast dynamics which make it well-suited for real-time applications in NNMPC [66–68], [94].

1.6 Conclusion

This chapter presented an overview of linear, nonlinear, and neural network MPC, and it mentioned the different optimization techniques and prediction models used in each MPC type. It highlighted the challenges and limitations of each MPC method, emphasizing the importance of neural network MPC. It introduced the advantages of metaheuristic optimization algorithms to solve the NNMPC optimization problem, neither than the conventional optimization techniques. The chapter provided the importance of this type of approach for the control of complex nonlinear systems. In addition, it has served as literature review, providing readers with a broad understanding of the current state of the field.

Chapter 2

Fundamentals of Metaheuristic Optimization Algorithms

2.1 Introduction

This chapter introduces the key concepts behind population-based meta-heuristic optimization algorithms and how the corresponding optimization problem is formulated. The definition and the principles of the particle swarm optimization algorithm, the teaching learning based optimization algorithm and the driving training based optimization algorithm are given. In the last section, some applications of metaheuristics algorithms in different fields are introduced.

2.2 Basic concepts of metaheuristic optimization algorithms

2.2.1 Origin and definition

In the real world, the existence of many complicated and hybrid problems such as the transport, storage and location of product sales, the design of communication networks, scheduling, planning, etc, requires techniques for optimally solving these problems within a reasonable and acceptable time, which cannot be guaranteed by exact optimization techniques. To remedy this problem Polya introduced new optimization methods in 1945, called heuristic methods [95], known for their effectiveness in finding good solutions at a reasonable computational cost, as confirmed by Caserta and Voß [96]. Several works and variants of this type of algorithm have been proposed [97]. Despite their successes, heuristic algorithms have several drawbacks, such as the risk of falling into local optimal traps without being able to escape from this situation, and the inability to solve real problems of high complexity in a given

computation time. This is why Fred Glover in the 1980s introduced a new type of algorithm that is an improvement on heuristic algorithms, called 'metaheuristic' [98]. This name is a combination of two words, 'meta' meaning a higher-level methodology, and 'heuristic' meaning finding a goal by trial and error. From this description, metaheuristics can be defined as higher-level heuristic techniques whose objective is the efficient search for near-optimal solutions by exploring and exploiting search spaces through the use of learning strategies for structuring information. Since their appearance, metaheuristic algorithms have been increasingly used to solve hard problems in several industries and professions, including administration, planning, architecture, engineering, healthcare, and logistics. Thanks to their ability to provide a good solution to more complicated optimization problems with incomplete information or limited computing capacity, unlike traditional optimization techniques that linearize the cost function or use derivatives and gradients.

Metaheuristics can frequently provide a superior trade-off between computing time and solution quality, particularly for complex optimization problems [60]. Metaheuristic algorithms are adaptable to various real-life optimization problems, balancing solution quality and computation time without imposing constraints on problem formulation. However, in order to attain good performance, this flexibility comes at the cost of requiring significant problem-specific customization. Most metaheuristic algorithms are characterized by the following properties:

- They are strategies that guide the search process.
- They have a stochastic behaviour.
- Their objective is to efficiently explore the search space in order to find solutions close to the optimum.
- They range from simple local search procedures to complex learning processes.
- They are approximate and generally non-deterministic.
- They are not problem-specific.

2.2.2 Classification

Several classifications have been proposed for metaheuristic algorithms according to several criteria, namely those cited below:

2.2.2.1 Nature-inspired or non-nature-inspired

Based on their sources of inspiration, metaheuristic algorithms can be roughly divided into two groups.The first one is nature-inspired algorithms. They draw

inspiration from natural events and processes and fall under several subcategories, which include:

- Evolutionary algorithms: where the process of natural evolution serves as inspiration, like GA [62].
- Swarm intelligence algorithms: where the collective behavior of decentralized systems serves as the foundation for these algorithms, like PSO [64].
- Physics-based algorithms: where physical principles and occurrences are the source of these algorithms, such as Simulated Annealing algorithm(SA) [99].
- Biologically inspired algorithms: which are the ones who mimic the way biological systems work, like the Artificial Immune Systems Algorithm (AISA) [100].
- Human-related algorithms: The human-based metaheuristic algorithms are motivated by social interaction or human behavioral patterns. For example: the Brainstorm Optimization (BSO) algorithm [101], which was used for data classification, works similarly to how individuals produce ideas. It uses a collaborative search approach to repeatedly create, evaluate, and refine viable solutions to solve optimization problems. Teaching learning-based optimization [66], takes into account the teacher's impact on the students in the class. To explore the search area and find the best solutions, it combines concepts from both teachers and students. Gaining Sharing Knowledge-Based Algorithm [102](GSKA)which leverages human knowledge acquisition and sharing to address optimization problems. To enhance the search process and enable the algorithm to effectively explore the search space and determine optimal solutions, it promotes collaboration and information sharing. These types of algorithms are based on the notion that people should share their knowledge and learn from one another.

The second group of metaheuristic algorithms is the non-nature-inspired one [103]. This type is derived from other heuristic techniques, such as:

- Tabu Search (TS) [104]: It avoids going over previously investigated solutions by using memory structures.
- Greedy Randomized Adaptive Search Procedure (GRASP) [105]:It combines randomization with greedy algorithms.
- Variable Neighborhood Search (VNS) [106]: It systematically modifies the search process neighborhood structure.

2.2.2.2 Memory consumption or memoryless

Another way to categorize metaheuristic algorithms is by memory usage, so there are memory-based and memoryless algorithms [107]. Algorithms of the first category use memory to retain information about the search history, like the past solutions founded. Various algorithms belonged to this category, such as TS algorithm, GA, and Ant Colony Optimization (ACO) algorithm [108].

Algorithms of the second one, which are memoryless algorithms, just use information about the current state to make decisions and they do not keep track of information about previous search states, such as SA algorithm, PSO and Random Search (RS) algorithm [109].

2.2.2.3 Stochastic or deterministic

It is possible to categorize metaheuristic algorithms according on whether they use determinism or stochasticity in their search process [107]. Stochastic algorithms, like GA, SA, PSO and ACO algorithms explore the search space using probabilistic judgments and random variables. Exploring different areas of the solution space and avoiding local optima are made easier by the randomization.

Deterministic algorithms employ a predetermined set of guidelines or heuristics to explore the search space, rather than depending on random processes, as it is the case for TS, deterministic crowding algorithm [110], and greedy algorithms [111].

2.2.2.4 Greedy or iterative

Greedy algorithms select the locally optimal option at each stage, in an attempt to reach a global optimum [111]. Although they are usually fast, they might not always come up with the right answer for challenging issues. Iterative algorithms continuously improve a collection of solutions, over several rounds. To progressively raise the caliber of the solution, they frequently entail an investigation and exploitation phase.

2.2.2.5 Local or global search

Based on their search tactics, metaheuristic algorithms can be divided into two categories: local search algorithms and global search algorithms [112]. The first category concentrates on investigating the area around the existing solution to find a better solution in a local area of the search space, one of the most cited algorithms of this category are TS, SA, and Hill Climbing (HC) algorithms [113].

Algorithms of the second category are global search algorithms, like GA, PSO and ACO. This category of algorithms seeks to more thoroughly investigate the

entire space, frequently employing several potential solutions to prevent becoming stuck in local optima.

2.2.2.6 Population-based or single-solution-based search

In order to resolve the optimization problem, some algorithms really employ a population of candidate solutions, while others use a single candidate solution [107].In population-based algorithms, a population of solutions is simultaneously maintained and evolved. GA, PSO and ACO are ones of this type of algorithms, which are often better at avoiding local optima and go further into the search space.

While single-solution-based algorithms, like TS and SA algorithms concentrate on iteratively refining a single candidate solution. They may be more likely to become trapped in local optima, although they are usually simpler and use less computing power.

2.2.2.7 Mimetic algorithms or hybridization

Mimetic algorithms are based on one optimization technique to solve its optimization problems, while hybrid algorithms are based on the hybridization of two or several optimization techniques to solve a given optimization problem. The creation of new hybrid algorithms based on the combination of the best qualities of different algorithms is growing all the time because they get rid of the trap of local optima, avoid premature convergence and efficiently explore the search space, offering a better compromise between the exploration and exploitation quality. Some exemples of hybride algorithms are: Hybrid Whale and Simulated Annealing (WOA-SA) [114], Hybrid Ant Colony Optimization and Particle Swarm Optimization (ACO-PSO) [115], and Hybrid Particle Swarm Optimization and TLBO (HPSO-TLBO) [116].

2.3 Population based metaheuristic algorithms principle and concepts

The algorithms of this family are typically founded on the same idea. They often begin with a population of initial solutions that are randomly generated, and then use certain search operators on the current population in an iterative attempt to enhance these solutions. Using a replacement strategy, some or all of the existing population is swapped out for the newly created population in order to produce a better population of solutions. This process keeps going until a stopping criterion is met.

The steps listed below provide a summary of these algorithms' basic idea.

The most commonly used technical words for this type of algorithm are as follows:

Algorithm 1 Pseudo code of population-based metaheuristic optimization algorithms

1: **STEP 0**: Initialization.
2: Set the initial population
3: **STEP 1**: Generation/Replacement
4: **for** counter=1 to maximum number of iterations **do**
5: Generate the new population.
6: Update some or all of the old population using the new generated one.
7: If the stopping criterion is satisfied, go to step 2
8: **end for**
9: **STEP 2**: output the best solution.

2.3.1 Initial population

Initial population is the first set of candidate solutions generated at the start of the optimization process. This process requires particular attention, whatever the meta-heuristic technique employed, as it can significantly influence the performance and convergence speed of the used optimization algorithm used. The choice of the initial population for any meta-heuristic algorithm is responsible for preparing the ground for the search for optimal solutions by that algorithm. If the search space is poorly covered, the optimization procedure may converge to a local optimum or take a long time to find the right answer, which can reduce the efficiency of that algorithm.

The various initialization approaches founded in the literature were divided into several categories [117], including random, quasi-random, chaos theory, Levy Flight, ad hoc knowledge of the domain, probability distributions, pseudo-random number or Monte Carlo methods, greedy, hybrid, and others methods.

2.3.2 Population size

Population size is the number of possible solutions (individuals) that are maintained and modified throughout the optimization process. The performance of the optimization algorithm can be strongly influenced by the population size.The solution space is more thoroughly explored by large populations, which can result in good solutions, but also requires more processing power. For less diversity, small populations risk being caught in local optima, even though the converge is fast. It should be noted that there is no absolute rule for determining the size of the population.

2.3.3 Exploration and exploitation

Exploration and exploitation are two essential concepts in population based metaheuristic algorithms, which enable the search process to be balanced in order to find optimal solutions. Exploration refers to the algorithm's ability to explore different areas of the analysis space. Its objective is to find new and possibly improved solutions by organizing the search effort. The aim is to avoid the best local solutions and ensure a thorough search of the entire solution space. Exploitation focuses on in-depth research into the best performing solutions in order to perfect and improve them. The aim is to get closer to the optimum solution by making small, incremental improvements in promising areas. The balance between exploration and exploitation is essential to the success of metaheuristic algorithms. Excessive computational effort caused by poor exploitation can lead to premature convergence towards sub-optimal solutions, while excessive exploration can lead to premature convergence towards sub-optimal solutions.It is essential to find an appropriate balance between exploration and exploitation in order to ensure the efficiency of the optimization algorithm. To this end, several methods have been proposed, namely adaptive techniques where the metaheuristic algorithms adjust their parameters dynamically to move from exploration to exploitation [118], hybrid approaches where several metaheuristic algorithms are combined for better exploration and exploitation and Diversity Metric (DM) methods [119].

2.3.4 Stopping criteria

The stopping criteria, in metaheuristic algorithms are the constraints that establish when the algorithm should stop. This requirement is necessary to balance the trade-off between computing effort and solution quality and to guarantee that the algorithm does not run endlessly.The criteria found in the literature can be divided into two classes:

- Static procedure: with this approach, the end of the optimization process is known in advance. A maximum number of cost function evaluations, a limit on computation resources, or a predetermined number of iterations could be used to do this. When there is a time constraint, this process is typically employed.
- Adaptive procedure: With this approach, it is impossible to predict the end of the optimization process in advance. This might be used when a satisfactory or optimal solution is found with a predetermined error tolerance, or it could be applied with a set number of non-improving repeats. There are additional steps that are unique to population-based metaheuristic algorithms. They typically

rely on certain population numbers or its evolution. For instance, in specific situations, the algorithm will halt if the population's diversity falls below a predetermined threshold, at which point it will be deemed stationary. When this occurs, there is no use in carrying on with the optimization procedure.

2.4 Solving an optimization problem using metaheuristic algorithms

Any optimization issue, in this instance minimization, has the following standard form:

$$\min_{X} f(X) \tag{2.1}$$

Subject to:

$$G(X) = 0 \tag{2.2}$$

$$H(X) \leq 0 \tag{2.3}$$

where the variables to be optimized are denoted by $X \in \mathbb{R}^n$, the number of optimized variables is denoted by n, the cost function is denoted by $f(X)$, and the constraints functions are denoted by $G(X)$ and $H(X)$.

The use of population-based metaheuristic algorithms for solving the minimization problem defined above can be given by the following standard algorithm:

Algorithm 2 Pseudo code of population-based metaheuristic optimization algorithms

STEP 0: Initialization.
$i = 0$
for $j = 1$ to Population size **do**
    Choose the initial solution from the search space.
end for
STEP 1: Evaluation.
for $j = 1$ to Population size **do**
    Evaluate the cost function given by equation (2.1) according to the candidate solution.
end for
STEP 2: Generation/Replacement.
for $j = 1$ to Population size **do**
    Generate a new candidate solution.
    Apply an updating strategy.
    if the replacement is necessary **then**
        The old solution=New candidate solution.
    end if
end for
STEP 3: Searching the best solution.
The best solution= The candidate solution for $j = 1$.
for $j = 2$ to Population size **do**
    if f(candidate solution)$<f$(best solution) **then**
        The best solution= The candidate solution.
    end if
end for

STEP 4: Stopping the optimization process.
if stopping criteria is satisfied **then**
    Report the best solutionas the solution.
    Exit the optimization process.
else
    $i = i + 1$
    Go back to step 1.
end if

The creation of the subset S of admissible solutions to the problem arising from the constraints defined above is not always easy in reality. This is why the method of managing these constraints is of great importance during the optimization process in order to guarantee the efficiency of the used algorithm. Several constraint handling techniques have been proposed, namely:

- Penalty methods: In these techniques, a penalty term is added to the cost function to transform the constrained problem into an unconstrained one.

The penalty term is determined by the degree of constraint function violation and the penalty factors, whose values can change as the optimization process progresses. Kawachi [120] proposes an adaptive method for determining the penalty factor throughout the evolution process. This method successfully addresses the problems of over or under penalization, which may cause the search process to deviate from the optimal one.

- Feasibility rules: Since improper values can cause the search process to deviate from the region of the optimal solution, it is necessary to test various penalty factor values to determine the best alternative. The reference [121] presents a novel method based on a tournament selection operator to compare two solutions based on the following criteria: any feasible solution is preferred over any infeasible solution; the solution with the best cost function value is preferred among two feasible solutions; and the solution with the small constraint violation value is preferred among two infeasible solutions.

- ϵ-constrained method: In order to provide comparison rules on limited optimization problems during the evolutionary process, Taka Hama [122] proposed a modified version of the feasibility rules technique. This method defines the overall constraint violation function as the sum of all constraints or as the maximum of all constraints.

- Stochastic ranking: The stochastic ranking method introduced by Runarsson and Yao [123] is a commonly used constraint handling strategy that avoids the drawbacks of penalty systems. In order to rank all nearby individuals in a population from best to worst, stochastic ranking compares them based on penalty and cost function dominance.

2.5 Used metaheuristic algorithms

To design efficient algorithms for different optimization problems, several metaheuristics algorithms have been proposed in the literature. For the work done in this thesis, the following algorithms are used:

2.5.1 Particle swarm optimization algorithm

The PSO principle, relies on a population of particles forming a swarm. Each particle in the swarm flies into the search zone to find the best solution (optimum) for landing, such that the group of flying particles represents changing solutions and the search zone the set of possible solutions. Each particle is defined by its position and flight speed (velocity), which changes throughout the iterations (generation)

thanks to its flight experience (its personal best solution) and that of its swarm (its swarm's best solution).The particle swarm continues its search until it reaches a promising zone that will enable it to reach the global optimum, which will be the solution to an optimization problem.

The PSO steps can be summarized as follows:

- **Step 0:** Population size (N_p), particles number (N_v), number of optimization steps (N_s), bounds of particles position ($X_{i_{min}}, X_{i_{max}}$) and stopping criteria are defined.Set the optimization step s=1 , and initialize randomly the position and velocity of each particle i, which are represented by X_i and V_i, respectively.
- **Step 1:** For each particle, evaluate the cost function (f) and determine the initial global best (g_{best}) and initial personal best (p_{best}).
- **Step 2:** Update the p_{best} and g_{best} according to the following conditions:

  - If $f\left(X_i(s+1)\right) < f\left(p_{\text{best },i}(s)\right)$, set $\quad p_{\text{best },i}(s+1) = X_i(s+1)$ (2.4)

  - If $f\left(X_i(s+1)\right) < f\left(g_{\text{best}}\right)$, set$g_{\text{best}} = X_i(s+1)$ (2.5)

- **Step 3:** update the velocity and position of each particle using the following equations:

  - $V_i(s+1) = c_1 r_1 \left[p_{\text{best },i}(s) - X_i(s)\right] + wV_i(s)$ (2.6)

  - $X_i(s+1) = X_i(s) + V_i(s)$ (2.7)

  where w is the inertia weight, c_1 and c_2 are acceleration coefficients, r_1 and r_2 are random numbers between 0 and 1.

- **Step 4:** Continue iterating through steps 1–3 until the stopping condition is met or the number of optimization steps is attained.

2.5.2 Teaching learning based optimization algorithm

Teaching learning based optimization algorithm is a metaheuristic algorithm inspired by the teaching and learning process that takes place in a classroom [66]. This algorithm is based on the principle of finding a global solution from a population of solutions. The population considered being a set of students and teachers, and

the search process takes place in two stages: the instructor phase and the student phase. In the first phase, students learn from their teacher, while in the second, learners learn through self-interaction. For the TLBO algorithm, the fitness value of the optimization problem is considered as the learners' result, the variables of the cost function represent the subjects taught, and the best solution is defined as the teacher. The following steps describe the process of the TLBO algorithm:

1. Ahead the TLBO algorithm runs, the initialization step is necessary to set the following parameters: The size of the population (N_p), the number of variables to be optimized (N_v), the bounds of each variables ($X_{i_{min}}, X_{i_{max}}$) where ($i = 1, 2, ..., N_v$), the maximum number of iterations (N_s). The initial solution $X^1_{i_k}$ with ($k = 1, 2, ..., N_p$), is generated according to the equation (2.8) defined below:

$$X^1_{i_k} = \text{rand}(0,1)\left(X_{i_{\max}} - X_{i_{\min}}\right) + X_{i_{\min}}, i = 1, 2, \ldots, N_v, \quad k = 1, 2, \ldots, N_p \tag{2.8}$$

2. Use the cost function to evaluate the TLBO population and select the elements with the best results to be the instructor $X^1_{i_{kbest}}$ for each optimized variable, with $k = 1, 2, ..., N_p$.

3. Calculate the average score $M_{i,j}$ for each optimized variables according to equation (2.9).

$$\mathcal{M}_{i,j} = \frac{\sum_{b=1}^{N_p} X^j_{i_b}}{N_p}, i = 1, 2, \ldots, N_v, \quad j = 1, 2, \ldots, N_s \tag{2.9}$$

4. Determine the distance $D^j_{i_k}$ among the average score of the students and the teacher for each optimized variable, using equation (2.10).

$$D^j_{i_k} = r_j\left(X^j_{i_{kbest}} - T_f \mathcal{M}_{i,j}\right), i = 1, 2, \ldots, N_v, j = 1, 2, \ldots, N_s, \quad k = 1, 2, \ldots, N_p \tag{2.10}$$

   where r_j is a random number $\in [01]$, and $T_f = round(1 + rand(0,1))$.

5. Update the previous solutions according to equation (2.11).

$$X^j_{n_n w_i k} = X^j_{ik} + D^j_{ik}, i = 1, 2, \ldots, N_v, j = 1, 2, \ldots, N_s, \quad k = 1, 2, \ldots, N_p \tag{2.11}$$

6. To retain the best solution, use greedy selection between the recently updated $X^j_{newi_k}$ solution and the previous one (X^j_{ik}).
7. Select randomly a pair of solution (X_α) and (X_β), and calculate their cost functions F_α and F_β.
8. Use the equations (2.12) and (2.13) to update the solutions for each pair.

$$Xnew^j_{i\alpha} = X^j_{i\alpha} + r_j \left(X^j_{i\alpha} - X^j_{i\beta}\right) \text{ if } F^j_\alpha < F^j_\beta \quad , i = 1, 2, \ldots, N_v, j = 1, 2, \ldots, N_s \tag{2.12}$$

$$\text{Xnew }{}^j_{i\beta} = X^j_{i\alpha} + r_j \left(X^j_{i\beta} - X^j_{i\alpha}\right) \text{ if } F^j_\beta < F^j_\alpha, i = 1, 2, \ldots, N_v, j = 1, 2, \ldots, N_s \tag{2.13}$$

9. Evaluate the cost function for each improved solution. Only accept the improved solution.
10. Verify if the limit of the stop criteria is reached and keep the best solution if it is yes or go back to the step (3) if it is not.

2.5.3 Driving training based optimization algorithm

Driving training based optimization is a recent metaheuristic algorithm, inspired by the human activities of driver training in driving schools [68]. It is based on a population of learners and instructors that are considered as candidate solutions. They are updated throughout the optimization process until the best solution is obtained, which ensures the best value of the cost function.

This search process can be modeled into three phases: phase1, also known as the training by the driving instructor phase, is used to select the instructors who will teach the learners-drivers how drive. Phase 2 is referred to as learner driver patterning from instructor skills phase, during which the student drivers replicate the instructor's gestures and maneuvers. Phase3, which is the personal practice phase, is designed to enhance each learner driver's progress and improve his driving skills through individual practice. To implement the DTBO algorithm we need to follow the following phases:

1. **Initialization phase:**
   - Define the initial population as candidate solutions to the optimization problem modelled using a matrix called population matrix given by equation (2.14).

$$X = \begin{bmatrix} X_1 \\ \vdots \\ X_i \\ \vdots \\ X_{N_p} \end{bmatrix}_{N_p * N_v} = \begin{bmatrix} x_{11} & \cdots & x_{1j} & \cdots & x_{1N_v} \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ x_{i1} & \cdots & x_{ij} & \cdots & x_{iN_v} \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ x_{N_p1} & \cdots & x_{N_pj} & \cdots & x_{N_pN_v} \end{bmatrix}_{N_{p*Nv}} \tag{2.14}$$

- Initialize randomly the position of the initial population using equation (2.15).

$$x_{i,j} = \mathrm{x}_{\min j} + r.\left(\mathrm{x}_{\max j} - \mathrm{x}_{\min j}\right) \tag{2.15}$$

  where X is the population of DTBO, X_i is the ith candidate solution, $x_(i,j)$ is the value of the jth variable determined by the ith candidate solution, N_p is the size of the DTBO population, N_v is the number of optimization problem variables, $r = random(0,1)$, x_{maxj} and x_{minj} are the lower and upper bounds of the jth variable, respectively.

- Use equation (2.16) to evaluate the cost function of each candidate solution for each variables.

$$F = \begin{bmatrix} F_1 \\ \vdots \\ F_i \\ \vdots \\ F_{N_p} \end{bmatrix}_{N_{p*1}} = \begin{bmatrix} F\left(X_1\right) \\ \vdots \\ F\left(X_i\right) \\ \vdots \\ F\left(X_{N_p}\right) \end{bmatrix}_{N_{p*1}} \tag{2.16}$$

  where F is the vector of cost functions and F_i is the value of the cost function provided by the candidate solution.

- Chose the member that has the best value for the cost function as the best solution (X_{best}).

- End initialization phase.

2. **Training phase(P_1):**

- Determine the driving instructor matrix DI using equation (2.17).

$$DI = \begin{bmatrix} DI_1 \\ \vdots \\ DI_i \\ \vdots \\ DI_{N_{DI}} \end{bmatrix}_{N_{DI}*N_v} = \begin{bmatrix} DI_{11} & \cdots & DI_{1j} & \cdots & DI_{1N_v} \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ DI_{i1} & \therefore & DI_{ij} & \therefore & DI_{iN_v} \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ DI_{N_{DI}1} & \cdots & DI_{N_{DI}j} & \cdots & DI_{N_{DI}N_v} \end{bmatrix}_{N_{DI}*N_v}$$
$$i = 1, 2, \ldots, N_p, j = 1, 2, \ldots, N_v \tag{2.17}$$

where DI_i is the driving instructor,DI_{ij} is the jth dimension,and $N_{DI} = 0.1N_p - (1 - z/N_s)$ is the number of driving instructors, and z is the current iteration.

- Randomly select the driving instructors from the matrix DI.

- Calculate the new position for each member using equation (2.18), and update it according to the equation (2.19).

$$x_{i,j}^{P1} = \begin{cases} x_{i,j} + r.\left(DI_{k_i,j} - I.x_{i,j}\right), & \mathrm{F}_{DI_{k_i}} < F_I \\ x_{i,j} + r.\left(x_{i,j} - DI_{k_i,j}\right), & \text{Otherwise} \end{cases} \tag{2.18}$$
$$i = 1, 2, \ldots, N_p, j = 1, 2, \ldots, N_v.$$

$$X_i = \begin{cases} X_i^{P1}, & F_i^{P1} < F_i \\ X_i, & \text{Otherwise} \end{cases} \tag{2.19}$$

where X_i^{P1} is the new computed status for the ith candidate solution based on the first phase of DTBO, $x_{i,j}^{P1}$ is its jth dimension, F_i^{P1} is its cost function value, I is a random number from $[1, 2]$,k_i is a random number from $[1, N_{DI}]$, $DI_{ki}$is a randomly selected driving instructor to train the ith member, $DI_{ki,j}$ is its jth dimension, and $F_{DI_{ki}}$ is its cost function value.

