Université Mohamed Khider de Biskra
Faculty of Sciences and Technology
Departement of Electrical Engineering

MÉMOIRE DE MASTER

Sciences et Technologies
Electrical Engineering
Electric command

Réf. : Entrez la référence du document

Présenté et soutenu par :

Presented and supported by: Guendouz Radjia and Mraoui Darine

Le : lundi 2 juin 2025

Optimization of PID controller with Metaheuristic algorithms for BLDC motor

---

Jury :

Mme.	Mahdad Belkacem	**PF**	Université de Biskra	President
Mme.	Rahoua Naima	**MCB**	Université de Biskra	Supervisor
Mme.	Benyahia Naima	**MAA**	Université de Biskra	Examiner

Année universitaire : Choisissez une année

University Mohamed Khider of Biskra
Faculty of Sciences and Technology
Departement of Electrical Engineering

MASTER'S THESIS

Sciences and Technology
Electrical Engineering
Electric command

Réf. : Entrez la référence du document

Optimization of PID controller with Metaheuristic algorithms for BLDC motor

On : Thursday, jun 26, 2025

Presented by :

Guendouz Radjia

Mraoui darine

Avis favorable de l'encadreur

Mrs,Rahoua Naima

Favorable opinion of the President of Jury:

Mrs. Mahdad Belkacem

Cachet et signature

Acknowledgement

At the conclusion of this joint work, we would like to express our deepest gratitude to everyone who contributed to the success of this research, which is not the result of an individual effort, but a true collaboration and partnership.

First, we sincerely thank our supervisor,Mrs **Naima Rahoua,** who did more than just teach us. She planted in our minds the spirit of research and excellence, guiding us throughout our academic journey.

We also highly appreciate all our professors, whose knowledge and dedication laid the scientific foundations in our minds, enriched our understanding, and motivated us to strive for the best.

We are deeply grateful to our families, who have been a constant source of support, patience, and encouragement. Their role was essential in pushing us toward success.

We warmly thank all the laboratory staff who provided technical assistance and guidance during the work,We especially thank Professor **Mimoune**, who played a special role and provided us with valuable support in the laboratory,as well as the universitie and institutions that opened their doors and provided the necessary materials and resources to complete this research.

Finally, we would like to offer special thanks to our colleague, PhD student **Fares Bettahar**,for his valuable advice and support, who was a loyal companion throughout this journey. This thesis is the result of a collective effort, and we are proud of what has been achieved through the cooperation and teamwork of everyone involved.

Dedication

*T*o our families**Guendouz,Mraoui**, for their endless love, patience, and belief in us even when we doubted ourselves. Your sacrifices are the roots of this success.

To our mothers, **Aicha Bouziane** and **Siham Radjab**, whose love carried us through the hardest moments, and whose strength gave us the courage to continue. You are the silent heroes behind this work.

To Mrs **Naima Rahoua**, for her guidance, trust, and the knowledge she generously shared throughout this journey.

To our thesis partner we walked this road together, step by step. This work is ours, in effort, in thought, and in heart.

To my sisters, **Maria** and **Rania**, for their quiet and steady support behind the scenes, and for every word of encouragement and every moment of care that helped me keep going.

To our laptop the quiet witness to every idea, every correction, every moment of frustration and breakthrough. You turned our thoughts into a thesis, our chaos into structure, our late nights into pages that now carry meaning.

And to ourselves for believing in our goals when things were unclear, for refusing to give up, and for proving that perseverance can turn effort into achievement.

This work is not only a product of knowledge, but of trust, struggle, and quiet persistence. To all who helped carry the dream this is for you.

Summary

The objective of this project is to control the speed of a **Brushless DC (BLDC)** motor using a **PID** controller. Initially, the PID parameters were tuned using the **Ziegler-Nichols** method and applied in a simulation to assess basic control performance. To enhance system efficiency, a **metaheuristic algorithm** was then employed to optimize the PID controller parameters, aiming to improve the motor's speed, torque, and current characteristics. The project followed a systematic design approach combining motor modeling, PID control, and optimization. Simulation results demonstrated improved performance, and the final system was implemented and tested on real hardware to validate the simulation outcomes and assess practical effectiveness.

يهدف هذا المشروع إلى التحكم في سرعة محرك تيار مستمر بدون فرش (BLDC) باستخدام منظم .PID في البداية، تم ضبط معاملات PIDباستخدام طريقة زيغلر-نيكولز، وتطبيقها في المحاكاة لتقييم الأداء الأساسي للتحكم. وبعد ذلك، وُظّف خوارزمية ميتا-استدلالية لتحسين معاملات PID بهدف تعزيز كفاءة النظام وتحسين خصائص السرعة والعزم والتيار في المحرك.

اتبع المشروع منهجية تصميم منهجية تجمع بين نمذجة المحرك، والتحكم بواسطةPID ، ومرحلة التحسين. وقد أظهرت نتائج المحاكاة أداءً محسّنًا، وتم تنفيذ النظام النهائي واختباره على عتاد حقيقي للتحقق من نتائج المحاكاة وتقييم فعاليته في الواقع العملي.