- End training phase.

3. **Learner driver patterning from instructor skills phase** (P_2)**:**

- Calculate the Patterning index (P) according to equation (2.20).

$$P = 0.01 + 0.9\left(1 - \frac{Z}{N_s}\right), \quad z = 1, 2, \ldots, N_S \tag{2.20}$$

- Generate the new position based on the linear combination of each member with the instructor using equation (2.21)and update it according to equation (2.22).

$$x_{i,j}^{P2} = P \cdot x_{i,j} + (1-P).DI_{k_i,j}, \quad i = 1,2,\ldots,N_p, j = 1,2,\ldots,N_v \quad (2.21)$$

$$X_i = \begin{cases} X_i^{P2}, & F_i^{P2} < F_i \\ X_i, & \text{Otherwise} \end{cases} \quad (2.22)$$

where X_i^{P2} is the new computed status for the ith candidate solution based on the second phase of DTBO, $x_{i,j}^{P2}$ is its jth dimension, F_i^{P2} is its cost function value.

- End learner driver patterning from instructor skills phase.

4. **Individual practice phase** (P_3)**:**

- Randomly generate a new position near each learner driver using equation (2.23) and update it according to equation (2.24), if it improve the cost function values.

$$\begin{gathered} x_{i,j}^{P3} = x_{i,j} + (1-2r) \cdot R \cdot \left(1 - \frac{z}{N_s}\right) \cdot x_{i,j} \\ i = 1,2,\ldots,N_p, j = 1,2,\ldots,N_v, z = 1,2,\ldots,N_s \end{gathered} \quad (2.23)$$

$$X_i = \begin{cases} X_i^{P3}, & F_i^{P3} < F_i \\ X_i, & \text{Otherwise} \end{cases} \quad (2.24)$$

where $R = 0.5$, X_i^{P3} is the new computed status for the ith candidate solution based on the third phase of DTBO, $x_{i,j}^{P3}$ is its jth dimension, F_i^{P3} is its cost function value.

- End individual practice.

5. Updated the best solution (X_{best}) in accordance with the best cost function value.

6. This process is continued until the maximum iteration number is attained.

2.6 Applications of metaheuristic algorithms

Metaheuristic algorithms have shown themselves to be beneficial tools for resolving complex optimization problems in various domains, such as:

- Engineering [124]: Where they are employed to solve complex optimization problems where traditional methods are inefficient. They help in areas such as structural design, scheduling, resource allocation, and control systems by finding near-optimal solutions in large, nonlinear, and multi-objective search spaces. They are used in manufacturing optimization, supply chain optimization, energy system optimization, process optimization, and structural optimization.
- Healthcare [125]: Where the metaheuristic algorithms are used to patient scheduling, resource allocation, treatment planning, medical image analysis, drug development, disease diagnosis, and medical equipment design.
- Computer science [125]: In this field, metaheuristic algorithms are applied to networks, machine learning, computer vision, optimization problems, and natural language processing.

2.7 Conclusion

In conclusion, this chapter provided a comprehensive overview of metaheuristic algorithm classification, with a particular focus on population-based metaheuristic optimization algorithms. It outlined their underlying principles and concepts, and detailed the operational steps of particle swarm optimization, teaching–learning-based optimization, and driving training–based optimization—three key algorithms employed in this thesis. Additionally, the chapter highlighted various application areas where population-based metaheuristics demonstrate significant potential.

Chapter 3

Neural Network Model and Metaheuristic Optimization based Predictive Control

3.1 Introduction

In this chapter, the principles of neural network model-based constrained predictive control using the metaheuristic optimization approach are introduced. Subsequently, the design steps of the developed control algorithms using PSO, TLBO, and DTBO algorithms are outlined.The results of applying these algorithms to control various nonlinear systems with different complexity are presented and discussed. A comparative study is conducted to demonstrate the effectiveness of the proposed control algorithms in terms of accuracy and computational efficiency.

3.2 Neural network model-based predictive control using metaheuristic optimization approach

Consider the system described by equation (3.1), as follows:

$$\begin{aligned} x(k+1) &= f(x(k), u(k), v(k)) \\ y(k) &= g(x(k), u(k)) \end{aligned} \tag{3.1}$$

where x, u, and y are the system state, input, and output vectors, respectively.v is the disturbance, and f and g are nonlinear functions. The Neural Network Model Predictive Control (NNMPC) based on metaheuristic approach is an advanced control technique that requires two essential steps:

- The first step consists of modeling the dynamics of the system to be controlled

using a neural network. The most commonly used model structure is given by:

$$y(k+1) = f_{NN}\left[u(k), u(k-1), \ldots, u\left(k-n_u\right), y(k), y(k-1), \ldots, y\left(k-n_y\right)\right] \tag{3.2}$$

where n_u and n_y denote the number of past control inputs and outputs, respectively. f_{NN} represents the neural network mathematical function.

The obtained model is used to predict the system outputs.

- The second step consists of determining the control to be applied to the system by solving the optimization problem given by :

$$\begin{aligned} &\min_{u(k)} J\left(\Delta u(k), \hat{y}(k), y_r(k)\right) \\ &= \sum_{j=N_1}^{N_2} \left[\left(y_r(k+j) - \hat{y}(k+j/k)\right)^T \Gamma_y \left(y_r(k+j) - \hat{y}(k+j/k)\right)\right] \\ &+ \sum_{j=1}^{N_u} \left[\Delta u^T(k+j-1) R \Delta u(k+j-1)\right] \end{aligned} \tag{3.3}$$

Subject to:

$$y_{\min} < \hat{y}(k+j) < y_{\max}, j = N_1, \ldots, N_2 \tag{3.4}$$

$$u_{\min} < u(k+j) < u_{\max}, j = 0, \ldots, N_u - 1 \tag{3.5}$$

$$\Delta u_{\min} < \Delta u(k+j) < \Delta u_{\max}, j = O, \ldots, N_u - 1 \tag{3.6}$$

$$\Delta u(k+j-1) = 0 \quad \text{For } j > N_u \tag{3.7}$$

where $\Delta u(k) = u(k) - u(k-1)$ is the control increment, $\hat{y}(k+j/k)$ is the predicted value of the system outputs, y_r is the desired reference trajectory,$\Gamma_y(y)$ is the output-dependent weight function, and R is a positive definite matrix. In the minimization problem defined above the input and output constraints are taken into consideration, and they are handled as follows:

- The input constraints given by equations (3.5), and (3.6) are handled by limiting the search space of the selected metaheuristic optimization algorithm.

- The output constraints are handled using the penalty approach, based on the addition of slack variables to the cost function, to penalize any violation of constraints [126]. The penalty function $\Gamma_y(y)$ is given by equation (3.8).

$$\Gamma_y(y) = \begin{bmatrix} \Gamma_{y_1}(y_1) & 0 & \dots & 0 \\ 0 & \Gamma_{y_2}(y_2) & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \Gamma_n(y_n) \end{bmatrix} \tag{3.8}$$

where

$$\Gamma_{yj}(y_j) = \begin{cases} \Gamma_{yj}(0) & \text{if } y_{j_{\min}} \leq y_j \leq y_{j_{\max}} \\ \Gamma_{yj}(0)\left[1 + C_{jy}\right] & \text{if } y_j \geq y_{j_{\max}} \text{ or } y_j \leq y_{j_{\min}} \end{cases} \tag{3.9}$$

with C_{jy} is employed to define the penalization degree, $j = 1, 2, ..., n$ where n is the number of system outputs and $y(k) = [y_1(k), y_2(k), \dots, y_n(k)]^T$.

3.3 Neural network model predictive control using the TLBO algorithm: Applied to speed control of PMSM

3.3.1 PMSM dynamic modeling

The state model for the Permanent Magnet Synchronous Motor(PMSM) in the two-phase (d, q) reference frame is given by equation (3.10) [127]:

$$\begin{cases} \dot{x}_1 = \left(\frac{1}{L_d}\right)\left[u_d - R_s x_1 + L_q x_2 x_3\right] \\ \dot{x}_2 = \left(\frac{1}{L_q}\right)\left[u_q - R_s x_2 - L_d x_1 x_3 - P_f x_3\right] \\ \dot{x}_3 = \left(\frac{Z_p}{J}\right)\left[T_e - \frac{f}{Z_p} x_3 - T_l\right] \end{cases} \tag{3.10}$$

where the state variables x_1, x_2 and x_3 are the stator currents i_d, i_q, and angular speed (w), respectively. The control inputs u_d and u_q are the stator voltages. R_s is the stator resistance, Z_p is the pole number, J is the inertia moment, L_d and L_q are the stator inductances, P_f is the flux linkage due to the permanent magnet, f is the viscous friction coefficient,and T_l is the load torque. T_e is the electromagnetic torque, given by equation (3.11).

$$T_e = \left(\frac{3}{2}\right) Z_p \left[P_f x_2 + (L_d - L_q)\, x_1 x_2\right] \tag{3.11}$$

The PMSM parameters employed in this work are listed in table 3.1 [127].

Table 3.1 – PMSM parameters.

Parameters	Values	Parameters	Values
Stator resistance(Ω)	2.98	Stator inductance L_q(H)	0.007
Poles number	2	Linkage flux (W_b)	0.125
Inertia moment(kg.m^2)	0.00004	Viscous coefficient	0.00011
Stator inductance L_d(H)	0.007		

3.3.2 Control strategy

In order to validate the proposed control strategy named Enhanced Neural Network Model Predictive Control based on TLBO algorithm (ENNMPC-TLBO), the speed control of the PMSM in the two-phase reference frame (d, q) is chosen as an application. The stator voltages are chosen as the control inputs vector given by: $\left(u(k) = [u_d, u_q]^T\right)$, the output to be controlled is chosen as the angular speed of the PMSM represented by:(w(k)), the reference trajectory is represented by:$(r(k))$, and the predicted angular speed of the PMSM is given by: $(\hat{w}(k))$. The control block diagram of the ENNMPC-TLBO applied to angular speed control of PMSM is illustrated in figure 3.1.

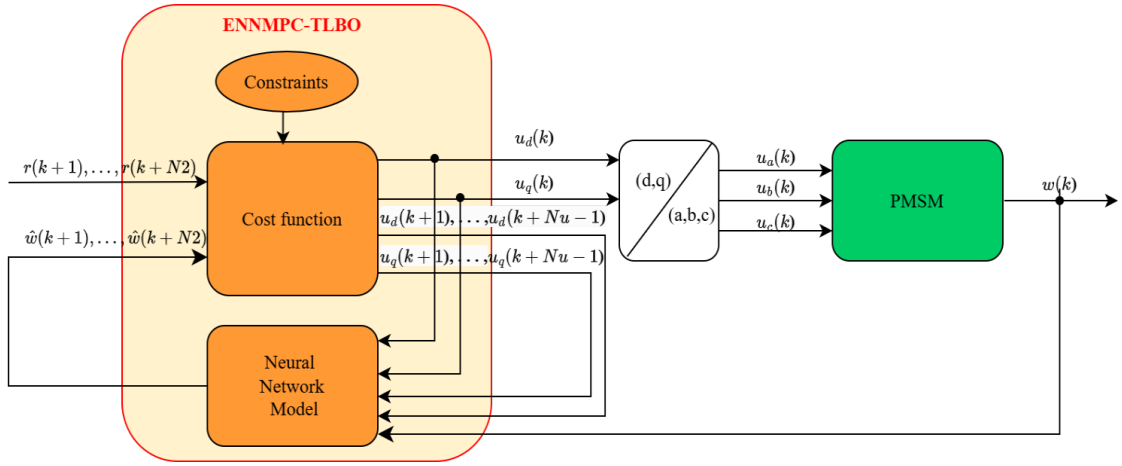

Figure 3.1 – Control block diagram of the ENNMPC-TLBO applied to the speed control of the PMSM.

The PMSM angular speed is predicted over a prediction horizon using the obtained model of the system under study. At each sampling time (k) an optimal sequence of the control input is obtained by online optimizing the cost function given by equation (3.12). To reduce the computational load a weighting factor is introduced into the cost function equation (3.12). This weighting factor is given by equation (3.13).

$$\begin{aligned}
&\min_{u(k)} J(\Delta u(k), \hat{y}(k), r(k)) \\
&= \sum_{j=N_1}^{N_2} \left[\mathrm{Q}_j \left(\hat{y}\left(k + \frac{j}{k}\right) - r(k+j) \right)^T \Gamma_y \left(\hat{y}\left(k + \frac{j}{k}\right) - r(k+j) \right) \right] \\
&+ \sum_{j}^{N_u} \left[\Delta u^T (k+j-1) R \Delta u(k+j-1) \right]
\end{aligned} \tag{3.12}$$

where:

$$Q_j = \frac{1}{1+j}, j = 1, 2, \ldots, N_2 \tag{3.13}$$

3.3.2.1 PMSM neural network modeling

Due to its simplicity and accuracy, a feed forward neural network is used to model the PMSM dynamic. The configuration of the used neural network model is described as follows:

- An input layer containing six inputs: control inputs $(u_d(k), u_d(k-1), u_q(k), u_q(k-1))$ and system outputs $(y(k), y(k-1))$.
- One hidden layer containing seven neurons with hyperbolic tangent sigmoid activation functions.
- An output layer composed of a one neuron with linear activation function.

For every operational regime of the system, two sets of data are produced using the state model of the PMSM provided by equation (3.10). The first set is used to train the neural network while the second one is used for the test purpose of the trained neural network. Figure 3.2 shows the test results of the neural network model:

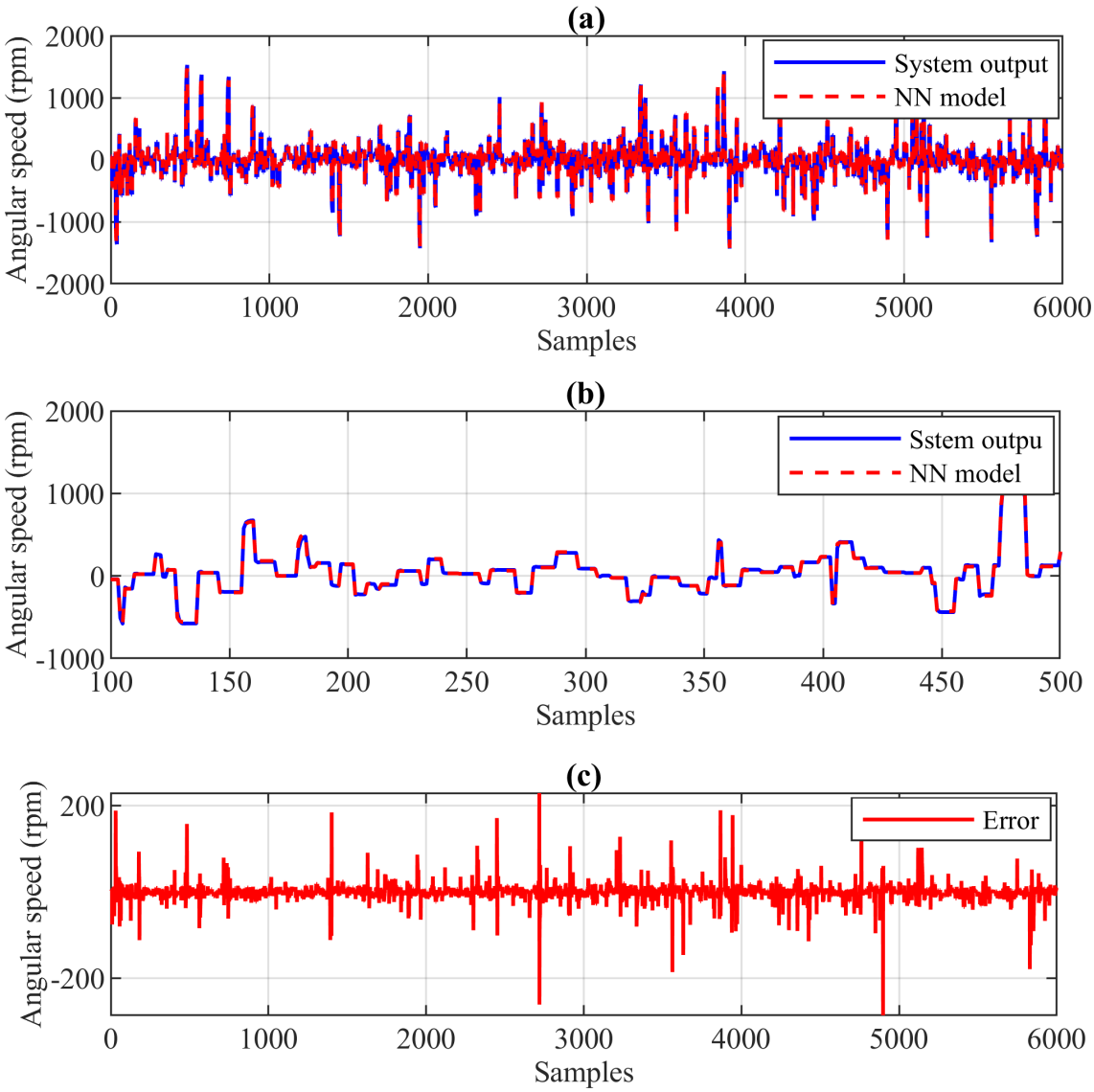

Figure 3.2 – Test results of the trained neural network model.

The modeling error shown in figure 3.2 is relatively low and the peak values are due to the hard change of the PMSM angular speed. The obtained values of Root Mean Square Error (RMSE), Mean Absolute Error (MAE), and the correlation coefficient (R^2) are 17.756(rpm), 9.131(rpm), and 0.996, respectively. These values confirm the good accuracy of the designed model.

3.3.2.2 Control algorithm

The main steps used to design the proposed control algorithm can be summarized as follows:

1. **Phase 1:** In this phase, the parameters of the ENNMPC-TLBO, the initial solutions and the reference trajectory are defined:

   - Set $X_{ik}^{j}, (i = 1, ..., N_v)$ as the control inputs at the sampling time k.

- Generate the initial solution according to equation (2.8).
- Define the desired reference trajectory from $k + N_1$ to $k + N_2$.

2. **Phase 2:** It's an optimization loop phase containing the following steps:

   - Determine the predicted values of the system output (PMSM angular speed) utilizing the obtained neural network prediction model.
   - Evaluate the cost function for the initial solution using equation (3.12).
   - Calculate the average score , and the distance ($D^j_{i_k}$) utilizing equations (2.9) and (2.10), respectively.
   - Update the previous solutions and calculate the ($X^j_{N_i k}$) using equation (2.11).
   - Calculate the future values of the system output using the new solutions ($X^j_{N_i k}$) and the neural network prediction model.
   - Evaluate the cost function for the new solutions using equation (3.12) and take as the best element as the new solution.
   - Select randomly a pair of solutions ($X^j_{i_\alpha}$) and ($X^j_{i_\beta}$) to interact, and calculate the corresponding cost function values F^j_α and F^j_β.
   - Use equations(2.12) and (2.13) to update the new founded solutions ($X^j_{N_i k}$).
   - Calculate the future values of the system output using the neural network prediction model.
   - Evaluate the corresponding values of the cost function of the new solutions using equation (3.12).
   - Determine the best solution.
   - Keep the best solution if the stop criteria is met or the number of the maximum iteration (T_{max}) is reached.

3. **Phase 3 :** Apply to the system, the first element of ($X^j_{i_{kbest}}$), which is the input control value .Return to phase 2 for the next sampling time.

3.3.3 Simulation results

Control performance of the proposed ENNMPC-TLBO is evaluated and compared to the Neural Network Model Predictive Control based on TLBO algorithm (NNMPC-TLBO) under the same parameters of the TLBO algorithm and the NNMPC, which are given in table 3.2, and table 3.3, respectively. The obtained simulation results are shown in figure 3.3, where the desired reference trajectory moves from 0 to 300 to 800 rpm. To evaluate the efficiency of the proposed controller in terms of computing time, and tracking accuracy, the Mean Absolute Error (MSE), the RMSE, the MAE, and the computing time are calculated and are given in table 3.4.

Table 3.2 – TLBO parameters

Parameters	Values	Parameters	Values
N_p	10	N_s	10
N_v	2		

Table 3.3 – NNMPC parameters

Parameters	Values	Parameters	Values
N_1	1	N_u	2
N_2	4	u_{qmax} (V)	28.87
Sampling time (s)	0.02	u_{dmin} (V)	-50
u_{dmax} (V)	50	$u_{q\ \text{min}}$ (V)	-28.87

Table 3.4 – MAE, MSE, RMSE and computing time values of the ENNMPC-TLBO AND NNMPC-TLBO

	NNMPC-TLBO	**ENNMPC-TLBO**
MAE (rpm)	17.7972	**16.6411**
MSE (rpm)	3980	**2055**
RMSE (rpm)	63.0865	**45.3325**
Computing time (ms)	19.4303	**12.8592**

As shown in figure 3.3, a good tracking accuracy of the reference trajectory is achieved for both controllers. On the other hand, after the analysis of the MSE, RMSE, MAE, and computing time values, gathered in table 3.4, we can say that the proposed controller is faster and more efficient than the NNMPC-TLBO.As a result, the ENNMPC-TLBO strategy gives good computational efficiency, and tracking accuracy, making it suitable for controlling nonlinear constrained systems with fast dynamics.

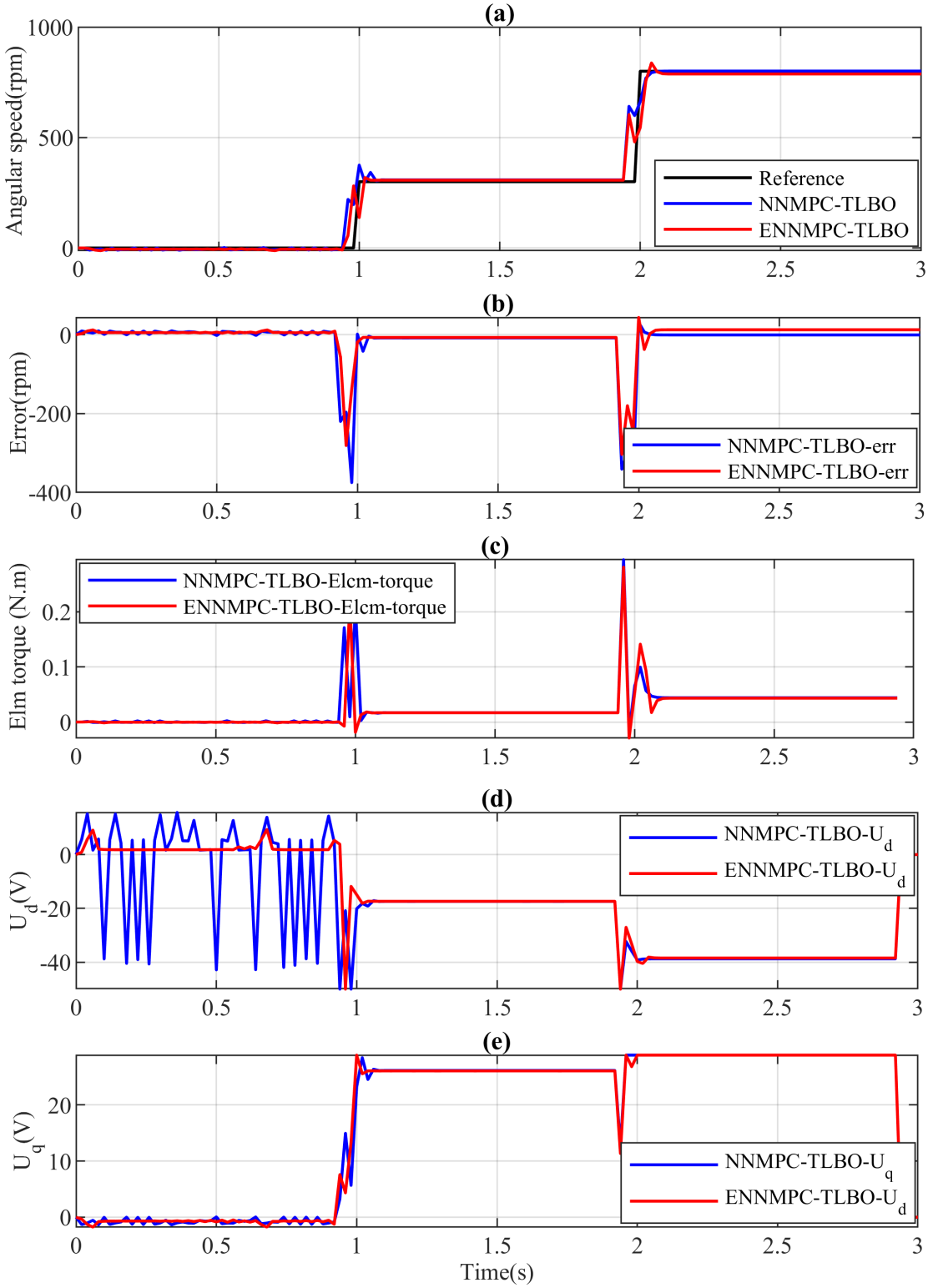

Figure 3.3 – Control results of the PMSM using ENNMPC-TLBO and NNMPC-TLBO controllers/ a) Speed response of the PMSM using multistep reference trajectory/ b) Error between reference trajectory and actual angular speed/ c) Electromagnetic torque response/ d) Input control signal u_d /e) Input control signal u_q.

3.4 DTBO-based neural network model predictive control: Applied to electric vehicle

3.4.1 Mathematical modeling of EV system

Several electric vehicle models have been proposed in the literature. The model used in this study [1], consists of a brushed DC motor, battery, velocity controller, mechanical transmission (gears) and tires, as shown in figure 3.4.

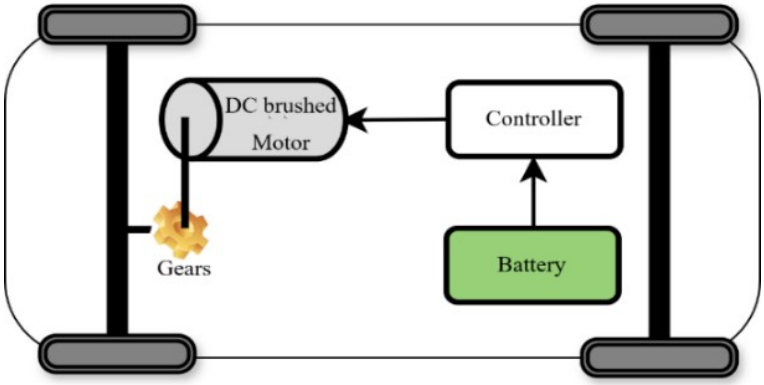

Figure 3.4 – EV block Diagram.

All the parameters used to derive the mathematical model of the EV are defined in table 3.5.

The equations representing the electrical circuit of the motor, the dynamics of the motor shaft, the dynamics of the load and the relationship between the angular velocity of the motor and the forward velocity of the vehicle are defined as follows [1]:

- Electrical circuit motor equations:

$$u = L_m \frac{di}{dt} + R_m i + k_b w_m \tag{3.14}$$

$$\frac{di}{dt} = \frac{-R_m}{L_m} i - \frac{k_b}{L_m} w_m + \frac{u}{L_m} \tag{3.15}$$

- Dynamics equations for the motor shaft:

$$J_m \frac{dw_m}{dt} = k_t i - w_m b_m - b\left(w_m - w_1\right) - k\left(\theta_m - \theta_1\right) \tag{3.16}$$

$$\frac{dw_m}{dt} = \frac{k_t}{J_m} i - \frac{(b_m + b)}{J_m} w_m - \frac{k}{J_m} \theta_m + \frac{b}{J_m} w_1 + \frac{k}{J_m} \theta_1 \tag{3.17}$$

$$\frac{d\theta_m}{dt} = w_m \tag{3.18}$$

Table 3.5 – EV system modeling Nomenclature.

Parameters	Definition
J_1	The inertia of a vehicle
J_m	The inertia of a motor
k	Constant for rotor connection
$\boldsymbol{k_b}$	Back EMF constant
k_t	Torque constant
L_m	Motor inductance
R_m	Motor resistance
s	Linear vehicle displacement
b	Springs for connection rotor damping
b_1	Vehicle damping
r_1	The motor-to-tire-meter gearing ratio
t_s	Rise time
w_1	Wheel rotational velocity
w_m	Rotational motor velocity
$\boldsymbol{x}$	State vector
β	Weight factor
θ_1	Displacement during wheel rotation
θ_m	Displacement during motor rotation
τ	Time constant.
$\boldsymbol{u}$	The rated voltage
ν	Electrical vehicle velocity

- Load dynamics equations:

$$J_1 \frac{dw_1}{dt} = b\left(w_m - w_1\right) + k\left(\theta_m - \theta_1\right) - b_1 w_1 \tag{3.19}$$

$$J_1 \frac{dw_1}{dt} = bw_m + k\theta_m - \left(b + b_1\right) w_1 - k\theta_1 \tag{3.20}$$

$$\frac{dw_1}{dt} = \frac{b}{J_1} w_m + \frac{k}{J_1}\theta_m - \frac{\left(b + b_1\right)}{J_1} w_1 - \frac{k}{J_1}\theta_1 \tag{3.21}$$

$$\frac{d\theta_1}{dt} = w_1 \tag{3.22}$$

- The relationship between the motor angular velocity and the vehicle driving velocity is given by equations (3.23)-(3.25).

$$\tau \frac{dv}{dt} + v = r_1 w_m \tag{3.23}$$

$$r_1 = \frac{v}{w_1} \tag{3.24}$$

$$\frac{ds}{dt} = v \tag{3.25}$$

The combination of the equations (3.14)-(3.25) gives the dynamical model representing the electric vehicle system given by equation (3.26) and (3.27).

$$\begin{cases} \dot{x}_1 = -\frac{R_m}{L_m}x_1 - \frac{k_b}{L_m}x_2 + \frac{1}{L_m}u \\ \dot{x}_2 = \frac{k_t}{J_m}x_1 - \frac{(b_m+b)}{J_m}x_2 - \frac{k}{J_m}x_3 + \frac{b}{J_m}x_4 + \frac{k}{J_m}x_5 \\ \dot{x}_3 = x_2 \\ \dot{x}_4 = \frac{b}{J_1}x_2 + \frac{k}{J_1}x_3 - \frac{(b+b_1)}{J_1}x_4 - \frac{k}{J_1}x_5 \\ \dot{x}_5 = x_4 \\ \dot{x}_6 = \frac{r_1}{\tau}x_4 - \frac{1}{\tau}x_6 \\ \dot{x}_7 = x_6 \\ y = x_6 \end{cases} \tag{3.26}$$

where:

$$x = \begin{bmatrix} i & d\theta_m/dt & \theta_m & d\theta_1/dt & \theta_1 & v & s \end{bmatrix}^T \tag{3.27}$$

3.4.2 Implementation of the control algorithm

The DTBO algorithm is used to solve the constrained optimization problem given by equation (3.3). The proposed control approach is evaluated by controlling the electric vehicle velocity, where the vehicle motor voltage is considered as the control input, and the vehicle velocity as the output to be regulated to the desired velocity. The application of the proposed controller named DTBO-based Constrained Neural Network Model Predictive Control (CNNMPC-DTBO) for EV velocity control is illustrated by the block diagram shown in figure 3.5.

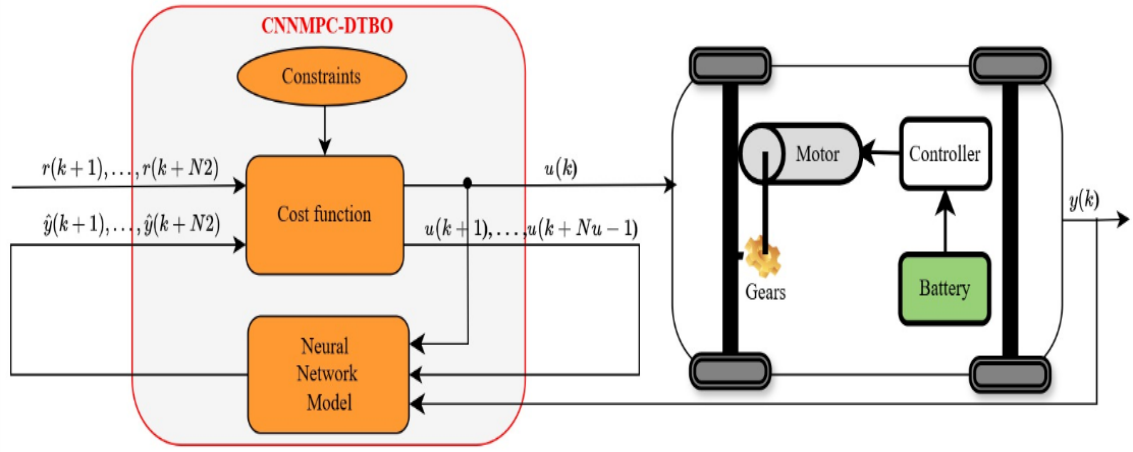

Figure 3.5 – Block diagram of the CNNMPC-DTBO applied to control EV system.

3.4.2.1 Neural network modeling of the EV system

To carry out the first stage in the implementation of the CNNMPC-DTBO algorithm, a feedforward neural network having the following structure is used to model

the electric vehicle system:

- An input layer containing four neurons and having the following input vector: $[u(k-1), u(k), v(k-1), v(k)]$
- A single hidden layer containing seven neurons with sigmoid activation function.
- An output layer with a single neuron and linear activation function representing the predicted electrical vehicle velocity.

Two sets of data are produced using the parameters given in table 3.6 and the EV dynamical model given in equation (3.26).The first set is used to train the neural network while the second one is used to test the trained neural network. Figure 3.6 shows the simulation test results of the trained neural network model.

Table 3.6 – EV and DC brushed motor parameters [1].

Parameters	Values	Parameters	Values
R_m	$0.1(\Omega)$	b_m	$1e-5$
L_m	0.01 (H)	u	36 (V)
k_b	$65e-5$ (V.sec/rad)	J_1	1000. J_m (Kg. m^2)
k_t	0.1 (Nm/A)	b_1	$1e-3$(N.m/(rad/s))
J_m	1e-4 ($Kg.m^2$)	k	1e2(N/m)
r_1	0.005 (m/rad)	b	0.1 (N.m/(rad/s))
τ	2 (s)		

As shown in figure 3.6 the modeling error is small and the obtained values of R^2 , MAE, MSE, and RMSE are $0.999999, 0.001044, 0.000004, and 0.002005$, respectively. These values demonstrate the good accuracy of the developed model.

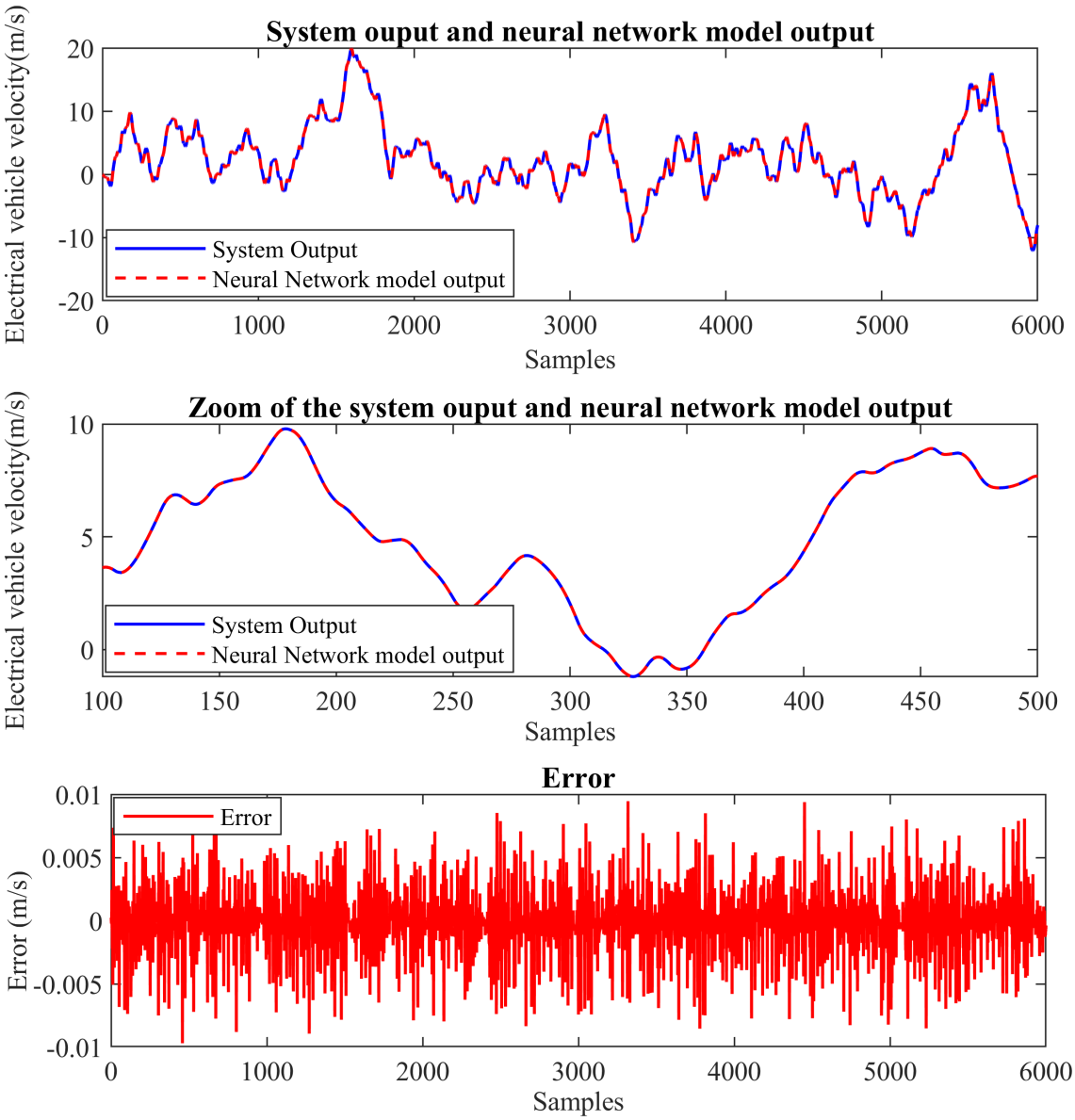

Figure 3.6 – Test results of the trained neural network model.

3.4.2.2 Control algorithm

The design steps of the proposed CNNMPC-DTBO algorithmare given below.

- **Stage 1:**
  - Initialize the CNNMPC-DTBO parameters which are (N_1, N_2, N_u, N_p, N_s , N_v, X_{min}, andX_{max}).
  - Define the reference trajectory of the EV velocity over the interval $[N_1, N_2]$. Let us take the control input that represents the motor voltage in the EV system, as $x_{i,j}$, where $i = [1, 2, ..., N_p]$, and $j = [1, 2, ..., N_v]$. Initialize $x_{i,j}$ according to equation (2.15)

  - Use the EV neural network model to predict the future values of the system output.
  - Evaluate the cost function value according to the equation (3.3).
  - Select the initial best solution X_{best}.
  - Set sampling time $k = 1$.

- **Stage 2:**
  - Use equation (2.18) to determine the new population position $x_{i,j}^{P1}$.
  - Utilize $x_{i,j}^{P1}$ to calculate the future values of the system output.
  - Evaluate the cost function value corresponding to $x_{i,j}^{P1}$.
  - Utilize equation (2.19) to update $x_{i,j}^{P1}$.
  - Use equation (2.20) to define the patterning index (P).
  - Generate the new population position $x_{i,j}^{P2}$ using equation (2.21).
  - Utilize $x_{i,j}^{P2}$ to calculate the future values of the system output.
  - Evaluate the cost function value corresponding to the $x_{i,j}^{P2}$.
  - Use equation (2.22) to update $x_{i,j}^{P2}$.
  - Generate a new population position for the third phase $x_{i,j}^{P3}$ using equation (2.23).
  - Use $x_{i,j}^{P3}$ to calculate the future values of the system output.
  - Compute the cost function value corresponding to the $x_{i,j}^{P3}$.
  - Use equation (2.24) to update $x_{i,j}^{P3}$.

- **Stage 3:**
  - Update the best founded solution X_{best}.
  - Check whether the number of iterations has been reached, keep the best solution X_{best} if this is the case and go on to stage 4, if not go on to stage 2.

- **Stage 4:**
  - Apply the obtained control value (the first element of X_{best}) on the system.
  - For the next sampling time ($k = k + 1$), return to stage 2.

3.4.3 Simulation results

The CNNMPC-DTBO algorithm is applied to control the EV velocity with multistep reference, and the obtained results are compared to those of the (Neural Network Model Predictive Control based on PSO algorithm: NNMPC-PSO) using the parameters gathered in table 3.7.The simulation results of these two techniques are illustrated in figures 3.7 and 3.8. The obtained values of MAE, MSE, RMSE and computing time of these two control techniques are gathered in table 3.8.

Table 3.7 – CNNMPC-DTBO parameters.

Parameters	Values	Parameters	Values
N_1	1	N_u	2
N_2	4	Sampling time (s)	0.2
X_{imin} (V)	30	X_{imax} (V)	-30
N_p	10	N_s	10
N_v	1		

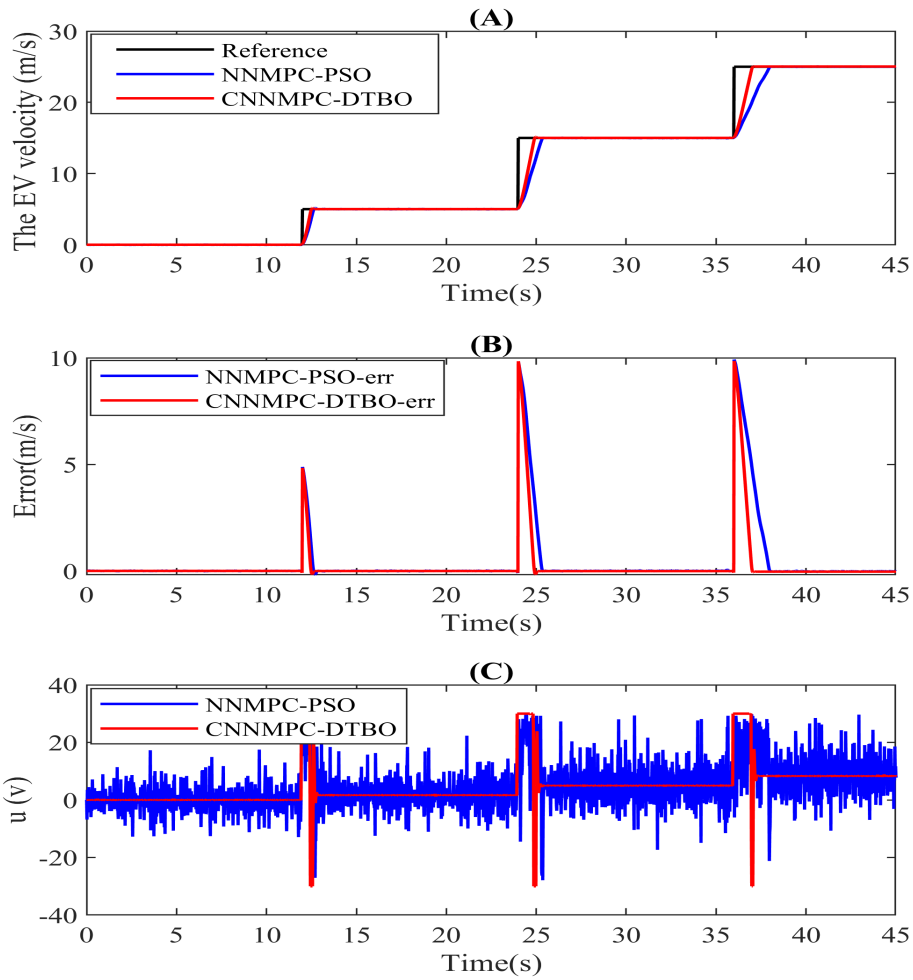

Figure 3.7 – Results of EV control velocity using CNNMPC-TLBO and NNMPC-PSO /A) Velocity response of the EV using multistep references/ B) Error between the reference and actual velocity/ C) Input control signal (Motor voltage).

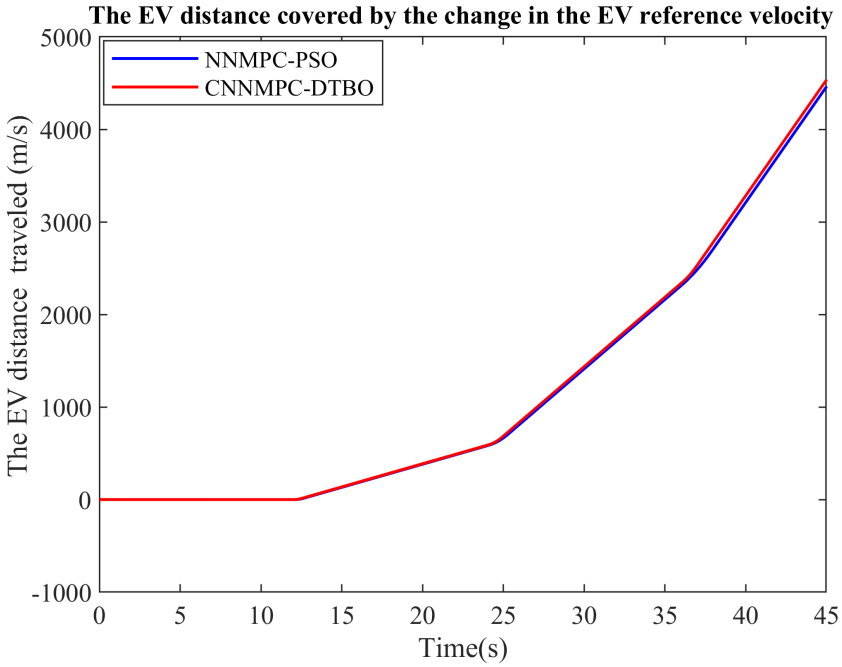

Figure 3.8 – The EV distance covered by the change in the EV reference velocity.

Table 3.8 – Computing time, MAE, MSE, and RMSE values of the CNNMPC-DTBO and NNMPC-PSO.

	NNMPC-PSO	**CNNMPC-DTBO**
MAE (m/s)	0.3770	**0.2521**
MSE (m/s)	2.4220	**1.6085**
RMSE (m/s)	1.5560	**1.2682**
Computing time (ms)	4.1915	**3.8809**

Due to the outcomes showed in figures 3.8 and 3.9, and the values gathered in table 3.8, we can conclude that the proposed algorithm is more efficient than the NNMPC-PSO algorithm in terms of MAE and computing time. Therefore, it can be concluded that the CNNMPC-DTBO outperforms the NNMPC-PSO in terms of accuracy, efficiency, and real-time application.

3.5 Improved neural network model predictive control based on DTBO algorithm: Applied to manipulator robot

3.5.1 Mathematical modelling and description of a two-degree-of-freedom manipulator robot

The Two Degree Of Freedom (2-DOF) manipulator robot used in this study is illustrated by figure 3.9.

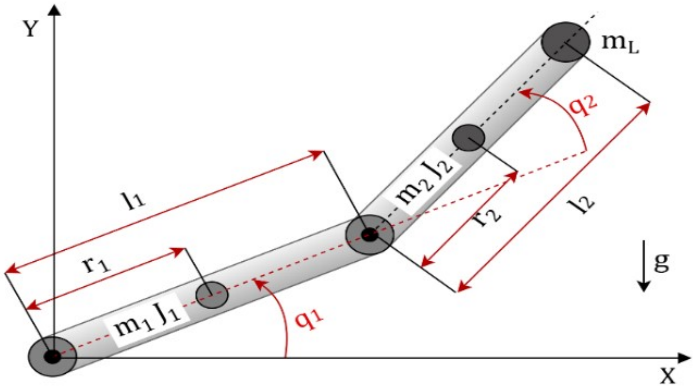

Figure 3.9 – 2-DOF arm robot.

The input variables of this arm robot are the torques of joints τ_1 and τ_2. The output variables are the angular positions q_1 and q_2 corresponding to joint 1 and 2, respectively. The mathematical model of the system under study is obtained by applying Lagrange's equation, and it is represented by equations (3.28) [128].

$$\begin{bmatrix} M_{11} & M_{12} \\ M_{21} & M_{22} \end{bmatrix} \begin{bmatrix} \ddot{q}_1 \\ \ddot{q}_2 \end{bmatrix} + \begin{bmatrix} V_1 \\ V_2 \end{bmatrix} + \begin{bmatrix} F_1 \\ F_2 \end{bmatrix} + \begin{bmatrix} G_1 \\ G_2 \end{bmatrix} = \begin{bmatrix} \tau_1 \\ \tau_2 \end{bmatrix} \tag{3.28}$$

Where:

$$M_{11} = I_1 + I_2 + m_1 r_1^2 + (m_2 + m_L) l_1^2 + m_2 + (2m_2 l_1 r_2 + 2m_L l_1 l_2) \cos(q_2) + m_L l_2^2$$
$$M_{12} = I_2 + m_2 r_2^2 + (m_2 l_1 r_2 + m_L l_1 l_2) \cos(q_2) + m_L l_2^2$$
$$M_{21} = M_{12}$$
$$M_{22} = I_2 + m_2 r_2^2 + m_L l_2^2$$
$$V_1 = -(m_2 l_1 r_2 + m_L l_1 l_2)(2\dot{q}_1 + \dot{q}_2)\dot{q}_2 \sin(q_2)$$
$$V_2 = (m_2 l_1 r_2 + m_L l_1 l_2)\dot{q}_1^2 \sin(q_2)$$
$$F_1 = \mu_1 \dot{q}_1$$
$$F_2 = \mu_2 \dot{q}_2$$
$$G_1 = (m_1 r_1 + m_2 l_1 + m_L l_1) g \cos(q_1) + (m_2 r_2 + m_L l_2) g \cos(q_1 + q_2)$$
$$G_2 = (m_2 + m_L) g r_2 \cos(q_1 + q_2)$$

This mathematical model of the 2-DOF arm robot is obtained using the nomenclature presented in table 3.9.

Table 3.9 – The nomenclature of 2-DOF arm robot.

Parameters	Definition
$m_{1,2}$	Mass of link 1 and 2 of the robot.
$l_{1,2}$	Length of each link of the robot.
$\boldsymbol{r_{1,2}}$	Length to center of mass of each link.
m_L	Load mass.
$I_{1,2}$	Inertia of each link.
$G_{1,2}$	Gravity vector
$\mu_{1,2}$	Friction coefficients.
g	Gravitational acceleration.
$\mathbf{M_{11,22}}$	Mass and inertia matrix.
$\boldsymbol{V_{1,2}}$	Centrifugal and Coriolis matrix
$F_{1,2}$	Friction vector.

The physical parameter values of the system described above are gathered in the table 3.10 [128].

3.5.2 Implementation of the proposed control strategy

For the NNMPC-DTBO controller, the predicted outputs of the system to be controlled are computed along a prediction horizon, so that for each sampling time (k), an optimal sequence of control input vector is obtained by on-line optimization of the cost function given by equation (3.3).

To validate the proposed control strategy named Improved Neural Network Model Predictive Control based on DTBO algorithm (INNMPC-DTBO), the control

Table 3.10 – Physical parameters of the 2-DOF arm robot.

Parameters	Values	Parameters	Values
m_1	0.3929243 kg	m_L	0.2 kg
m_2	0.0944039 kg	I_1	0.00114111 m^2 kg
l_1	0.2032 m	I_2	0.00202470 m^2 kg
l_2	0.1524 m	μ_1	0.141231 N m
r_1	0.104648 m	μ_2	0.353078 N m
r_2	0.081788 m	g	9.81 m/s^2

of the 2-DOF manipulator arm's angular positions is considered. The joint torques τ_1 and τ_2 are the control inputs vector, and the angular positions q_1 and q_2 of joints 1 and 2, respectively, are the outputs to be controlled. The control block diagram of the INNMPC-DTBO algorithm is illustrated in figure 3.10.