Table of Contents

Acknowledgement 1

Dedication 2

General Introduction 10

CHAPITRE 1 : BLDC Motor Modeling and PID Control 13

1.1 Introduction . . . 13
I.2 Generalities on Brushless DC Motor : . . . 13
1.2.1 History and Evolution of Brushless DC Motors (BLDC) : . . . 13
1.2.2 Control Methods for BLDC Motors : . . . 13
1.2.3 The difference between conventional DC motors (DC) and brushless DC motors (BLDC) : . . . 14
1.3 Fundamentals of BLDC Motor : . . . 14
1.3.1 Mathematical Model of BLDC : . . . 15
1.3.2 Electrical equations : . . . 16
1.3.3 Electromagnetic Torque : . . . 17
1.3.4 Dynamic equation : . . . 17
1.4 Fundamentals of PID Controller . . . 17
1.4.1 Historical Evolution of PID Controllers (From Ziegler-Nichols to modern optimization methods) . . . 17
1.4.2 PID Control Theory and Structure . . . 18
1.4.3 PID Tuning and parameter effect : . . . 18
1.5 Ziegler–Nichols Tuning Method : . . . 19
1.5.1 First Method : . . . 19
1.5.2 Second Method : . . . 20
1.6 SIMULATION RESULTS : . . . 21
1.6.1 Open-loop system : . . . 22
1.6.2. Closed loop system : . . . 25
I.7 Conclusion : . . . 29

CHAPITRE II : Metaheuristic Algorithms for PID Optimization 31

II.1. Introduction . . . 31
II.2. Overview of Metaheuristic Algorithms : . . . 31
II.3. Classification : . . . 31
II.4. Comparative analysis of Conventional and Metaheuristic Optimization Approaches : . . . 32
II.5. Selected Modified Metaheuristic Algorithms for PID Optimization : . . . 33

II.5.1 Generalized Normal Distribution Optimization (GNDO) : 33
II.5.2 Equilibrium Optimizer (EO) : 38
II.5.3 Slime Mould Algorithm (SMA) : 41
II.6 Comparative Analysis of Modified GNDO, EO, and SMA : 45
II.7 Control of BLDC Motor with GNDO, EO, and SMA : 45
II.8 SIMULATION RESULTS : 46
II.9 Conclusion 50

CHAPITRE III : Application and Real-Time Control of BLDC Motor **52**
III.1. Introduction 52
III.2 Simulation of BLDC Motor Control in Proteus 52
III.2.1 Simulation Setup 52
III.2.2 Control Methods for BLDC Motors : 53
III.2.3 Simulation Observations 55
III.2.4 Oscilloscope Analysis of Motor Phases 55
III.3 Real Hardware Implementation : 56
III.3.1 Components Used : 57
III.3.2 Arduino Code for ESC Control : 57
III.4. Real Hardware Implementation 57
III.4.1 BLDC Motor (A2212/15T 930KV) 58
III.4.2 Electronic Speed Controller (ESC) 59
III.4.3 Potentiometer 59
III.5 Serial Monitoring and Speed Control Analysis 60
III.6 Observations and Discussion 61
III.7 Conclusion 61

General Conclusion **63**

BIBLIOGRAPHY **65**

list of figures

I.1 : Evolution of electric motor technology. 13
I.1 : Evolution of electric motor technology. 14
I.2. :BLDC motor. 15
I.3. :BLDC Motor Control System and Mechanical Dynamics. 16
I.4. :Evolution and optimization of PID controller [8]. 17
I.5. : The working principle of the PID controller 18
I.6. : Unit step response of a plant and its S-shaped curve. 20
I.7. : The sustained oscillations with period Pcr. 21
I.8. : Open-loop design of BLDC motor. 22
I.9.– Open-Loop Speed Response of the BLDC Motor (Load = 0 Nm) . . 22
I.10.– Open-Loop Electromagnetic Torque of the BLDC Motor (Load =0 Nm) 23
I.11. – Open-Loop Stator Current Response of the BLDC Motor (Load = 0 Nm) 23
I.12.– Open-Loop Speed Response of the BLDC Motor (Load = 5 Nm) . 23
I.13.– Open-Loop Electromagnetic Torque of the BLDC Motor (Load = 5 Nm) 24
I.14.– Open-Loop Stator Current Response of the BLDC Motor (Load = 5 Nm) 24
I.15. : closed-loop design of BLDC motor 25
I.16.– Closed-Loop Speed Response of the BLDC Motor (Load = 0 Nm). 26
I.17.– Closed-Loop Electromagnetic Torque of the BLDC Motor (Load = 0 Nm) 26
I.18.– Closed-Loop Stator Current Response of the BLDC Motor (Load = 0 Nm) 26
I.19.– Closed-Loop Speed Response of the BLDC Motor (Load = 5 Nm) 27
I.20.– Closed-Loop Electromagnetic Torque of the BLDC Motor (Load = 5 Nm) 27
I.21.– Closed-Loop Stator Current Response of the BLDC Motor (Load = 5 Nm) 27
I.22. : Speed response of BLDC motor with inverter and PID) 28
I.2.3 : : Torque response of BLDC motor with inverter and PID 28
I.24. : Stator current response with inverter and PID 29
II.1 : Optimization Methods Classification) 32
II..2 : Differences Between Classical and Metaheuristic Optimization) . 33
II.3 : Flowchart of GNDO 37
II.4 : Methode EO 38