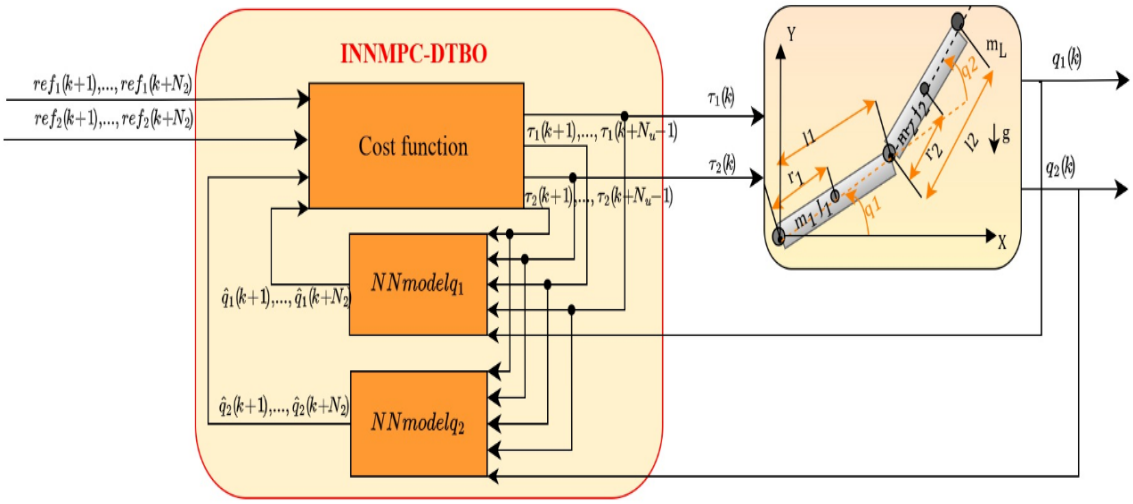

Figure 3.10 – Control block diagram of 2-DOF arm robot.

3.5.2.1 Neural network model of the 2-DOF arm robot

Since the robot manipulator has two outputs: the angular position at joints 1 and 2, two separate neural networks are used, one to predict each output with a sampling time $T_s = 10ms$. Both neural networks have the same configuration: an input layer containing eight neurons, one hidden layer containing twenty neurons with a sigmoid activation function, and an output layer with one linear neuron. The output of each neural network is the predicted angular position at each joint $\hat{q}_1(k+1)$ and $\hat{q}_2(k+1)$, respectively. The input vector of each neural network is structured as follows: $[\tau_1(k), \tau_1(k-1), \tau_2(k), \tau_2(k-1), q_1(k), q_1(k-1), q_2(k), q_2(k-1)]$

The training and test datasets are generated by solving equation (3.28) for random inputs within the operating range of the system.

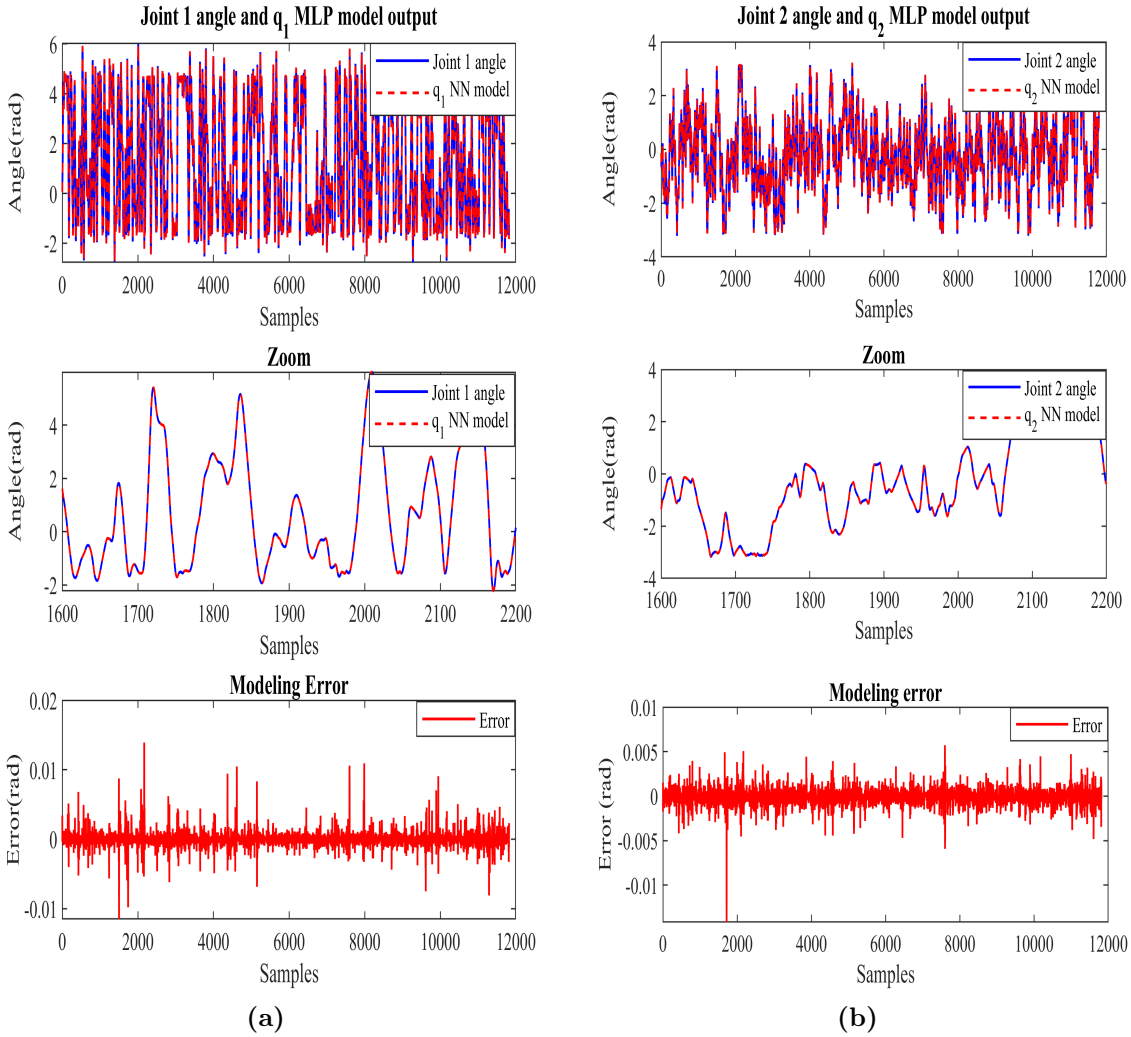

Figure 3.11 – Test results of the neural network model of angle1 and angle2.

The test results of the trained models are displayed in figure 3.11, where it is shown that the modeling error is quite small for the two neural network models. For the two separate neural networks, the corresponding RMSE and R^2 values are $0.00097279801, 0.99999982890, 0.00074415828,$ and 0.99999968739, respectively. The derived models show good accuracy, as affirmed by the close proximity of R^2 values to one and RMSE values to zero.

3.5.2.2 Control algorithm

The INNMPC-DTBO approach's phases are listed below.

1. Set the values of the INNMPC-DTBO parameters($N_1, N_2, N_u, N_p, N_s, N_v, X_{min}$, and X_{max}).

2. Determine the reference trajectories of the angular positions (q_1, q_2) for the interval $[N_1, N_2]$. Let us take the torques of joints 1 and 2 of the robot

manipulator as the control inputs, which are represented by $x_{i,j}$ with $j = [1, 2, ..., N_v]$, $i = [1, 2, ..., N_p]$. Initialize $x_{i,j}$ using equation (2.15).

3. Use the robot manipulator neural network models to predict the future values of the system outputs.
4. Use equation (3.12), to determine the cost function value of $x_{i,j}$.
5. Determine the initial best solution X_{best}.
6. Set sampling time $k = 1$.
7. Use equation (2.18) to calculate the new population position $x_{i,j}^{P1}$ and update it using equation (2.19).
8. Utilize $x_{i,j}^{P1}$ to predict the future values of outputs $\hat{q}_1(k+1)$ and $\hat{q}_2(k+1)$.
9. Determine the cost function value that corresponds to $x_{i,j}^{P1}$.
10. Use equation (2.19) to update $x_{i,j}^{P1}$.
11. Define the patterning index P using equation (2.20), and use it to compute $x_{i,j}^{P2}$ according to equation (2.21).
12. Use $x_{i,j}^{P2}$ to predict the outputs $\hat{q}_1(k+1)$ and$\hat{q}_2(k+1)$.
13. Determine the value of the cost function that corresponds to $x_{i,j}^{P2}$.
14. Use equation (2.22) to update $x_{i,j}^{P2}$.
15. Generate the population position $x_{i,j}^{P3}$ using equation (2.23).
16. Use $x_{i,j}^{P3}$ to calculate the predicted outputs of the system.
17. Determine the value of the cost function that corresponds to $x_{i,j}^{P3}$.
18. Update $x_{i,j}^{P3}$ using equation (2.24).
19. Update the value of X_{best}.
20. Verify if the maximum number of iterations has been reached. If so, continue to step 21 and retain the X_{best}; if not, return to step 7.
21. Apply to the system the first elements of the X_{best} , which are the optimal control inputs.
22. For the next sampling time ($k = k + 1$), return to step 7.

3.5.3 Simulation results

The control performance of the proposed INNMPC-DTBO is evaluated through a comparative study with TLBO-based NNMPC, PSO-optimized PID, and DTBO-based NNMPC. Using INNMPC-DTBO parameters gathered in table 3.11. The simulation study focuses on controlling the 2-DOF arm robot angular positions to track given reference trajectories. Figures 3.12 and 3.13 show the obtained simulation results. In order to evaluate the suggested controller concerning computing effectiveness and tracking precision, the MSE, RMSE, MAE, and the computational duration required to derive the control signals values, are computed and presented in table 3.12 for the four controllers.

Table 3.11 – INNMPC-DTBO parameters.

Parameters	Values	Parameters	Values
N_1	1	N_u	2
N_2	4	Sampling time (s)	0.01
X_{imax} (q_1)	15	X_{imin} (q_1)	-15
X_{imax} (q_2)	7	X_{imin} (q_2)	-7
N_p	8	N_s	8
N_v	2		

Table 3.12 – MAE, MSE, RMSE and computing time values of PID-PSO, NNMPC-TLBO, NNMPC-DTBO and INNMPC-DTBO.

	PID -PSO	NNMPC -TLBO	NNMPC -DTBO	**INNMPC -DTBO**
MAE	0.0862	0.0276	0.0284	**0.0250**
MSE	0.0698	0.0130	0.0126	**0.0124**
RMSE	0.3647	0.1608	0.1584	**0.1571**
Computing time (ms)	5.1906	4.9777	4.5107	**1.8086**

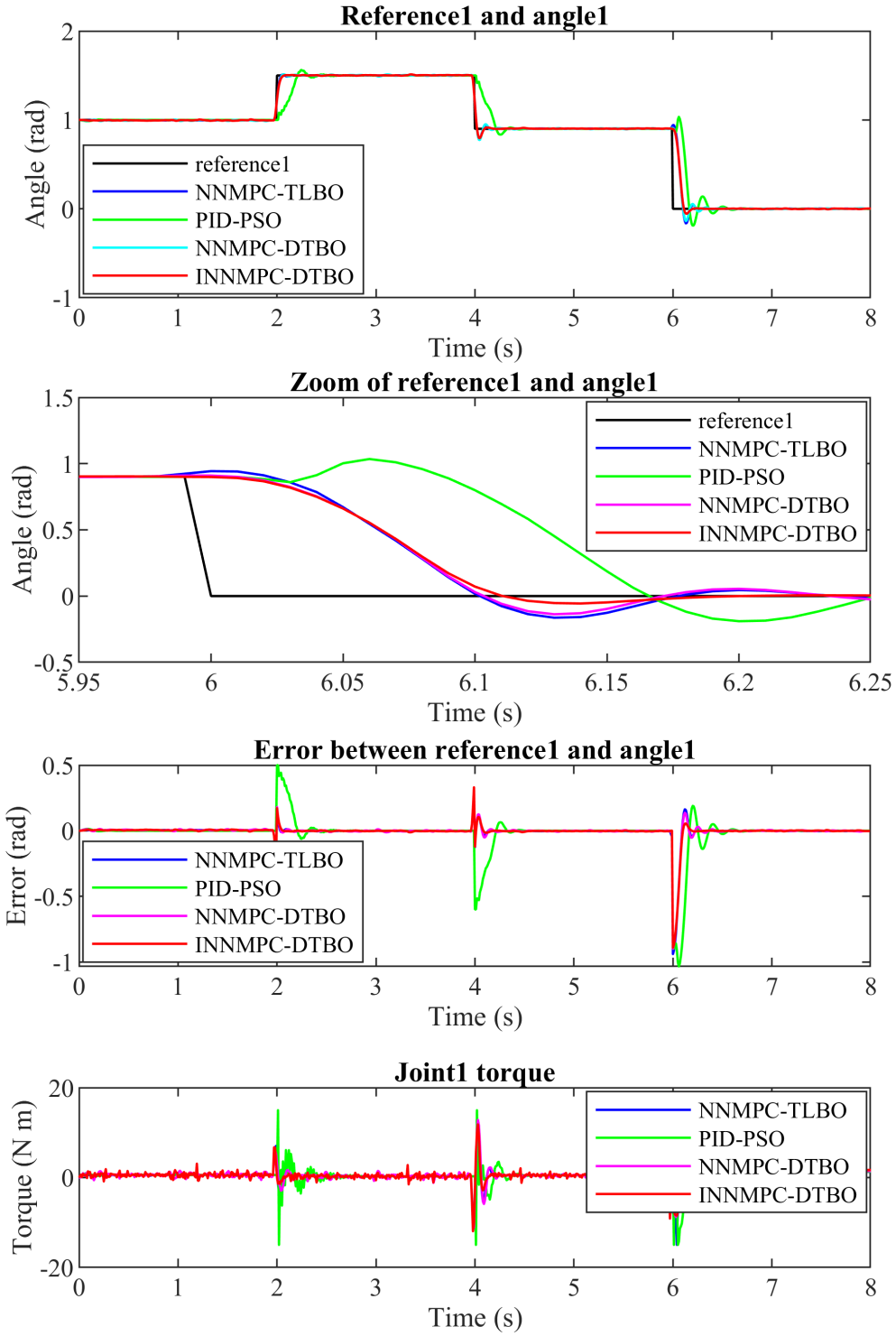

Figure 3.12 – Control performance of the robot with multi-step trajectories (angular position of joint1).

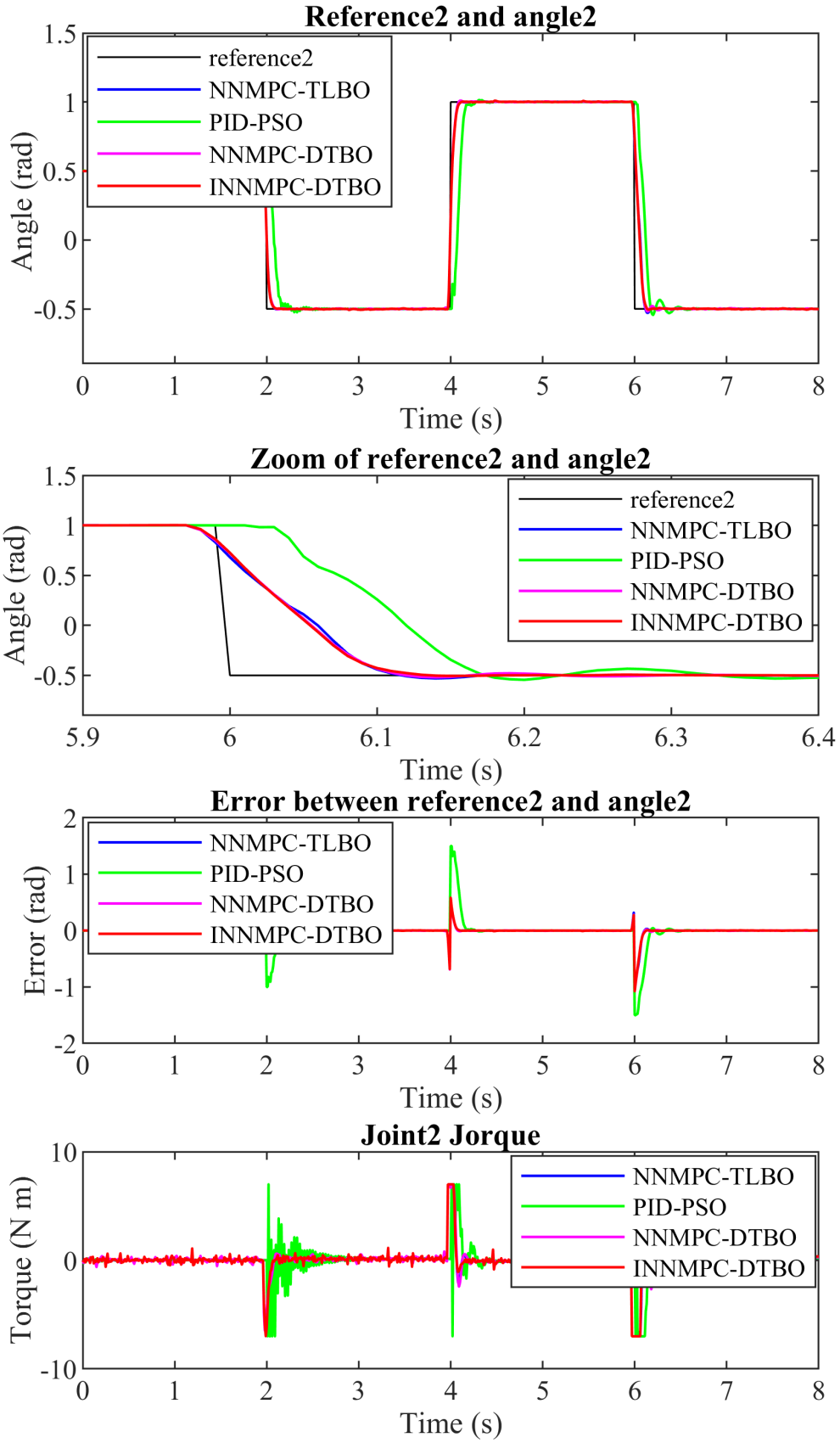

Figure 3.13 – Control performance of the robot with multi-step trajectories (angular position of joint2).

From the results illustrated by table 3.12, figures 3.12 and 3.13, we can say that

every controller give a good tracking accuracy of the reference trajectories. Nonetheless, the suggested controller's tracking error and computation time are somewhat lower than those of the other controllers. The proposed INNMPC-DTBO strategy gives good computational efficiency, and tracking accuracy, making it suitable for controlling nonlinear constrained systems with fast dynamics.

3.6 Conclusion

In this chapter, the principle of predictive control using neural network model and metaheuristic optimization approach, and three corresponding control were introduced. The first, enhanced the cost function of the TLBO-NNMPC, demonstrated superior accuracy and fast computation in PMSM motor speed control compared to conventional TLBO-NNMPC. The second, utilizing a DTBO optimization approach, proved highly effective in electric vehicle speed control, outperforming PSO-based NNMPC in both precision and computation time. The third, an improved version of the NNMPC based DTBO with a refined cost function, was applied to position control of a 2-DOF manipulator robot, showing better performance than the other controllers such as PSO-optimized PID, TLBO and DTBO-based NNMPC.

Chapter 4

Neural Network Predictive Control Design Using Lyapunov Function and Metaheuristic Optimization

4.1 Introduction

In this chapter, we introduce a new technique to control constrained nonlinear systems, named Lyapunov-based neural network model predictive control using metaheuristic optimization approach. This controller utilizes a feedforward neural network model as a prediction model and employs the driving training-based optimization algorithm to resolve the related constrained optimization problem. The proposed controller relies on the simplicity and accuracy of the feedforward neural network model and the convergence speed of the driving training-based optimization algorithm. The closed-loop stability of the developed controller is ensured by including the Lyapunov function as a constraint. The efficiency of the suggested controller is illustrated by controlling the angular speed of the three-phase squirrel cage induction motor. The obtained results are contrasted to those of other methods, specifically the fuzzy logic controller optimized by teaching learning-based optimization algorithm, the optimized PID with particle swarm optimization algorithm, the neural network model predictive controller based on particle swarm optimization algorithm, and the neural network model predictive controller using driving training based optimization algorithm.

4.2 Controller design

4.2.1 Principle

Lyapunov -based Model Predictive Control (LMPC) is a control approach integrating Lyapunov function as constraint in the optimization problem of MPC [129, 130] . This technique characterizes the region of the closed-loop stability, which makes it possible to define the operating conditions that maintain the system stability. Since its appearance, the LMPC method has been extensively utilized for controlling a various nonlinear systems, such as electrical systems [131] and wind power generation systems [130]. The proposed method includes Lyapunov -based constraint in the optimization problem of the neural network model predictive control, and uses the DTBO algorithm to compute the input control. This method is named Lyapunov-based Neural Network Model Predictive Control using DTBO algorithm (LNNMPC-DTBO).The corresponding minimization problem is given by equation (4.1), as follows:

$$\begin{aligned} &\min_{u(k)} J\left(\Delta u(k), \hat{y}(k), y_r(k)\right) \\ &= \sum_{j=N_1}^{N_2} \left[\left(y_r(k+j) - \hat{y}\left(k+{}^{j}/k\right)\right)^T \Gamma_y \left(y_r(k+j) - \hat{y}(k+j/k)\right)\right] \\ &\quad + \sum_{j=1}^{N_u} \left[\Delta u^T(k+j-1) R \Delta u(k+j-1)\right] \end{aligned} \tag{4.1}$$

Subject to:

$$y_{\min} < \hat{y}(k+j) < y_{\max}, j = N_1, \ldots, N_2 \tag{4.2}$$

$$u_{\min} < u(k+j) < u_{\max}, j = O, \ldots, N_u - 1 \tag{4.3}$$

$$\Delta u_{\min} < \Delta u(k+j) < \Delta u_{\max}, j = O, \ldots, N_u - 1 \tag{4.4}$$

$$\Delta u(k+j-1) = 0 \quad \text{For } j > N_u \tag{4.5}$$

$$\frac{\partial V}{\partial x} f(x(k), u(k), v(k)) \leq \frac{\partial V}{\partial x} f(x(k-1), u(k-1), v(k-1)) \tag{4.6}$$

where $u(k)$ is the actual input control, $\Gamma_y(y)$ is the output-dependent weight function, and V represents the continuous differentiable Lyapunov function.

4.2.2 Control algorithm

Considering $x_{i,j}$ as the control inputs, with $j = [1, 2, ..., N_v]$, $i = [1, 2, ..., N_p]$ and N_v is the control inputs number, the implementation steps of the LNNMPC-DTBO control algorithm are given as follows:

Algorithm 3 Pseudo code of the LNNMPC-DTBO algorithm

STEP 1: Initialization.
Set the MPC and DTBO parameters $(N_1, N_2, N_u, N_p, N_s, N_v, X_{min}, X_{max})$.
Initialize the position of the population $x_{i,j}$ using equation (2.15).
Set sampling time $k = 1$.
STEP 2: Evaluation of solutions.
Calculate the predicted outputs of the system using the neural network model.
Set a desired reference trajectory along $k + N_1$ to $k + N_2$.
Evaluate the cost function for $x_{i,j}$ using equation (4.1).
Select the best solution X_{best} that corresponds to the best value of the cost function.
STEP 3: Optimization loop.
for $z = 1$ to N_s **do**
  for $i = 1$ to N_p **do**
    for $j = 1$ to N_v **do**
      Generate the new position $x_{i,j}^{p1}$for the DTBO population using equation (2.18).
      Calculate the predicted outputs of the system by utilizing the neural network model and $x_{i,j}^{p1}$.
      Compute the cost function value for the new position $x_{i,j}^{p1}$.
      Use equation (2.19) to update $x_{i,j}^{p1}$.
      Calculate the patterning index according to the equation (2.20).
      Compute the new position $x_{i,j}^{p2}$ using equation (2.21).
      Calculate the system predicted outputs using the neural network model and $x_{i,j}^{p2}$.
      Compute the new value of the cost function for the new position $x_{i,j}^{p2}$.
      Update $x_{i,j}^{p2}$ using equation (2.22).
      Generate the new position of the DTBO population $x_{i,j}^{p3}$ with equation (2.23).
      Calculate the system predicted outputs using the neural network model and $x_{i,j}^{p3}$.
      Update $x_{i,j}^{p3}$ using equation (2.24).
    end for
  end for
  Update the best solution X_{best} after evaluationg the cost function of the best DTBO members.
end for
STEP 4:
Apply the best solution X_{best} to the system, which is the optimal control inputs.
Set $k = k + 1$ and go back to the point 5 for the next sampling time.

4.3 Stability analysis

This section uses the stability analysis given in [130] and performed in continuous time, to prove the proposed controller's closed-loop stability. Knowing that the optimization criterion of the proposed controller, defined in equation (4.1), can be written in continuous time as follows:

$$\min_{u} \int_{t_k}^{t_k+N} \left(y_r(\tau) - \hat{y}(\tau)\right)^T \Gamma_y \left(y_r(\tau) - \hat{y}(\tau)\right) d\tau + \int_{t_k}^{t_k+N_u} (\Delta u(\tau))^T R(\Delta u(\tau)) d\tau \tag{4.7}$$

Subject to:

$$y_{\min} < \hat{y}(t) < y_{\max}, \quad t \leq t_k + N \tag{4.8}$$

$$u_{\min} < u(t) < u_{\max}, t \leq t_k + N_u \tag{4.9}$$

$$\Delta u_{\min} < \Delta u(t) < \Delta u_{\max}, t \leq t_k + N_u \tag{4.10}$$

$$\Delta u(t) = 0 \quad \text{For } t > N_u \tag{4.11}$$

$$\frac{\partial V\left(x\left(t_k\right)\right)}{\partial x} f\left(x\left(t_k\right), u\left(t_k\right), v\left(t_k\right)\right) \leq \frac{\partial V\left(x\left(t_k\right)\right)}{\partial x} f\left(x\left(t_k\right), h\left(x\left(t_k\right)\right), v\left(t_k\right)\right) \tag{4.12}$$

where $h(x(t_k))$ is the previous input control, $u(t_k)$ is the actual input control, $N = N_2$ is the prediction horizon, $t \in [t_k, t_{k+1}]$ with $t_k = kT_s$ and T_s is the sampling time.

Assumption 1: Assuming that at the nominal operating point, the asymptotic stability of the closed-loop system is assured; all state constraints are satisfied in the stability region with respect to the existing feedback control input $u = h(x) \in U$ for all $x \in X_e$, where U and X_e are the spaces where the control u and the state are defined, respectively. Therefore, the inequalities bellow, given by equations (4.13)-(4.15) are valid for the inverse Lyapunov theorem-based nominal closed-loop system [132, 133].

$$\alpha_1(|x|) \leq V(x) \leq \alpha_2(|x|) \tag{4.13}$$

$$\frac{\partial V(x)}{\partial x} f(x, h(x), v) \leq -\alpha_3(|x|) \tag{4.14}$$

$$\left|\frac{\partial V(x)}{\partial x}\right| \leq \alpha_4(|x|) \tag{4.15}$$

where $\alpha_{i=1,2,3,4}$ are the κ-functions [134].

Let S as the region where the controlled system is stable under the input law $u = h(x)$ and $S_\gamma$the Lyapunov-based stability zone given by $S_\gamma := \{x \in X_e : V(x) <$

$\gamma\}$ with γ is a constant.

Lemma: The inequalities bellow, given by equations (4.16)-(4.18) are hold if there exist positive constants $C, L_x, L_v, \bar{L}_x$ and , $\bar{L}_v$, such that the constraints are satisfied for all x and u.

$$|f(x,u,v) - f(\bar{x},u,\bar{v})| \leq L_x|x-\bar{x}| + L_v|v-\bar{v}| \tag{4.16}$$

$$|f(x,u,v)| \leq C \tag{4.17}$$

$$\left|\frac{\partial V(x)}{\partial x} f(x,u,v) - \frac{\partial V(\bar{x})}{\partial x} f(\bar{x},u,\bar{v})\right| \leq \bar{L}_x|x-\bar{x}| + \bar{L}_v|v-\bar{v}| \tag{4.18}$$

where $\bar{x} = x(t_k)$ and $\bar{v} = v(t_k)$.

Proof: The equations (4.16) to (4.18) can be easily deduced by using the Lipschitz property of the function f and the continuous differentiability of the Lyapunov function $V(x)$, that's why the details are omitted, and the proof is considered complete.

Assumption2: assuming that the following scalar exists and is defined by the equation (4.19), as follows:

$$\delta_{\min} = \max\{V(x(t+T_s)) : V(x(t) \leq \delta_s)\} \tag{4.19}$$

where δ_s is a constant and δ_{min} is the maximum value of $V(x(t+T_s))$ at the next sampling time when $V(x(t))$ is less than the constant δ.

Theorem: For the closed-loop system given in equation (3.1) and controlled by solving the minimization problem given by equations (4.7)- (4.12), if it exists $x(t_0) \in S_\delta, \epsilon > 0$ and $\delta > \delta_s > 0$where ϵ is a constant, the inequality given by equation (4.20) can be hold assuming that the equations (4.13) to (4.15) are satisfied.

$$-\alpha_3\left(\alpha_2^{-1}(\delta\ \mathrm{s})\right) + \overline{\mathrm{L}}_{\mathrm{x}}\mathrm{C}T_s + \overline{\mathrm{L}}_{\mathrm{v}}\varphi \leq -\frac{\varepsilon}{T_s} \tag{4.20}$$

where for all $t \geq t_0$ there exists ($\delta > \delta_{min} > 0$) guaranteeing the maintaining of states $x(t)$ in the stability region and ϕ is the upper limit of disturbance variation over a sampling period.

Proof: The time-derivative of the Lyapunov function can be computed using equation (4.21), taking into account the closed-loop system at time $t \in [t_k, t_{k+1}]$:

$$\dot{V}(x(t)) = \frac{\partial V(x(t))}{\partial x} f(x(t),u(t),v(t)) \tag{4.21}$$

The following inequality can be extended from equation (4.21) based on the constraint (4.12) as follows:

$$\dot{V}(x(t)) \leq \frac{\partial V(x(t))}{\partial x} f(x(t), u(t), v(t)) - \frac{\partial V\left(x\left(t_k\right)\right)}{\partial x} f\left(x\left(t_k\right), u\left(t_k\right), v\left(t_k\right)\right) + \frac{\partial V\left(x\left(t_k\right)\right)}{\partial x} f\left(x\left(t_k\right), h\left(x\left(t_k\right)\right), v\left(t_k\right)\right) \tag{4.22}$$

Using equation (4.1), equation (4.22) can be written as follows:

$$\dot{V}(x(t)) \leq \frac{\partial V\left(x\left(t_k\right)\right)}{\partial x} f\left(x\left(t_k\right), h\left(x\left(t_k\right)\right), v\left(t_k\right)\right) + \bar{L}_x \left|x(t) - x\left(t_k\right)\right| + \bar{L}_v \left|v(t) - v\left(t_k\right)\right| \tag{4.23}$$

The equation (4.24) below can be written according to equation (4.22) and during a sampling period where $x(t)$ is continuous:

$$\left|x(t) - x\left(t_k\right)\right| \leq C T_s \tag{4.24}$$

It is deduced from equation (4.14) that:

$$\frac{\partial V\left(x\left(t_k\right)\right)}{\partial x} f\left(x\left(t_k\right), h\left(x\left(t_k\right)\right), v\left(t_k\right)\right) \leq -\alpha_3\left(\alpha_2^{-1}\left(\delta_s\right)\right) \tag{4.25}$$

Replacing equations (4.24) and (4.25) in equation (4.23), the equation (4.26) bellow can be deduced:

$$\dot{V}_{(x(t))} \leq -\alpha_3\left(\alpha_2^{-1}(\delta s)\right) + \bar{L}_x C T_s + \bar{L}_v \varphi \tag{4.26}$$

If equation (4.20) is fulfilled, then there is an $\epsilon > 0$ such that the subsequent inequality given by equation (4.27) hold.

$$\dot{V}(x(t)) \leq -\frac{\varepsilon}{T_s} \tag{4.27}$$

The function $V(x(t))$ continues to decrease at $t \in [t_k, t_{k+1}]$, as demonstrated by the previous inequality. From equation (4.27) we can write:

$$\dot{V}(x(\mathrm{t})) = \frac{V\left(x\left(t_{k+1}\right)\right) - V\left(x\left(t_k\right)\right)}{T_s} \leq \frac{-\varepsilon}{T_s} \tag{4.28}$$

Which result to the following inequality.

$$V\left(x\left(t_{k+1}\right)\right) \leq V\left(x\left(t_k\right)\right) - \varepsilon \tag{4.29}$$

Therefore, if $x(t_k) \in S_\delta$then at time $t \geq t_k$, all states of $x(t)$ remain in the stable area S_δ. Furthermore, over a finite number of sampling times, the states $x(t)$ will progressively converge to $S_{\delta_{min}}$ and remain in this stable area for all future periods.

4.4 Simulation study

Several simulations are carried out to demonstrate the efficiency of the proposed LNNMPC-DTBO controller, using as an application a highly nonlinear system: the squirrel cage induction motor. A comparison study between the suggested controller and the NNMPC-DTBO, NNMPC-PSO, FLC-TLBO, and PID-PSO is conducted.

4.4.1 Squirrel cage induction motor description

The squirrel cage induction motor is one of the three-phase induction motors currently used in industry in various applications, such as pumps, conveyors, and turbines. This type of motor represents an excellent application for scientific research thanks to the number of nonlinearities it contains. The state model of this type of motors is given by equation (4.30), as follows [135]:

$$\begin{cases} \dot{x}_1 = w_b \left(U_{ds} + \frac{w_e}{w_b} x_2 + \frac{R_s}{X_{is}} \left(\frac{X_{ml}}{X_{ir}} x_3 + \left(\frac{X_{ml}}{X_{is}} - 1 \right) x_1 \right) \right) \\ \dot{x}_2 = w_b \left(U_{qs} - \frac{w_e}{w_b} x_1 + \frac{R_s}{X_{is}} \left(\frac{X_{ml}}{X_{ir}} x_4 + \left(\frac{X_{ml}}{X_{is}} - 1 \right) x_2 \right) \right) \\ \dot{x}_3 = w_b \left(U_{dr} + \frac{(w_e - w_r)}{w_b} x_4 + \frac{R_r}{X_{ir}} \left(\frac{X_{ml}}{X_{is}} x_1 + \left(\frac{X_{ml}}{X_{ir}} - 1 \right) x_3 \right) \right) \\ \dot{x}_4 = w_b \left(U_{qr} - \frac{(w_e - w_r)}{w_b} x_3 + \frac{R_r}{X_{ir}} \left(\frac{X_{ml}}{X_{is}} x_2 + \left(\frac{X_{ml}}{X_{ir}} - 1 \right) x_4 \right) \right) \\ y = w_r \end{cases} \tag{4.30}$$

where the state vector $x = [F_{ds} F_{qs} F_{dr} F_{qr}]^T$ are the stator and rotor flux linkages in the reference frame dq axis. The dq axis stator voltages are the control inputs $[U_{ds} U_{qs}]$ and $U_{dr} = U_{qr} = 0$ are the rotor dq axis voltages. The motor's angular electrical base frequency is given by $w_b = 2\pi f_b$, w_e is the angular speed of the reference frame, and w_r is the angular speed of the rotor. X_{ml}, X_i, X_{ir}, X_m and $\dot{w}_r$ are given by equations (4.31)-(4.35), respectively.

$$X_{ml} = \frac{1}{\frac{1}{X_{is}} + \frac{1}{X_{ir}} + \frac{1}{X_m}} \tag{4.31}$$

$$X_{is} = w_b L_{is} \tag{4.32}$$

$$X_{ir} = w_b L_{ir} \tag{4.33}$$

$$\mathrm{X_m} = \mathrm{w_b}\ \mathrm{L_m} \tag{4.34}$$

$$\dot{w}_r = \left(\frac{P}{2J}\right)(T_e - T_r) \tag{4.35}$$

where T_e is the electrical output torque, T_r is the load torque, L_{is} is the stator phase inductance , L_{ir} is the rotor phase inductance and L_m is the mutual inductance.

The electrical parameters of the considered squirrel cage induction motor are given in Table 4.1 [135]:

Table 4.1 – Induction motor parameters.

Parameters	Value
Stator phase inductance ($\boldsymbol{L_{is}}$)	$0.21e-3H$
Rotor phase inductance ($\boldsymbol{L_{ir}}$)	0.6e-3H
Mutual inductance ($\boldsymbol{L_m}$)	4e-3H
Rotor phase resistance ($\boldsymbol{R_s}$)	0.19Ω
Stator phase resistance ($\boldsymbol{R_r}$)	0.39Ω
Poles number ($\boldsymbol{P}$)	2
Inertia moment (J)	0.0226kgm^2
Base frequency ($\boldsymbol{f_b}$)	100 Hz

4.4.2 Neural network modeling of the squirrel cage induction motor

The model of the induction motor given in equation (4.30) is approximated using four feedforward neural network with a similar structure: an input layer containing 6 neurons, one hidden layer containing 15 neurons with a sigmoid activation function, and an output layer with one neuron and having a linear activation function. The input vector for each neural network is given by $[U_{ds}(k), U_{qs}(k-1), U_{qs}(k), F_i(k-1), F_i(k)]$ where $i = \{ds, qs, dr, qr\}$. The outputs are $\hat{F}_{ds}, \hat{F}_{qs}, \hat{F}_{dr}$ and $\hat{F}_{qr}$, that represent the dq stator and rotor fluxes, respectively. A dataset is generated using random inputs and the state model given in equation (4.1). The test results of the obtained neural network models are shown in figure 4.1 and the prediction errors are shown in figure 4.2.

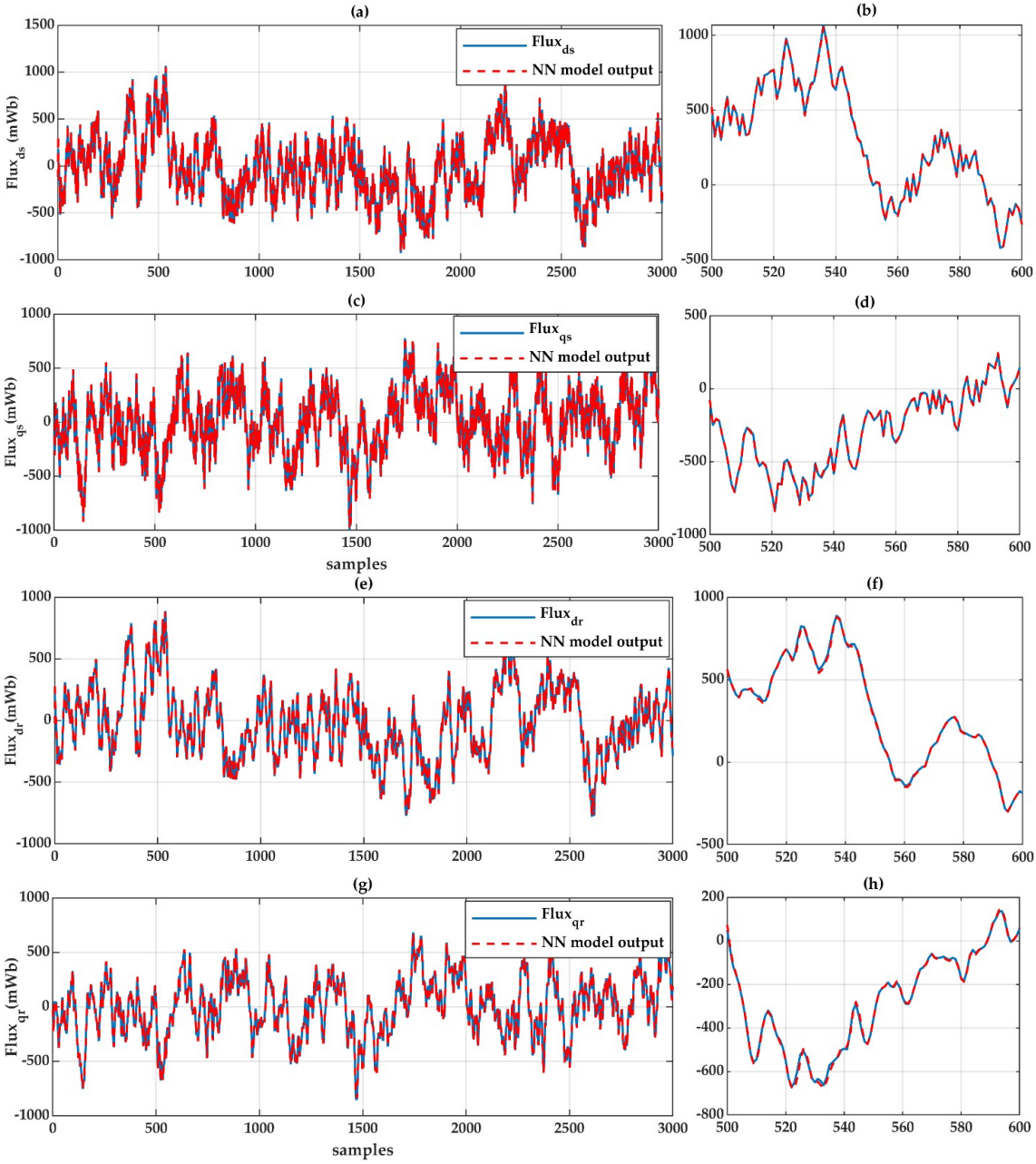

Figure 4.1 – Estimated and mesured fluxes: (a) Flux ds (continuous line) and NN model output (dashed line), (b) Zoom of (a), (c) Flux qs (continuous line) and NN model output (dashed line), (d) Zoom of (c), (e) Flux dr (continuous line) and NN model output (dashed line), (f) Zoom of (e)., (g) Flux qr (continuous line) and NN model output (dashed line), (h) Zoom of (g).

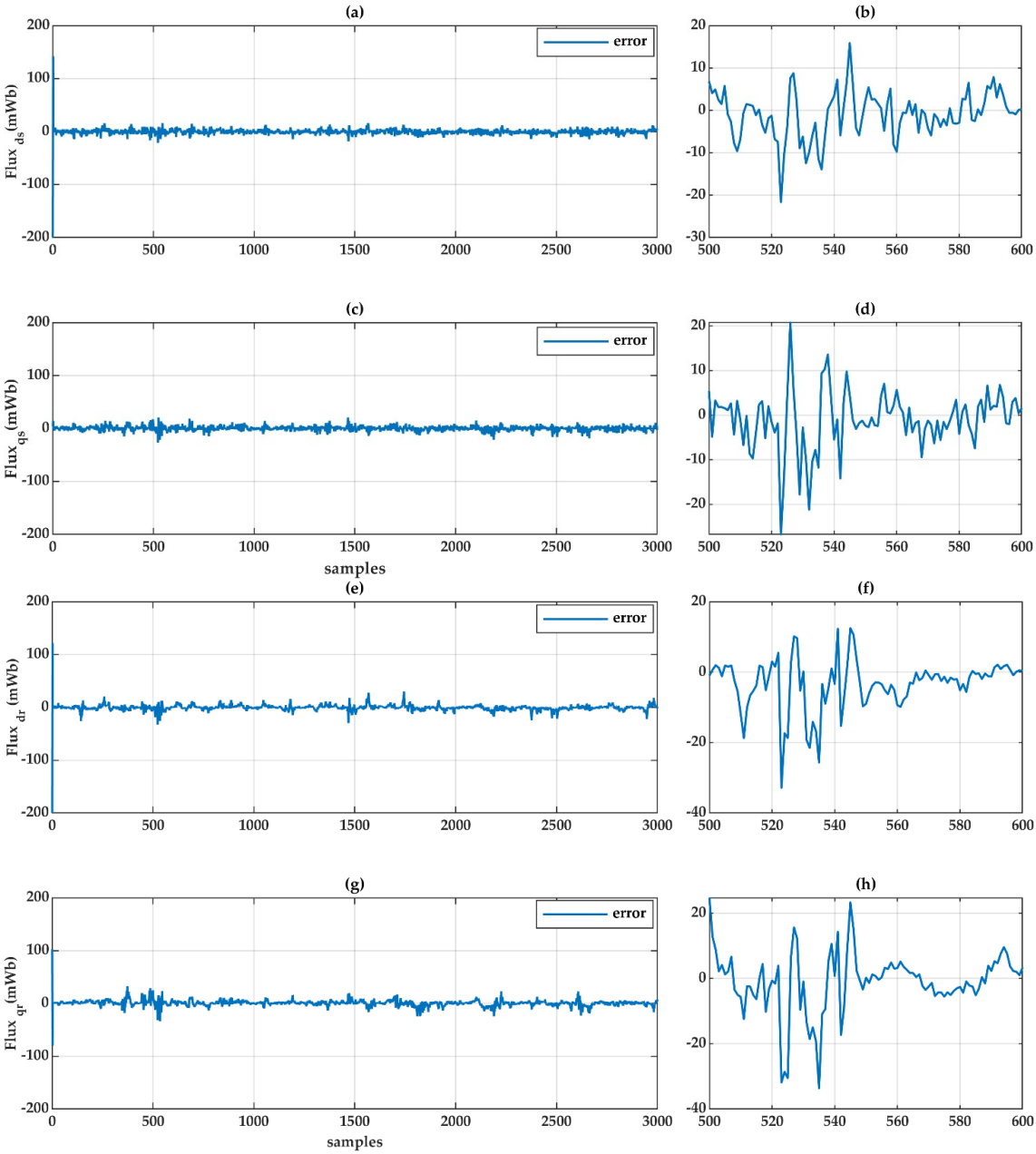

Figure 4.2 – Modeling errors: (a) Modeling error of ds flux model, (b) Zoom of (a), (c) Modeling error of qs flux model. (d) Zoom of (c), (e)) Modeling error of dr flux model, (f) Zoom of (e), (g)) Modeling error of dr flux model (f) Zoom of (e), (g) Modeling error of qr flux model.

The data of table 4.2 shows that the obtained models are quite accurate.

Table 4.2 – Metrics of the NNs prediction models.

Neural Network Metrics	F_{ds}	F_{qs}	F_{dr}	F_{qr}
MAE	3.1710	3.1625	3.2703	3.6247
MSE	37.7623	18.3092	3.2703	33.3081
RMSE	6.1451	4.2789	6.7616	5.7713
R^2	0.9996	0.9998	0.9995	0.9995

4.4.3 Control implementation

The proposed LNNMPC-DTBO is illustrated by the block diagram of figure 4.3 ,given below.

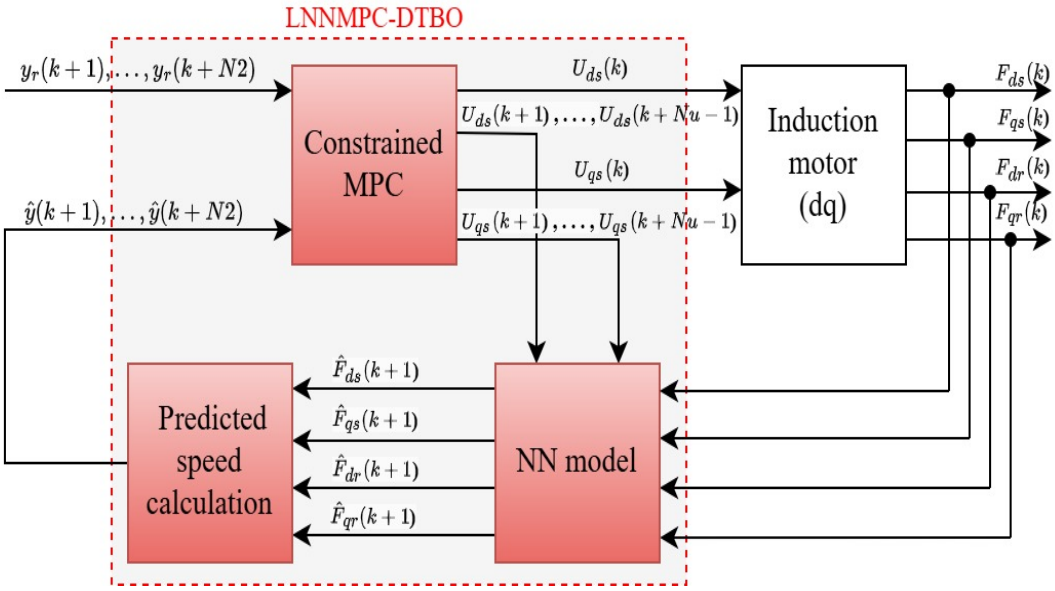

Figure 4.3 – Block diagram of induction motor speed control.

Several simulations are carried out to demonstrate the efficiency of the proposed LNNMPC-DTBO controller, using as application a highly nonlinear system: the squirrel cage induction motor. A comparison study between the suggested controller and the NNMPC-DTBO, NNMPC-PSO, FLC-TLBO and PID-PSO is conducted.

Two simulation cases are performed using the obtained neural network models and the parameters of the NNMPC, PSO, and DTBO presented in tables 4.3 to 4.5, respectively.

In the first case, no overshoot constraint is considered, and the multistep reference trajectory is used to evaluate the control performance of the proposed controller. The results are compared to those obtained from PID-PSO, FLC-TLBO, NNMPC-PSO, and NNMPC-DTBO, as shown in figure 4.4. The values of the MAE, MSE,

Table 4.3 – NNMPC parameters.

Parameters	Values	Parameters	Values
N_1	1	N_u	1
N_2	4	R	$4e-4$
Sampling time (s)	0.01	$U_{\min}$ (V)	-326
$U_{\max}$(V)	326		

Table 4.4 – PSO parameters.

Parameters	Values	Parameters	Values
c_1	2	w	1
c_2	2	w_d	0.99
N_p	10		

Table 4.5 – DTBO parameters.

Parameters	Values	Parameters	Values
N_p	10	N_v	2
N_s	10		

and RMSE are given in table 4.6. These values demonstrate that the proposed controller (LNNMPC-DTBO) gives the best tracking accuracy.The values that are bolded represent the best values.

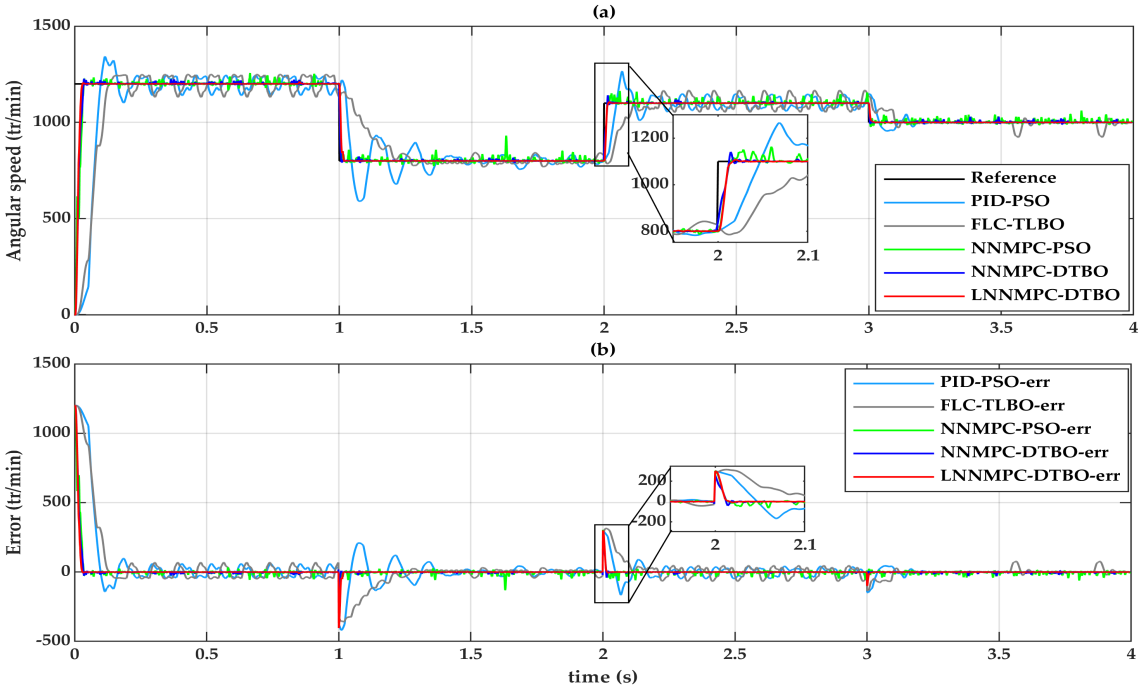

Figure 4.4 – Performance of induction motor speed control with multi-step trajectory, without constraint: (a) Reference trajectory and actual angular speed, (b) Error between reference trajectory and actual angular speed.

Table 4.6 – MAE, MSE, RMSE values without constraint.

	MAE	MSE	RMSE
LNNMPC-DTBO	6.2805	4.3214e + 03	65.7375
NNMPC-DTBO	8.0086	4.4242e + 03	66.5149
NNMPC-PSO	12.9689	4.4818e + 03	66.9462
FLC-TLBO	57.7139	2.3954e + 04	154.7719
PID-PSO	53.3979	2.4424e + 04	156.2831

In the second case, a 1% overshoot constraint is imposed for both multi-step and sinusoidal reference trajectories.

The obtained results for the second case, using multistep and sinusoidal reference trajectories are presented in figures 4.5 and 4.8, respectively, and the control efforts signals are given in figures 4.7 and 4.9.

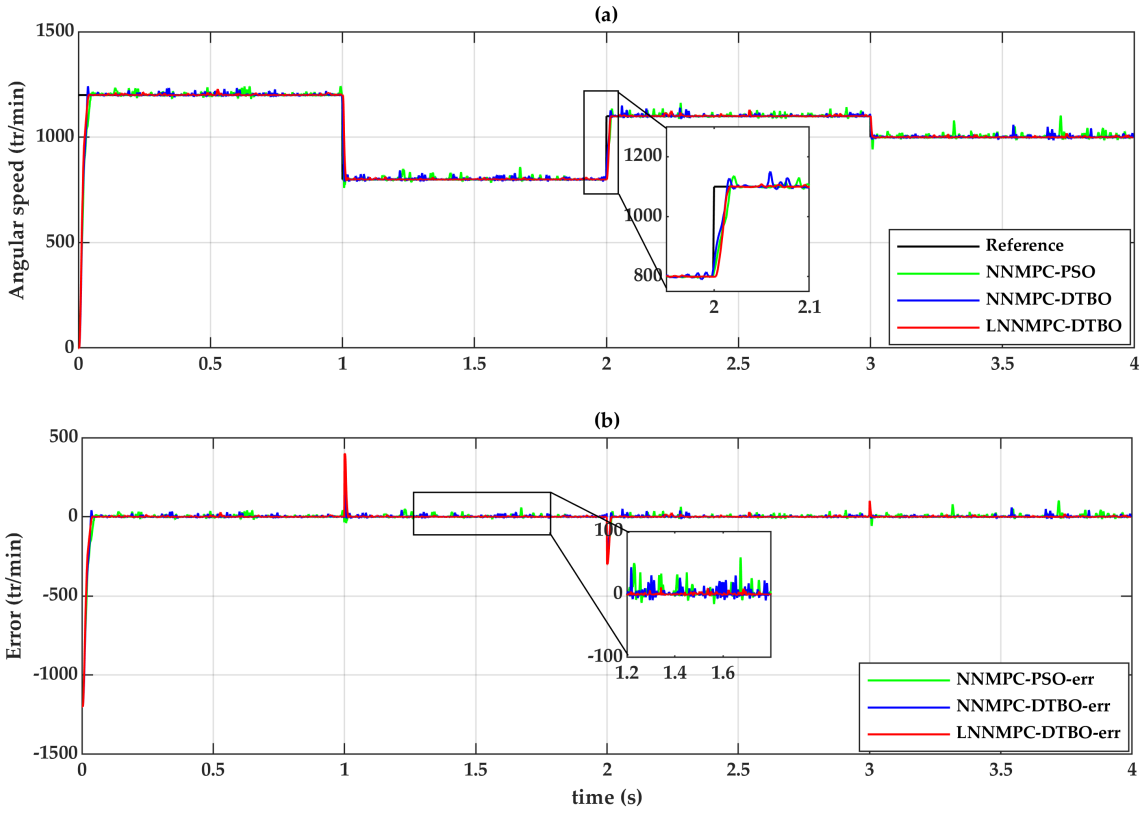

Figure 4.5 – Performance of induction motor speed control with multi-step trajectory, and overshoot constraint: (a) Reference trajectory and actual angular speed, (b) Error between reference trajectory and actual angular speed.

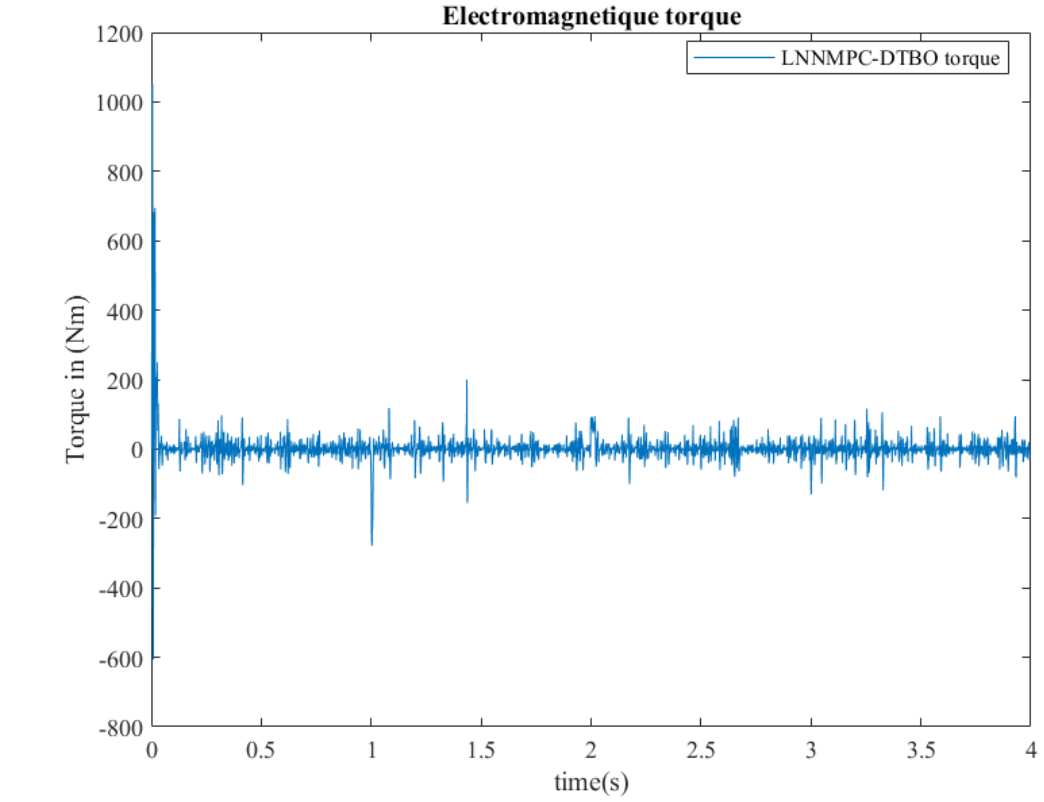

Figure 4.6 – Electomagnatic torque generated using NNMPC-DTBO algorithm .

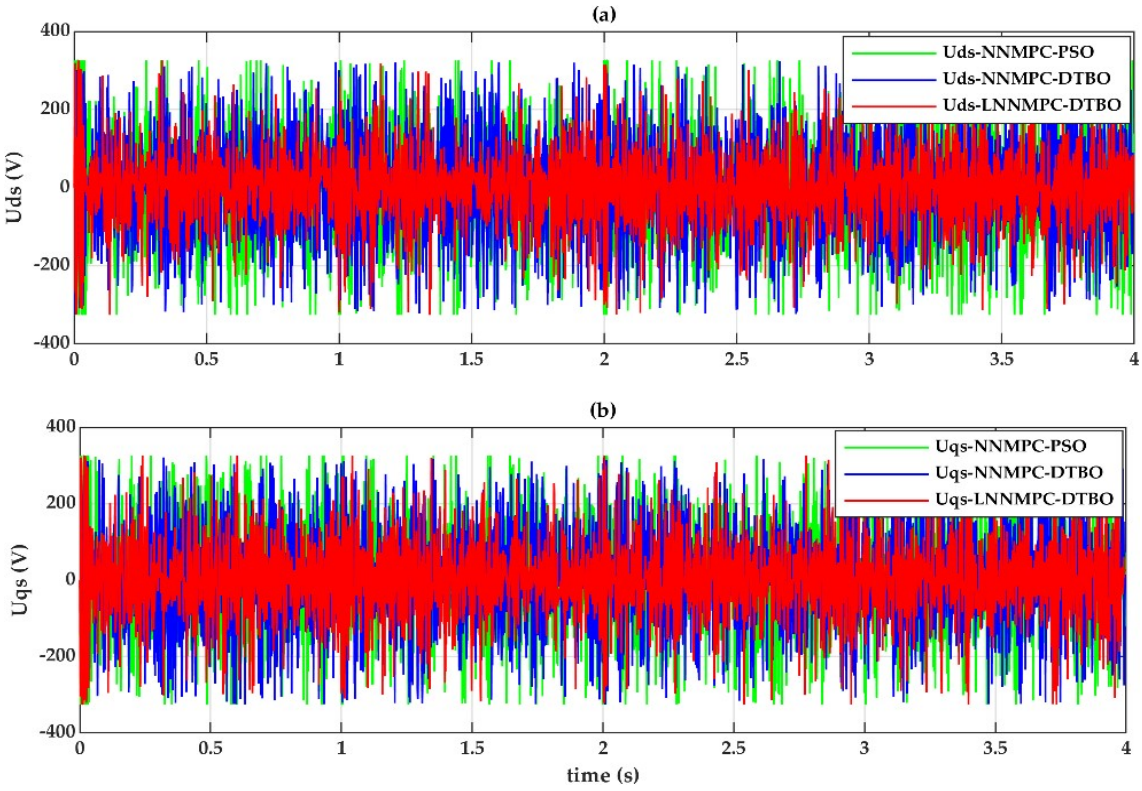

Figure 4.7 – Control effort in the case of multi-step trajectory, with overshoot constraint: (a) Control effort Uds, (b) Control effort Uqs.

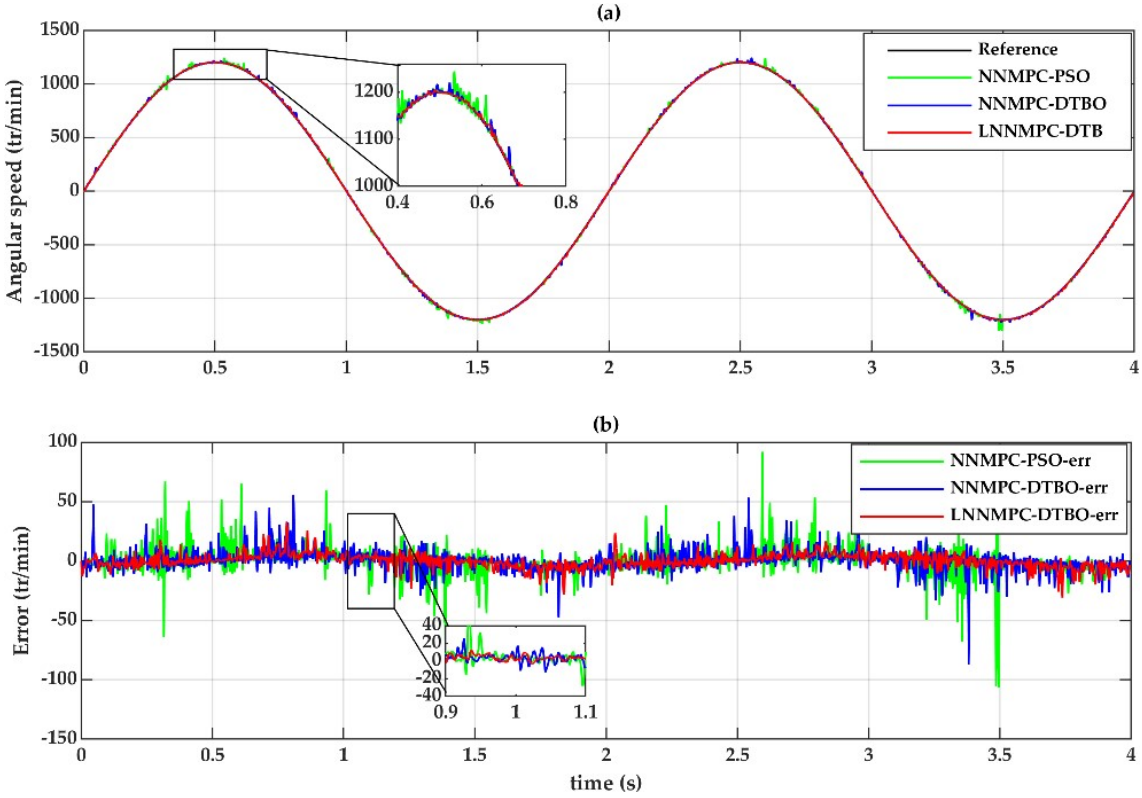

Figure 4.8 – Performance of induction motor speed control with sinusoidal trajectory and overshoot constraint: (a) Reference trajectory and actual angular speed, (b) Error between reference trajectory and actual angular speed.

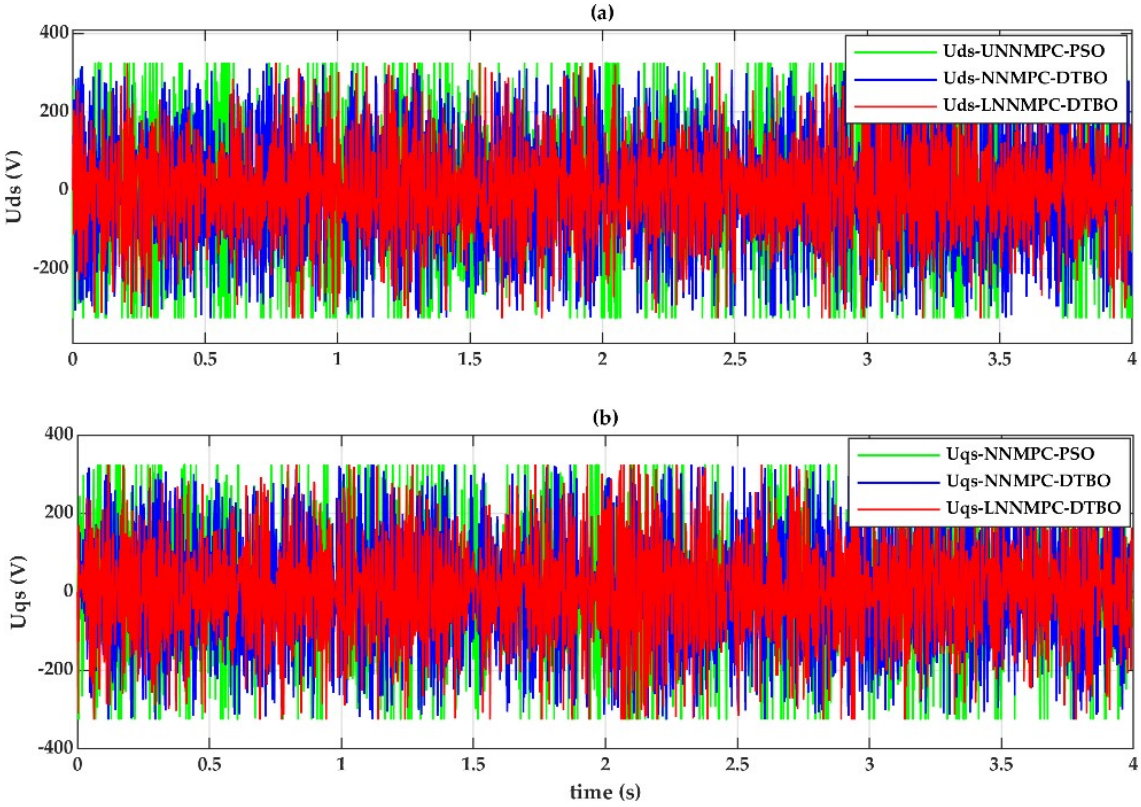

Figure 4.9 – Control efforts in the case of sinusoidal trajectory and overshoot constraint: (a) Control effort Uds, (b) Control effort Uqs.

To evaluate the efficiency of the proposed controller, the values of the MAE, MSE, RMSE, and computing time are calculated and are given in table 4.7 for the multistep and sinusoidal reference trajectories. From The values that are bolded, which represent the best values, it can be concluded that the LNNMPC- DTBO gives the best tracking accuracy with minimal computing time and without overshoot.

Table 4.7 – MAE, MSE, RMSE values with overshoot constraint.

Trajectory	Metric	LNNMPCDTBO	NNMPCDTBO	NNMPCPSO
Multistep trajectory	MAE	8.1908	9.7505	11.7835
	MSE	4.6128e + 03	4.6858e + 03	4.9124e + 03
	RMSE	67.9176	68.4532	70.0886
	Time (ms)	1.334174	1.985236	2.1657730
Sinusoidal trajectory	MAE	4.3815	5.8203	6.3724
	MSE	35.1412	74.8054	115.6211
	RMSE	5.9280	8.6490	10.7527
	Time (ms)	1.148899	1.333881	2.809656

4.4.3.1 Effect of external disturbances and measurement noise

In order to show the LNNMPC-DTBO controller's proficiency to reject external disturbances and compensate for measurement noise, several simulation cases are envisaged.

In the first case, a measurement noise having an amplitude of 10 (rpm), a mean value of 0.4955, and a variance of 0.0854 is added to the system output throughout the simulation as shown in figures 4.10 and 4.11.

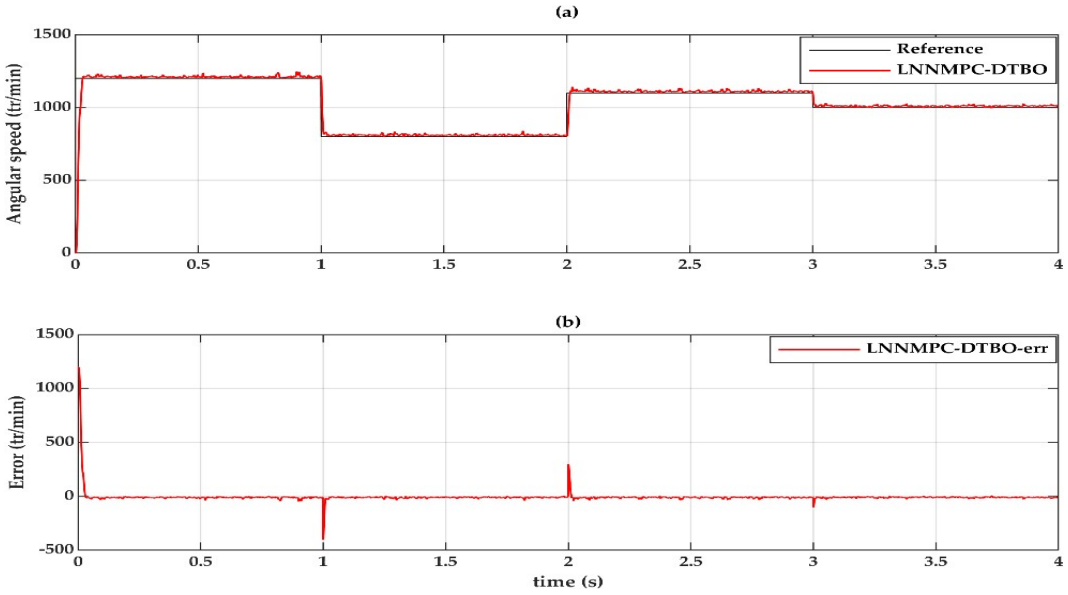

Figure 4.10 – Performance of the induction motor speed control with measurement noise and multistep trajectory: (a) Reference trajectory and actual angular speed, (b) Error between reference trajectory and actual angular speed.

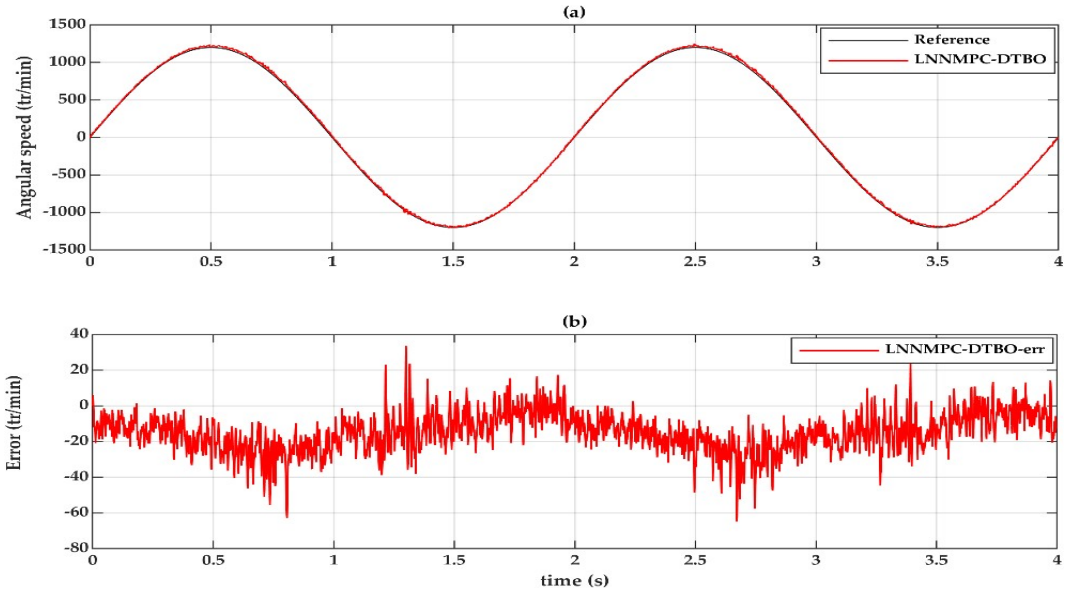

Figure 4.11 – Performance of the induction motor speed control with measurement noise and sinusoidal trajectory, (a) Reference trajectory and actual angular speed, (b) Error between reference trajectory and actual angular speed.

For the second case, two different disturbance types are used with the multistep and sinusoidal reference trajectories, respectively. The results are gathered in table 4.8. A pulse of -25% of the input is added to the control input during the interval $[1.5s, 2.5s]$ as shown in figures 4.12 and 4.13. An output disturbance with an amplitude of $+30\%$ of the output is added during the interval $[1.5s, 2.5s]$ as illustrated by figures 4.14 and 4.15.

Table 4.8 – RMAE, MSE, and MAE values for different disturbances and noise.

References	Disturbances	RMSE	MSE	MAE
Multistep trajectory	-25% input disturbance	68.3632	4.6735e + 03	9.0594
	+30%output disturbance	69.5875	4.8424e + 03	8.3371
	Measurement noise	65.4385	4.2822e + 03	16.4817
Sinusoidal trajectory	-25% input disturbance	6.8910	47.4861	4.8273
	+30%output disturbance	13.1040	171.7138	6.8056
	Measurement noise	18.4433	340.1548	15.7536

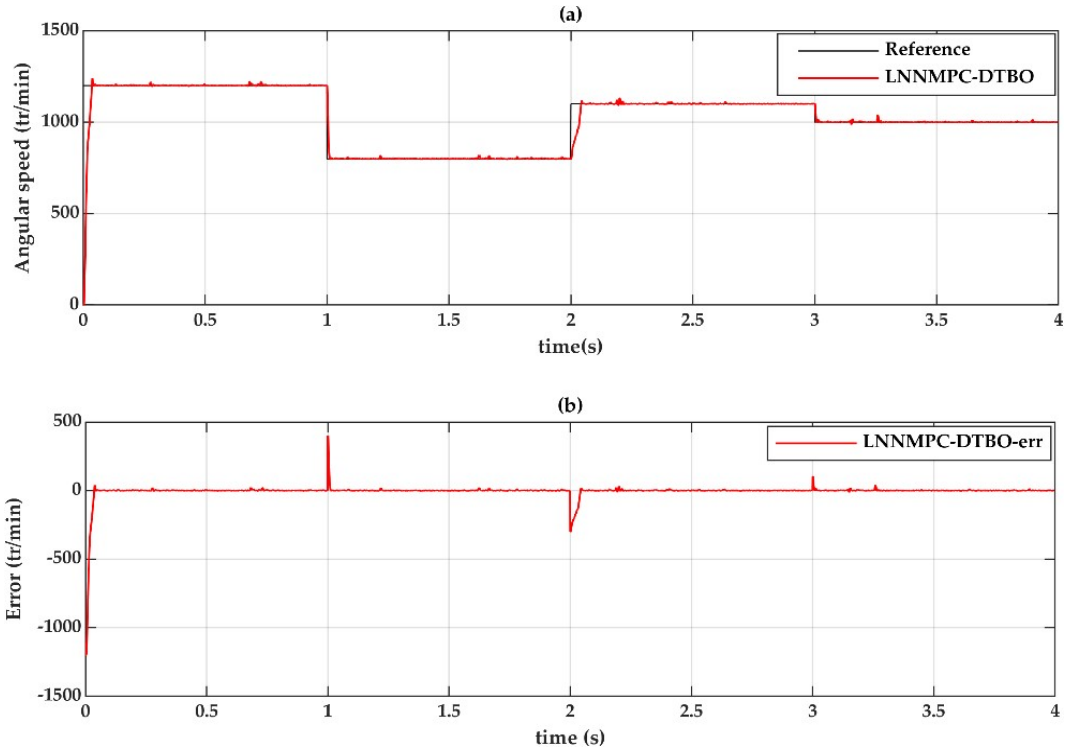

Figure 4.12 – Performance of the induction motor speed control with input disturbance and multistep trajectory: (a) Reference trajectory and actual angular speed, (b) Error between reference trajectory and actual angular speed.

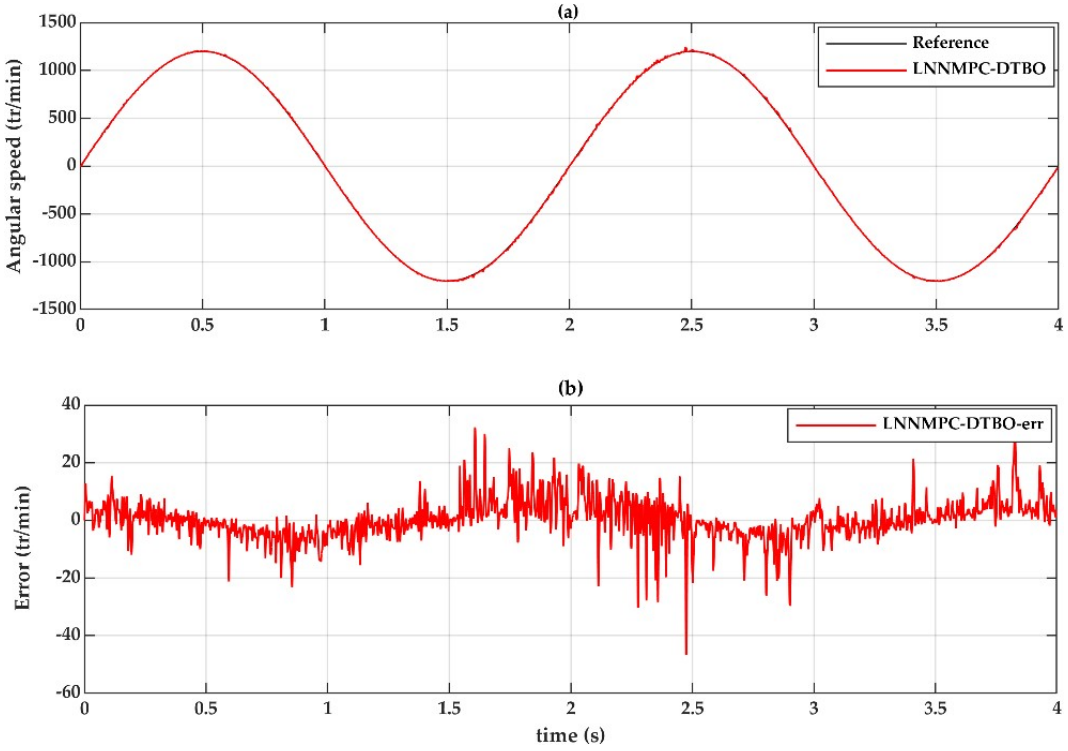

Figure 4.13 – Performance of the induction motor speed control with input disturbance and sinusoidal trajectory: (a) Reference trajectory and actual angular speed, (b) Error between reference trajectory and actual angular speed.

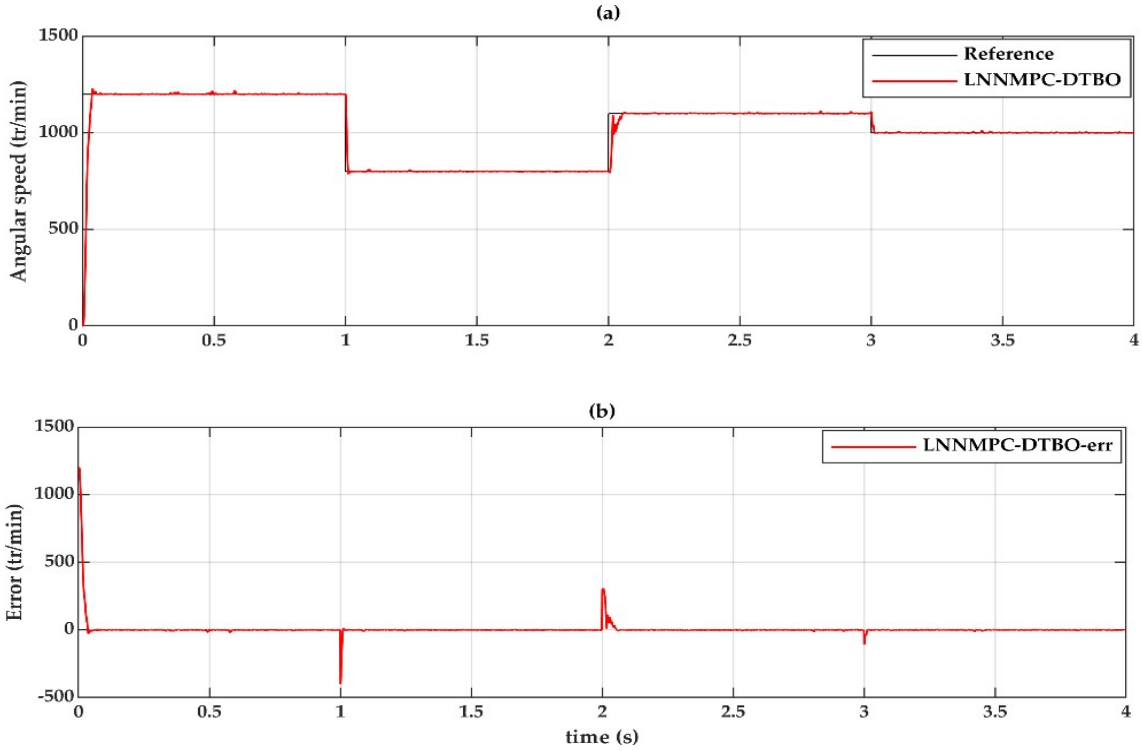

Figure 4.14 – Performance of the induction motor speed control with output disturbance and multistep trajectory: (a) Reference trajectory and actual angular speed, (b) Error between reference trajectory and actual angular speed.

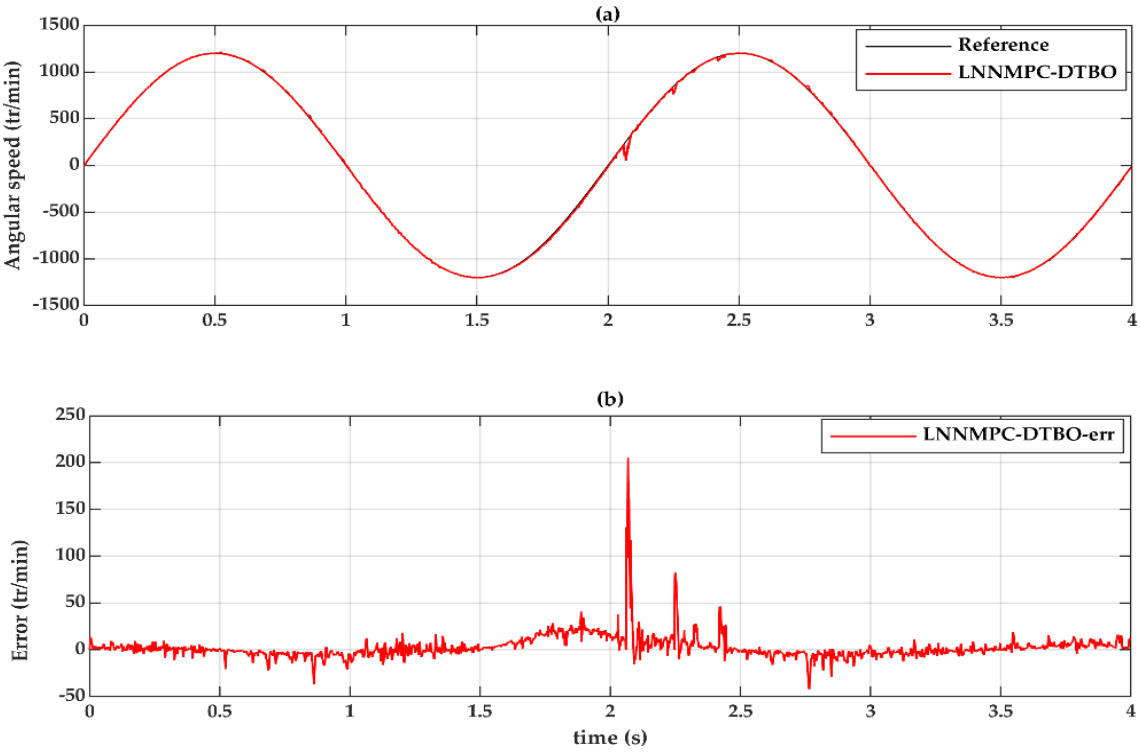

Figure 4.15 – Performance of the induction motor speed control with output disturbance and sinusoidal trajectory: (a) Reference trajectory and actual angular speed, (b) Error between reference trajectory and actual angular speed.

In the third case, different perturbations are applied by changing the values of the load torque T_r as follows: $T_r = 0Nm$, $T_r = 5Nm$, $T_r = 10Nm$, and $T_r = 25Nm$ as illustrated in table 4.9. The system's response is illustrated in figures 4.16 and 4.17 with multistep and sinusoidal reference trajectories, respectively.

Table 4.9 – RMSE, MSE, and MAE values for different load torques.

References	Load torques	RMSE	MSE	MAE
Multistep trajectory	$\boldsymbol{T_r} = \mathbf{0}$Nm	67.3731	4.5391e + 03	7.6975
	$\boldsymbol{T_r} = 5$Nm	68.2399	4.6567e + 03	7.6946
	$T_r = 10$Nm	66.4552	4.4163e + 03	7.2174
	$\boldsymbol{T_r} = \mathbf{25}$ Nm	69.6708	4.8540e + 03	8.0448
Sinusoidal trajectory	$\boldsymbol{T_r} = \mathbf{0}$Nm	6.4061	41.0384	4.6520
	$\boldsymbol{T_r} = 5$Nm	5.4623	29.8365	3.9243
	$T_r = 10$Nm	5.8357	34.0556	4.1151
	$\boldsymbol{T_r} = \mathbf{25}$ Nm	6.1909	38.3274	4.3836

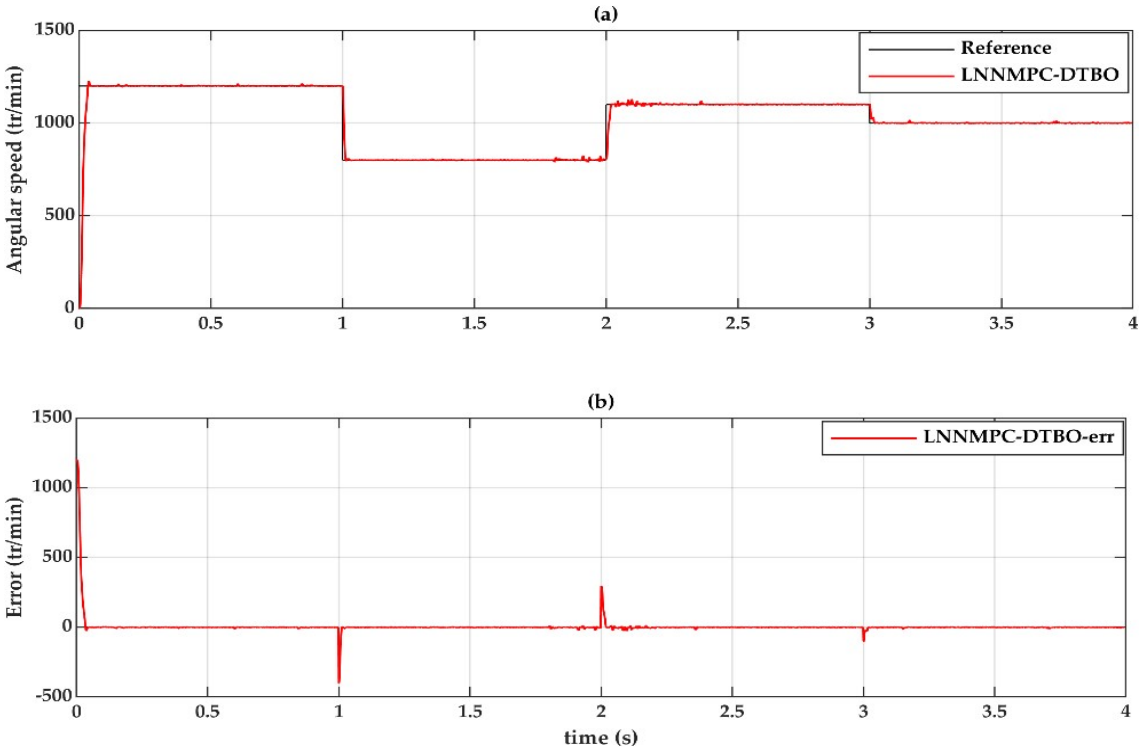

Figure 4.16 – Performance of the induction motor speed control with load torque Tr = 25Nm and multistep trajectory: (a) Reference trajectory and actual angular speed, (b) Error between reference trajectory and actual angular speed.

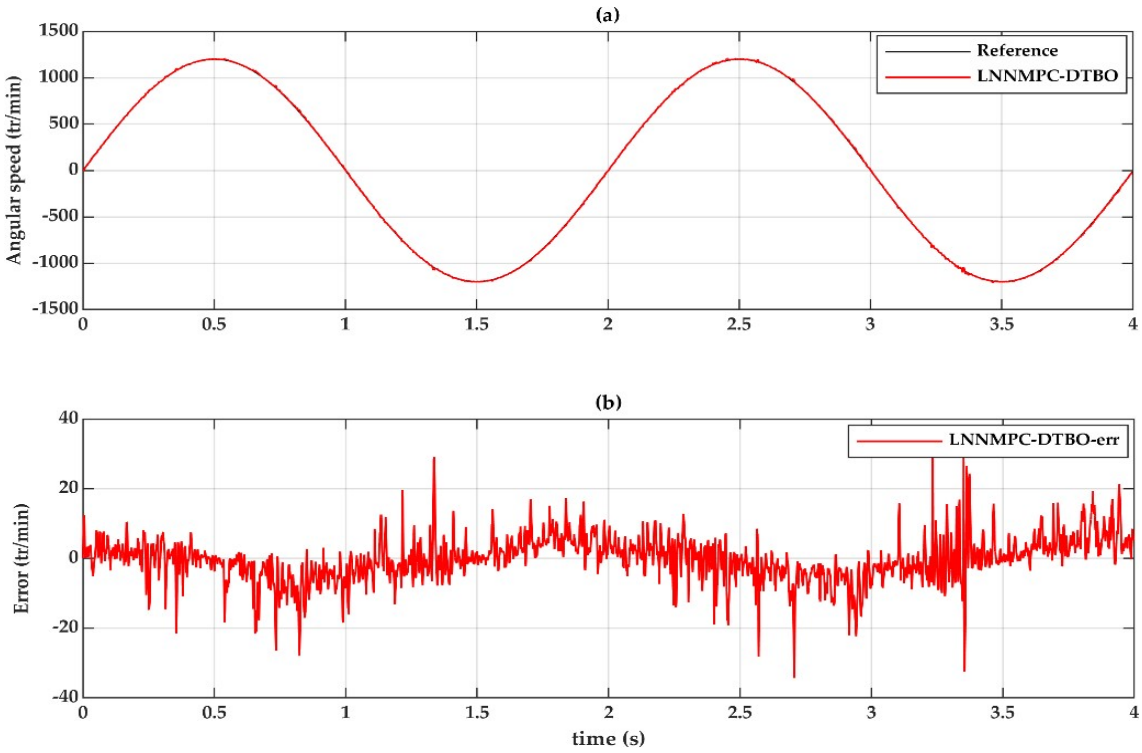

Figure 4.17 – Performance of the induction motor speed control with load torque Tr = 25Nm and sinusoidal trajectory, (a) Reference trajectory and actual angular speed, (b) Error between reference trajectory and actual angular speed.

After applying various disturbances to the induction motor with multistep and sinusoidal desired reference trajectories, and based on the results gathered in tables 4.8 and 4.9, the proposed controller demonstrates its robustness with respect to disturbance rejection and less sensitivity to noise measurement.

4.5 Conclusion

In this work, a new control method called Lyapunov-based neural network model predictive control using metaheuristic optimization approach was suggested. The efficiency of this approach is due to the use of neural network models on the one hand, known for their simplicity and ability to model complex and highly nonlinear systems, and the use of the DTBO optimization algorithm, known for its fast convergence, to optimize the cost function of predictive controller, on the other hand. The stability of the proposed technique was mathematically proved, and the carried out simulations demonstrated that the proposed controller can successfully manage the imposed constraints during the optimization process and offers good results in terms of robustness, accuracy and computation time than the other considered controller. Considering the obtained results, it can be concluded that the LNNMPC-DTBO can be used to control highly nonlinear, multivariable, and constrained systems with fast dynamics.

Conclusion and Future Works

The main purpose of this thesis was to improve the performance of the neural network model predictive control by developing new simple and efficient NNMPC control algorithms based on the use of metaheuristic optimization techniques. In this thesis, four algorithms were designed to improve the performance of the NN-MPC control approach, such as the tracking accuracy, and computational efforts. The effectiveness of these algorithms was validated by their implementation, using Intel(R) core (TM) i7-8665U CPU on a wide range of nonlinear systems, including both single-input, single-output, and multi-input, multi-output systems.

The first controller named improved NNMPC approach based on the TLBO algorithm proposed a modification at the level of the cost function of the NNMPC. It was applied for the speed control of a PMSM motor. The simulation results of this controller proved its efficiency in terms of computation time and accuracy compared to the results provided by the neural network predictive controller based on the TLBO algorithm. The second control algorithm named DTBO-based constrained NNMPC algorithm proposed uses a DTBO metaheuristic algorithm as a novel optimization approach. The effectiveness of this control algorithm was approved by its application to the speed control of an electric vehicle, the obtained results of this technique have demonstrated more efficiency in terms of accuracy and computation time compared to those obtained by the NNMPC based on the PSO approach. The third algorithm called improved NNMPC based on DTBO approach, proposed an enhancement of the second proposed algorithm by modifying its cost function. This algorithm was applied to the position control of a 2-DOF manipulator robot, and the obtained results were compared to those of the PID optimized by the PSO-controller and the neural network model predictive controllers based on the TLBO and DTBO approaches.

The fourth control algorithm suggested a neural network predictive control design using Lyapunov function and metaheuristic optimization. This algorithm integrates a Lyapunov function as constraint in the cost function of the NNMPC and uses a DTBO algorithm as an optimization approach. The efficiency of this approach is due to the use of neural network models on the one hand, known for their simplicity and ability to model complex and highly nonlinear systems, and the use

of the DTBO optimization algorithm, known for its fast convergence, on the other hand. The stability of the proposed technique was mathematically proved. The obtained simulations results of the proposed technique applied to control the speed of induction motor demonstrated that the proposed controller can successfully manage the imposed constraints during the optimization process and offers better results in terms of robustness, accuracy and computation time than the others controller such as optimized PID by PSO algorithm, fuzzy logic controller using TLBO algorithm, NNMPC based on PSO and DTBO algorithms. Considering the obtained results, it can be concluded that the proposed control algorithm can be used to control highly nonlinear, multivariable, and constrained systems with fast dynamics.

There are several possible extensions of this current work; some of the perspectives that can be backed up are as follows:

- Implementing the proposed control approaches on specialized hardware such as FPGAs and Raspberry Pi to achieve high computational efficiency. This is crucial for deploying these controllers in resource-constrained environments.
- Development of a new techniques to apply the NNMPC in experimental configurations with reduced computation time and improved robustness by introducing a new neural network modelling approaches such as Long Short-Term Memory (LSTM) and a new optimization algorithms like multi-objective optimization and reinforcement learning-based algorithms.
- Combining the proposed control approaches with other techniques such as Active disturbances rejection, in order to improve their robustness.

Bibliography

[1] M. Shamseldin, M. A. Ghany, and Y. Hendawey, "Optimal nonlinear pid speed control based on harmony search for an electric vehicle," *Future Engineering Journal*, vol. 2, no. 1, p. 4, 2021. (Cited on pages iv, 43, and 46)

[2] S. V. Raković, "Model predictive control: classical, robust, and stochastic [bookshelf]," *IEEE Control Systems Magazine*, vol. 36, no. 6, pp. 102–105, 2016. (Cited on pages 4 and 10)

[3] J. Richalet, A. Rault, J. Testud, and J. Papon, "Model predictive heuristic control," *Automatica (journal of IFAC)*, vol. 14, no. 5, pp. 413–428, 1978. (Cited on pages 4, 7, and 9)

[4] C. R. Cutler and B. Ramaker, "Dmc—a computer control algorithm," in *AIChE Annual Meeting, Houston*, 1979. (Cited on pages 4 and 7)

[5] Y. Xi and D. Li, *Predictive control: Fundamentals and developments*. John Wiley & Sons, 2019. (Cited on page 4)

[6] K. Holkar and L. M. Waghmare, "An overview of model predictive control," *International Journal of control and automation*, vol. 3, no. 4, pp. 47–63, 2010. (Cited on page 4)

[7] J. H. Lee, "Model predictive control: Review of the three decades of development," *International Journal of Control, Automation and Systems*, vol. 9, pp. 415–424, 2011. (Cited on page 4)

[8] C. E. Garcia, D. M. Prett, and M. Morari, "Model predictive control: Theory and practice—a survey," *Automatica*, vol. 25, no. 3, pp. 335–348, 1989. (Cited on pages 4, 7, and 9)

[9] E. C. Kerrigan and J. M. Maciejowski, "Designing model predictive controllers with prioritised constraints and objectives," in *Proceedings. IEEE International Symposium on Computer Aided Control System Design*. IEEE, 2002, pp. 33–38. (Cited on page 4)

[10] S. J. Qin and T. A. Badgwell, "A survey of industrial model predictive control technology," *Control engineering practice*, vol. 11, no. 7, pp. 733–764, 2003. (Cited on page 4)

[11] J. Richalet, "Industrial applications of model based predictive control," *Automatica*, vol. 29, no. 5, pp. 1251–1274, 1993. (Cited on page 4)

[12] M. Kvasnica, M. Herceg, L. Čirka, and M. Fikar, "Model predictive control of a cstr: A hybrid modeling approach," *Chemical papers*, vol. 64, no. 3, pp. 301–309, 2010. (Cited on page 5)

[13] B. Durmuş, H. Temurtaş, N. Yumuşak, and F. Temurtaş, "A study on industrial robotic manipulator model using model based predictive controls," *Journal of intelligent manufacturing*, vol. 20, pp. 233–241, 2009. (Cited on page 5)

[14] M. Morari, C. E. Garcia, and D. M. Prett, "Model predictive control: Theory and practice," *IFAC Proceedings Volumes*, vol. 21, no. 4, pp. 1–12, 1988. (Cited on page 5)

[15] Y. Xi and D. Li, *Predictive control: Fundamentals and developments*. John Wiley & Sons, 2019. (Cited on page 5)

[16] M. Lundh and M. Molander, "State-space models in model predictive control," *ABB Automation Products AB*, 2002. (Cited on page 6)

[17] X. Tian, P. Gao, J. Yan, and Y. Zheng, "Rbf-arx model-based robust predictive control strategy with one degree-of-freedom," *IEEE Access*, 2024. (Cited on page 6)

[18] M. M. Morato, J. E. Normey-Rico, and O. Sename, "Model predictive control design for linear parameter varying systems: A survey," *Annual Reviews in Control*, vol. 49, pp. 64–80, 2020. (Cited on page 6)

[19] M. Rau and D. Schroder, "Model predictive control with nonlinear state space models," in *7th International Workshop on Advanced Motion Control. Proceedings (Cat. No. 02TH8623)*. IEEE, 2002, pp. 136–141. (Cited on page 6)

[20] Y. Huang, H. H. Lou, J. Gong, and T. F. Edgar, "Fuzzy model predictive control," *IEEE Transactions on Fuzzy Systems*, vol. 8, no. 6, pp. 665–678, 2000. (Cited on page 6)

[21] G. Hou, L. Gong, C. Huang, and J. Zhang, "Fuzzy modeling and fast model predictive control of gas turbine system," *Energy*, vol. 200, p. 117465, 2020. (Cited on pages 6 and 11)

[22] K. Kara, T. E. Missoum, K. E. Hemsas, and M. L. Hadjili, "Control of a robotic manipulator using neural network based predictive control," in *2010 17th IEEE International Conference on Electronics, Circuits and Systems*. IEEE, 2010, pp. 1104–1107. (Cited on page 6)

[23] J. Luo, F. Abdullah, and P. Christofides, "Neural ordinary differential equation modeling for predictive control of nonlinear processes," in *2023 AIChE Annual Meeting*. AIChE, 2023. (Cited on page 6)

[24] S. Konishi, *Introduction to multivariate analysis: linear and nonlinear modeling*. CRC Press, 2014. (Cited on page 6)

[25] M. Schwenzer, M. Ay, T. Bergs, and D. Abel, "Review on model predictive control: An engineering perspective," *The International Journal of Advanced Manufacturing Technology*, vol. 117, no. 5, pp. 1327–1349, 2021. (Cited on pages 7 and 9)

[26] T. Tran, K. Ling, and J. M. Maciejowski, "Economic model predictive control-a review," in *ISARC. Proceedings of the International Symposium on Automation and Robotics in Construction*, vol. 31. IAARC Publications, 2014, p. 1. (Cited on page 7)

[27] M. Y. Lamoudi, M. Alamir, and P. Béguery, "Model predictive control for energy management in buildings part 2: Distributed model predictive control," *IFAC Proceedings Volumes*, vol. 45, no. 17, pp. 226–231, 2012. (Cited on page 8)

[28] Y. Zhang, Z. Zhang, O. Babayomi, and Z. Li, "Weighting factor design techniques for predictive control of power electronics and motor drives," *Symmetry*, vol. 15, no. 6, p. 1219, 2023. (Cited on page 8)

[29] S. V. Raković, "Robust model predictive control," in *Encyclopedia of systems and control*. Springer, 2021, pp. 1965–1975. (Cited on page 8)

[30] D. Mayne and H. Michalska, "An implementable receding horizon controller for stabilization of nonlinear systems," in *29th IEEE Conference on Decision and Control*. IEEE, 1990, pp. 3396–3397. (Cited on page 8)

[31] J. B. Rawlings and K. R. Muske, "The stability of constrained receding horizon control," *IEEE transactions on automatic control*, vol. 38, no. 10, pp. 1512–1516, 1993. (Cited on page 8)

[32] M. Reble and F. Allgöwer, "Design of terminal cost functionals and terminal regions for model predictive control of nonlinear time-delay systems," in *Time*

delay systems: methods, applications and new trends. Springer, 2012, pp. 355–366. (Cited on page 9)

[33] J. Nan, X. Ye, and W. Cao, "Nonlinear model predictive control with terminal cost for autonomous vehicles trajectory follow," *Applied Sciences*, vol. 12, no. 22, p. 11359, 2022. (Cited on page 9)

[34] A. Bemporad and M. Morari, "Robust model predictive control: A survey," in *Robustness in identification and control*. Springer, 2007, pp. 207–226. (Cited on page 9)

[35] Q. Xia, M. Rao, and J. Qian, "Model algorithmic control for paper machines," in *Proceedings of IEEE International Conference on Control and Applications*. IEEE, 1993, pp. 203–208. (Cited on page 9)

[36] C. E. Garcia and A. Morshedi, "Quadratic programming solution of dynamic matrix control (qdmc)," *Chemical Engineering Communications*, vol. 46, no. 1-3, pp. 73–87, 1986. (Cited on page 10)

[37] H. Guolong and S. Youxian, "Study and application of dynamic matrix control in refinery processes," in *2002 IEEE Region 10 Conference on Computers, Communications, Control and Power Engineering. TENCOM'02. Proceedings.*, vol. 3. IEEE, 2002, pp. 1667–1671. (Cited on page 10)

[38] D. W. Clarke, C. Mohtadi, and P. S. Tuffs, "Generalized predictive control—part i. the basic algorithm," *Automatica*, vol. 23, no. 2, pp. 137–148, 1987. (Cited on page 10)

[39] D. W. Clarke, "Application of generalized predictive control to industrial processes," *IEEE Control systems magazine*, vol. 8, no. 2, pp. 49–55, 1988. (Cited on page 10)

[40] L. Kwek, A. Tan, and E. Wong, "Application of generalized predictive control with learning-based disturbance compensator in repetitive operations," in *2016 International Conference on Robotics, Automation and Sciences (ICORAS)*. IEEE, 2016, pp. 1–6. (Cited on page 10)

[41] M. Kamel, M. Burri, and R. Siegwart, "Linear vs nonlinear mpc for trajectory tracking applied to rotary wing micro aerial vehicles," *IFAC-PapersOnLine*, vol. 50, no. 1, pp. 3463–3469, 2017. (Cited on page 10)

[42] F. Allgower, R. Findeisen, Z. K. Nagy *et al.*, "Nonlinear model predictive control: From theory to application," *Journal-Chinese Institute Of Chemical Engineers*, vol. 35, no. 3, pp. 299–316, 2004. (Cited on page 10)

[43] D. FJ III, R. Pearson, and B. Ogunnaike, *Identification and control using Volterra models*. Springer Science & Business Media, 2001. (Cited on page 11)

[44] A. Draeger, E. Sebastian, and R. Horst, "Model predictive control using neural networks," *IEEE Control Systems*, 2020. (Cited on page 11)

[45] S. Piche, B. Sayyar-Rodsari, D. Johnson, and M. Gerules, "Nonlinear model predictive control using neural networks," *IEEE Control Systems Magazine*, vol. 20, no. 3, pp. 53–62, 2000. (Cited on page 11)

[46] P. E. Gill and E. Wong, "Sequential quadratic programming methods," in *Mixed integer nonlinear programming*. Springer, 2011, pp. 147–224. (Cited on page 11)

[47] M. Diehl, H. J. Ferreau, and N. Haverbeke, "Efficient numerical methods for nonlinear mpc and moving horizon estimation problem formulation," *Nonlinear model predictive control*, pp. 391–417. (Cited on page 11)

[48] C. Kirches, "The direct multiple shooting method for optimal control," in *Fast Numerical Methods for Mixed-integer Nonlinear Model-Predictive control*. Springer, 2011, pp. 13–29. (Cited on page 11)

[49] M. Kelly, "An introduction to trajectory optimization: How to do your own direct collocation," *SIAM review*, vol. 59, no. 4, pp. 849–904, 2017. (Cited on page 12)

[50] G. Sánchez, M. Murillo, L. Genzelis, N. Deniz, and L. Giovanini, "Mpc for nonlinear systems: A comparative review of discretization methods," in *2017 XVII Workshop on Information Processing and Control (RPIC)*. IEEE, 2017, pp. 1–6. (Cited on page 12)

[51] C. Rösmann, A. Makarow, and T. Bertram, "Stabilising quasi-time-optimal nonlinear model predictive control with variable discretisation," *International Journal of Control*, vol. 95, no. 10, pp. 2682–2694, 2022. (Cited on page 12)

[52] T. Parisini and R. Zoppoli, "A receding-horizon regulator for nonlinear systems and a neural approximation," *Automatica*, vol. 31, no. 10, pp. 1443–1451, 1995. (Cited on page 12)

[53] J. G. Ortega and E. Camacho, "Mobile robot navigation in a partially structured static environment, using neural predictive control," *Control Engineering Practice*, vol. 4, no. 12, pp. 1669–1679, 1996. (Cited on page 12)

[54] B. Karg and S. Lucia, "Efficient representation and approximation of model predictive control laws via deep learning," *IEEE transactions on cybernetics*, vol. 50, no. 9, pp. 3866–3878, 2020. (Cited on page 12)

[55] K. J. Chan, J. A. Paulson, and A. Mesbah, "Deep learning-based approximate nonlinear model predictive control with offset-free tracking for embedded applications," in *2021 American Control Conference (ACC)*. IEEE, 2021, pp. 3475–3481. (Cited on page 12)

[56] C. Gonzalez, H. Asadi, L. Kooijman, and C. P. Lim, "Neural networks for fast optimisation in model predictive control: A review," *arXiv preprint arXiv:2309.02668*, 2023. (Cited on pages 12 and 13)

[57] Y. Li, K. Hua, and Y. Cao, "Using stochastic programming to train neural network approximation of nonlinear mpc laws," *Automatica*, vol. 146, p. 110665, 2022. (Cited on page 13)

[58] M. Hertneck, J. Köhler, S. Trimpe, and F. Allgöwer, "Learning an approximate model predictive controller with guarantees," *IEEE Control Systems Letters*, vol. 2, no. 3, pp. 543–548, 2018. (Cited on page 13)

[59] W. Daosud, P. Kittisupakorn, M. Fikar, S. Lucia, and R. Paulen, "Efficient robust nonlinear model predictive control via approximate multi-stage programming: A neural networks based approach," in *Computer Aided Chemical Engineering*. Elsevier, 2019, vol. 46, pp. 1261–1266. (Cited on page 13)

[60] S. Desale, A. Rasool, S. Andhale, and P. Rane, "Heuristic and meta-heuristic algorithms and their relevance to the real world: a survey," *Int. J. Comput. Eng. Res. Trends*, vol. 351, no. 5, pp. 2349–7084, 2015. (Cited on pages 13 and 18)

[61] V. Tomar, M. Bansal, and P. Singh, "Metaheuristic algorithms for optimization: A brief review," *Engineering Proceedings*, vol. 59, no. 1, p. 238, 2024. (Cited on page 13)

[62] X. Du, K. K. Tan, and K. K. K. Htet, "Development of a genetic algorithm based nonlinear model predictive scheme," in *2015 10th Asian Control Conference (ASCC)*. IEEE, 2015, pp. 1–6. (Cited on pages 13 and 19)

[63] G. H. Negri, V. H. Preuss, M. S. Cavalca, and J. de Oliveira, "Differential evolution optimization applied in multivariate nonlinear model-based predictive control," in *2015 Latin America Congress on Computational Intelligence (LA-CCI)*. IEEE, 2015, pp. 1–6. (Cited on page 13)

[64] J. Zietkiewicz, P. Kozierski, and W. Giernacki, "Particle swarm optimisation in nonlinear model predictive control; comprehensive simulation study for two selected problems," *International Journal of Control*, vol. 94, no. 10, pp. 2623–2639, 2021. (Cited on pages 13 and 19)

[65] O. A. Sahed, K. Kara, and A. Benyoucef, "Artificial bee colony-based predictive control for non-linear systems," *Transactions of the Institute of Measurement and Control*, vol. 37, no. 6, pp. 780–792, 2015. (Cited on page 13)

[66] M. Benrabah, K. Kara, O. AitSahed, and M. L. Hadjili, "Constrained nonlinear predictive control using neural networks and teaching–learning-based optimization," *Journal of Control, Automation and Electrical Systems*, vol. 32, no. 5, pp. 1228–1243, 2021. (Cited on pages 13, 16, 19, and 27)

[67] A. Aouaichia, K. Kara, M. Benrabah, and M. L. Hadjili, "Constrained neural network model predictive controller based on archimedes optimization algorithm with application to robot manipulators," *Journal of Control, Automation and Electrical Systems*, vol. 34, no. 6, pp. 1159–1178, 2023. (Cited on pages 13, 14, and 16)

[68] C. Stiti, M. Benrabah, A. Aouaichia, A. Oubelaid, M. Bajaj, M. B. Tuka, and K. Kara, "Lyapunov-based neural network model predictive control using metaheuristic optimization approach," *Scientific Reports*, vol. 14, no. 1, p. 18760, 2024. (Cited on pages 13, 16, and 29)

[69] T. Oliveira Cabral, A. Bagheri, and D. B. Pourkargar, "Learning-based model reduction and predictive control of an ammonia synthesis process," *Industrial & Engineering Chemistry Research*, vol. 63, no. 23, pp. 10 325–10 342, 2024. (Cited on page 13)

[70] K. S. Narendra and K. Parthasarathy, "Identification and control of dynamical systems using neural networks," *IEEE Transactions on neural networks*, vol. 1, no. 1, pp. 4–27, 1990. (Cited on page 14)

[71] K. S. Narendra and K. Parthasarathy, "Neural networks and dynamical systems," *International Journal of Approximate Reasoning*, vol. 6, no. 2, pp. 109–131, 1992. (Cited on page 14)

[72] A. U. Levin and K. S. Narendra, "Control of nonlinear dynamical systems using neural networks. ii. observability, identification, and control," *IEEE transactions on neural networks*, vol. 7, no. 1, pp. 30–42, 1996. (Cited on page 14)

[73] A. K. Paul, A. Prasad, and A. Kumar, "Review on artificial neural network and its application in the field of engineering," *J. Mech. Eng. Prakash*, vol. 1, no. 1, pp. 53–61, 2022. (Cited on page 14)

[74] D. Karimanzira and T. Rauschenbach, "Deep learning based model predictive control for a reverse osmosis desalination plant," *Journal of Applied Mathematics and Physics*, vol. 8, no. 12, pp. 2713–2731, 2020. (Cited on page 14)

[75] H. Wang, "A study on the model predictive control based on convolutional neural network," in *Journal of Physics: Conference Series*, vol. 2580, no. 1. IOP Publishing, 2023, p. 012024. (Cited on page 14)

[76] T. Xiao, Z. Wu, P. D. Christofides, A. Armaou, and D. Ni, "Recurrent neural-network-based model predictive control of a plasma etch process," *Industrial & Engineering Chemistry Research*, vol. 61, no. 1, pp. 638–652, 2021. (Cited on page 14)

[77] E. Terzi, F. Bonassi, M. Farina, and R. Scattolini, "Model predictive control design for dynamical systems learned by long short-term memory networks," *arXiv preprint arXiv:1910.04024*, 2019. (Cited on page 14)

[78] S. Kuninti and S. Rooban, "Backpropagation algorithm and its hardware implementations: A review," in *Journal of Physics: Conference Series*, vol. 1804, no. 1. IOP Publishing, 2021, p. 012169. (Cited on page 14)

[79] T. Salzmann, E. Kaufmann, J. Arrizabalaga, M. Pavone, D. Scaramuzza, and M. Ryll, "Real-time neural mpc: Deep learning model predictive control for quadrotors and agile robotic platforms," *IEEE Robotics and Automation Letters*, vol. 8, no. 4, pp. 2397–2404, 2023. (Cited on page 14)

[80] F. Bonassi, C. F. O. da Silva, and R. Scattolini, "Nonlinear mpc for offset-free tracking of systems learned by gru neural networks," *IFAC-PapersOnLine*, vol. 54, no. 14, pp. 54–59, 2021. (Cited on page 14)

[81] X. Yang, K. Wang, J. Kim, and K.-B. Park, "Artificial neural network-based fcs-mpc for three-level inverters," *Journal of Power Electronics*, vol. 22, no. 12, pp. 2158–2165, 2022. (Cited on page 14)

[82] S. de Jongh, S. Steinle, A. Hlawatsch, F. Mueller, M. Suriyah, and T. Leibfried, "Neural predictive control for the optimization of smart grid flexibility schedules," in *2021 56th International Universities Power Engineering Conference (UPEC)*. IEEE, 2021, pp. 1–6. (Cited on page 14)

[83] L. Wu, X. Yin, L. Pan, and J. Liu, "Smart energy management: Process structure-based hybrid neural networks for optimal scheduling and economic predictive control in integrated systems," *Applied Energy*, vol. 380, p. 124965, 2025. (Cited on page 14)

[84] Y. Wang, S. Li, and B. Zhang, "General regression neural network-based data-driven model-free predictive functional control for a class of discrete-time nonlinear systems," *Nonlinear Dynamics*, vol. 107, no. 1, pp. 953–966, 2022. (Cited on page 14)

[85] H. Hose, J. Köhler, M. N. Zeilinger, and S. Trimpe, "Approximate nonlinear model predictive control with safety-augmented neural networks," *arXiv preprint arXiv:2304.09575*, 2023. (Cited on page 14)

[86] S. Mamedov, R. Reiter, S. M. B. Azad, R. Viljoen, J. Boedecker, M. Diehl, and J. Swevers, "Safe imitation learning of nonlinear model predictive control for flexible robots," in *2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2024, pp. 3613–3619. (Cited on page 15)

[87] F. Mohammad and Y.-C. Kim, "Energy load forecasting model based on deep neural networks for smart grids," *International Journal of System Assurance Engineering and Management*, vol. 11, no. 4, pp. 824–834, 2020. (Cited on page 15)

[88] D. Li, K. Lu, Y. Cheng, H. Wu, H. Handroos, S. Yang, Y. Zhang, and H. Pan, "Hybrid nonlinear model predictive motion control of a heavy-duty bionic caterpillar-like robot," *Journal of Bionic Engineering*, vol. 21, no. 5, pp. 2232–2246, 2024. (Cited on page 15)

[89] Z. Wu, D. Rincon, Q. Gu, and P. D. Christofides, "Statistical machine learning in model predictive control of nonlinear processes," *Mathematics*, vol. 9, no. 16, p. 1912, 2021. (Cited on page 15)

[90] Z. Zhang, Z. Wu, D. Rincon, and P. D. Christofides, "Real-time optimization and control of nonlinear processes using machine learning," *Mathematics*, vol. 7, no. 10, p. 890, 2019. (Cited on page 15)

[91] F. A. Potra and S. J. Wright, "Interior-point methods," *Journal of computational and applied mathematics*, vol. 124, no. 1-2, pp. 281–302, 2000. (Cited on page 15)

[92] A. F. Izmailov and M. V. Solodov, "Newton-type methods: A broader view," *Journal of Optimization Theory and Applications*, vol. 164, pp. 577–620, 2015. (Cited on page 15)

[93] A. Jaber, R. Younes, P. Lafon, and J. Khoder, "A review on multi-objective mixed-integer non-linear optimization programming methods," *Eng*, vol. 5, no. 3, pp. 1961–1979, 2024. (Cited on page 16)

[94] C. Stiti, K. Kara, M. Benrabah, and A. Aouaichia, "Neural network model predictive control based on pso approach: Applied to dc motor," in *2023 2nd International Conference on Electronics, Energy and Measurement (IC2EM)*, vol. 1. IEEE, 2023, pp. 1–6. (Cited on page 16)

[95] J. H. Conway and G. Polya, *How to Solve It: A New Aspect of Mathematical Method*. Princeton university press, 1945. (Cited on page 17)

[96] M. Caserta and S. Voß, "Metaheuristics: intelligent problem solving," in *Matheuristics: Hybridizing metaheuristics and mathematical programming*. Springer, 2009, pp. 1–38. (Cited on page 17)

[97] A. Hertz and M. Mittaz, "Heuristic algorithms," in *Arc Routing: Theory, Solutions and Applications*. Springer, 2000, pp. 327–386. (Cited on page 17)

[98] M. Sevaux, F. Glover, and K. Sörensen, "A history of metaheuristics," *Handbook of Heuristics*, pp. 791–808, 2018. (Cited on page 18)

[99] T. Guilmeau, E. Chouzenoux, and V. Elvira, "Simulated annealing: A review and a new scheme," in *2021 IEEE statistical signal processing workshop (SSP)*. IEEE, 2021, pp. 101–105. (Cited on page 19)

[100] P. K. Mishra and M. Bhusry, "Artificial immune system: State of the art approach," *International Journal of Computer Applications*, vol. 120, no. 20, 2015. (Cited on page 19)

[101] S. Cheng, Q. Qin, J. Chen, and Y. Shi, "Brain storm optimization algorithm: a review," *Artificial Intelligence Review*, vol. 46, no. 4, pp. 445–458, 2016. (Cited on page 19)

[102] A. W. Mohamed, A. A. Hadi, and A. K. Mohamed, "Gaining-sharing knowledge based algorithm for solving optimization problems: a novel nature-inspired algorithm," *International Journal of Machine Learning and Cybernetics*, vol. 11, no. 7, pp. 1501–1529, 2020. (Cited on page 19)

[103] M. Shehab, R. Sihwail, M. Daoud, H. Al-Mimi, and L. Abualigah, "Nature-inspired metaheuristic algorithms: a comprehensive review," 2024. (Cited on page 19)

[104] M. Gendreau and J.-Y. Potvin, "Tabu search," *Search methodologies: introductory tutorials in optimization and decision support techniques*, pp. 165–186, 2005. (Cited on page 19)

[105] M. G. Resende and C. C. Ribeiro, "Greedy randomized adaptive search procedures: Advances, hybridizations, and applications," *Handbook of metaheuristics*, pp. 283–319, 2010. (Cited on page 19)

[106] P. Hansen and N. Mladenović, "Variable neighborhood search: Principles and applications," *European journal of operational research*, vol. 130, no. 3, pp. 449–467, 2001. (Cited on page 19)

[107] H. Stegherr, M. Heider, and J. Hähner, "Classifying metaheuristics: Towards a unified multi-level classification system," *Natural Computing*, vol. 21, no. 2, pp. 155–171, 2022. (Cited on pages 20 and 21)

[108] M. Dorigo and T. Stützle, "Ant colony optimization: overview and recent advances," *Handbook of metaheuristics*, pp. 311–351, 2018. (Cited on page 20)

[109] S. Andradóttir, "A review of random search methods," *Handbook of simulation optimization*, pp. 277–292, 2014. (Cited on page 20)

[110] V. Yannibelli and A. Amandi, "A deterministic crowding evolutionary algorithm to form learning teams in a collaborative learning context," *Expert systems with applications*, vol. 39, no. 10, pp. 8584–8592, 2012. (Cited on page 20)

[111] A. García, "Greedy algorithms: a review and open problems," *Journal of Inequalities and Applications*, vol. 2025, no. 1, p. 11, 2025. (Cited on page 20)

[112] S. M. Almufti, A. A. Shaban, Z. A. Ali, R. I. Ali, and J. D. Fuente, "Overview of metaheuristic algorithms," *Polaris Global Journal of Scholarly Research and Trends*, vol. 2, no. 2, pp. 10–32, 2023. (Cited on page 20)

[113] S. Chinnasamy, M. Ramachandran, M. Amudha, and K. Ramu, "A review on hill climbing optimization methodology," *Recent Trends in Management and Commerce*, vol. 3, no. 1, pp. 1–7, 2022. (Cited on page 20)

[114] M. M. Mafarja and S. Mirjalili, "Hybrid whale optimization algorithm with simulated annealing for feature selection," *Neurocomputing*, vol. 260, pp. 302–312, 2017. (Cited on page 21)

[115] Y. Zhang, M. Zhang, and Y.-c. Liang, "A hybrid aco/pso algorithm and its applications," *International Journal of Modelling, Identification and Control*, vol. 8, no. 4, pp. 309–316, 2009. (Cited on page 21)

[116] Š. Hubálovskỳ, M. Hubálovská, and I. Matoušová, "A new hybrid particle swarm optimization–teaching–learning-based optimization for solving optimization problems," *Biomimetics*, vol. 9, no. 1, p. 8, 2023. (Cited on page 21)

[117] J. O. Agushaka and A. E. Ezugwu, "Initialisation approaches for population-based metaheuristic algorithms: a comprehensive review," *Applied Sciences*, vol. 12, no. 2, p. 896, 2022. (Cited on page 22)

[118] M. N. M. Salleh, K. Hussain, S. Cheng, Y. Shi, A. Muhammad, G. Ullah, and R. Naseem, "Exploration and exploitation measurement in swarm-based metaheuristic algorithms: An empirical analysis," in *Recent Advances on Soft Computing and Data Mining: Proceedings of the Third International Conference on Soft Computing and Data Mining (SCDM 2018), Johor, Malaysia, February 06-07, 2018*. Springer, 2018, pp. 24–32. (Cited on page 23)

[119] V. Osuna-Enciso, E. Cuevas, and B. M. Castañeda, "A diversity metric for population-based metaheuristic algorithms," *Information Sciences*, vol. 586, pp. 192–208, 2022. (Cited on page 23)

[120] T. Kawachi, J.-I. Kushida, A. Hara, and T. Takahama, "Efficient constraint handling based on the adaptive penalty method with balancing the objective function value and the constraint violation," in *2019 IEEE 11th International Workshop on Computational Intelligence and Applications (IWCIA)*. IEEE, 2019, pp. 121–128. (Cited on page 26)

[121] T. P. Runarsson and X. Yao, "Stochastic ranking for constrained evolutionary optimization," *IEEE Transactions on evolutionary computation*, vol. 4, no. 3, pp. 284–294, 2000. (Cited on page 26)

[122] T. Takahama and S. Sakai, "Constrained optimization by ε constrained particle swarm optimizer with ε-level control," in *Soft Computing as Transdisciplinary Science and Technology: Proceedings of the fourth IEEE International Workshop WSTST'05*. Springer, 2005, pp. 1019–1029. (Cited on page 26)

[123] T. P. Runarsson and X. Yao, "Stochastic ranking for constrained evolutionary optimization," *IEEE Transactions on evolutionary computation*, vol. 4, no. 3, pp. 284–294, 2000. (Cited on page 26)

[124] E. H. Houssein, M. K. Saeed, G. Hu, and M. M. Al-Sayed, "Metaheuristics for solving global and engineering optimization problems: review, applications, open issues and challenges," *Archives of Computational Methods in Engineering*, pp. 1–35, 2024. (Cited on page 33)

[125] K. Rajwar, K. Deep, and S. Das, "An exhaustive review of the metaheuristic algorithms for search and optimization: taxonomy, applications, and open challenges," *Artificial Intelligence Review*, vol. 56, no. 11, pp. 13 187–13 257, 2023. (Cited on page 33)

[126] C. A. C. Coello, "Theoretical and numerical constraint-handling techniques used with evolutionary algorithms: a survey of the state of the art," *Computer methods in applied mechanics and engineering*, vol. 191, no. 11-12, pp. 1245–1287, 2002. (Cited on page 36)

[127] Y. Wei, Y. Wei, Y. Gao, H. Qi, X. Guo, M. Li, and D. Zhang, "A variable prediction horizon self-tuning method for nonlinear model predictive speed control on pmsm rotor position system," *IEEE Access*, vol. 9, pp. 78 812–78 822, 2021. (Cited on page 36)

[128] F. Lin, *Robust control design: an optimal control approach*. John Wiley & Sons, 2007. (Cited on pages 51 and 52)

[129] P. Mhaskar, N. H. El-Farra, and P. D. Christofides, "Stabilization of nonlinear systems with state and control constraints using lyapunov-based predictive control," *Systems & Control Letters*, vol. 55, no. 8, pp. 650–659, 2006. (Cited on page 61)

[130] J. Luo, Y. Li, P. Liu, S. Ye, R. Feng, and J. Wang, "Lyapunov based nonlinear model predictive control of wind power generation system with external disturbances," *IEEE Access*, vol. 12, pp. 5103–5116, 2024. (Cited on pages 61 and 63)

[131] B. Babaghorbani, M. T. Beheshti, and H. A. Talebi, "A lyapunov-based model predictive control strategy in a permanent magnet synchronous generator wind turbine," *International Journal of Electrical Power & Energy Systems*, vol. 130, p. 106972, 2021. (Cited on page 61)

[132] R. Wang and J. Bao, "A differential lyapunov-based tube mpc approach for continuous-time nonlinear processes," *Journal of Process Control*, vol. 83, pp. 155–163, 2019. (Cited on page 63)

[133] P. Mhaskar, N. H. El-Farra, and P. D. Christofides, "Stabilization of nonlinear systems with state and control constraints using lyapunov-based predictive control," *Systems & Control Letters*, vol. 55, no. 8, pp. 650–659, 2006. (Cited on page 63)

[134] R. Wang and J. Bao, "A differential lyapunov-based tube mpc approach for continuous-time nonlinear processes," *Journal of Process Control*, vol. 83, pp. 155–163, 2019. (Cited on page 63)

[135] B. Ozpineci and L. M. Tolbert, "Simulink implementation of induction machine model-a modular approach," in *IEEE International Electric Machines and Drives Conference, 2003. IEMDC'03.*, vol. 2. Ieee, 2003, pp. 728–734. (Cited on pages 66 and 67)