II.5 : Flowchart of EO 40
II.6 : Methode SMA 41
II.7 : Flowchart of SMA 44
II.8 : optimization shema of PID controller with GNDO, EO, and SMA . 45
II.9 Fitness function curve of GNDO ,EO and SMA 47
II.10 : speed response obtained by using GNDO, EO, SMA and Z-N PID in Empty test 47
II.11 : speed response obtained by using GNDO, EO, SMA and Z-N PID in Reverse speed test 48
II.12 : speed response obtained by using GNDO, EO, SMA and Z-N PID in With load operation 48
III.1 Arduino-Based Sensorless BLDC Motor Control Circuit in proteus 53
III .2 Proteus Oscilloscope Waveforms of Three Phases (A, B, C) 55
III.3 –Real Hardware Setup of BLDC Motor 56
III.4 Real Hardware Setup of BLDC Motor with Arduino 56
III.4 Serial Monitor Output with 11.77 V Power Supply 60
III.5 Graph of AnalogRead Input and PWM Speed Command 61

List of Abbreviations

— - :

BLDC Brushless Direct Current

DC Direct Current

PID Proportional-Integral-Derivative

Z-N Ziegler-Nichols

EMF Electromotive Force

FOC Field-Oriented Control

GNDO Gradient-based Optimizer (GNDO)

EO Equilibrium Optimizer

SMA Slime Mould Algorithm

RPM Revolutions Per Minute

Kp Proportional Gain

Ki Integral Gain

Kd Derivative Gain

MATLAB Matrix Laboratory (software)

SIMULINK Simulation and Link (software)

TL Load Torque

Te Electromagnetic Torque

J Rotor Inertia

B Friction Coefficient

General Introduction

The construction of modern drive systems demands ever-more continuous and accurate control of torque, speed, and position with high-performance, stability, and efficiency requirements. Therefore, contemporary research aims to create high-performance electric actuators that are more robust, less electromechanically sensitive, more efficient, and with better torque and speed.

Among the advanced actuators, the Brushless Direct Current Motor (BLDCM) is a quantum leap over the conventional DC motor. In BLDCMs, mechanical commutation is forsaken and replaced by purely electronic switching, thereby discontinuing direct physical contact between rotor and stator. One feature of BLDC motors is that there are sensors in the stator that measure the rotor's position precisely to modulate the electronic commutation of phase currents.

The primary goal of this research is to improve the dynamic speed response of the BLDC motor by employing the design of an optimized PID controller based on selected metaheuristic algorithms (GNDO, EO, and SMA). The primary phases of this work are :

- Establishing a model for the BLDC motor.
- Studying PID control by employing the Ziegler-Nichols (Z-N) tuning rule.
- Exploring the controller design based on the selected metaheuristic techniques.
- Comparative analysis with Z-N PID control (with GNDO, EO, SMA) and control methods using metaheuristic algorithms.

This thesis consists of three main chapters :

The first chapter gives the historical background of BLDC motors, introducing their applications, merits, demerits, operation, design, and mathematical modeling. It also gives the basics of PID control, introducing its proportional, integral, and derivative components, and feedback systems. The Ziegler-Nichols tuning technique is also presented along with its mathematical foundation. Although a MATLAB/SIMULINK simulation of the Z-N PID controller was used to regulate motor speed, the performance was not satisfactory, and hence, other control techniques were explored, which are presented in the second chapter.

The second chapter discusses metaheuristic algorithms, their development, fundamental differences from conventional control schemes, and reporting case studies of GNDO, EO, and SMA algorithms. It also comprises a comparative study with the simulation of these algorithms on the BLDC motor and performance analysis.

The third chapter : This chapter focuses on the experimental implementation of a control strategy for a Brushless DC (BLDC) motor using open-loop methods. The project was carried out in two ways : through software simulation using Proteus and hardware implementation using an Arduino Uno board, an Electronic Speed Controller (ESC), and a potentiometer.

Chapter I
BLDC Motor Modeling and PID Control

I.1 Introduction :

Brushless DC (BLDC) motors are widely used because they are efficient, reliable, and easy to control. To better understand how a BLDC motor works, we will first create a model that includes its electrical, mechanical, and electromagnetic components. Then, we will apply a PID controller to improve how the motor is controlled. Finally, we will test the motor's performance under different conditions using MATLAB/Simulink[1][2].

I.2 Generalities on Brushless DC Motors :

1.2.1 History and Evolution of Brushless DC Motors (BLDC) :

BLDC motors emerged in the mid-20th century with advancements in semiconductors and electronics. This section explores their historical development, This section provides an overview of their historical development and evolution[1][2].

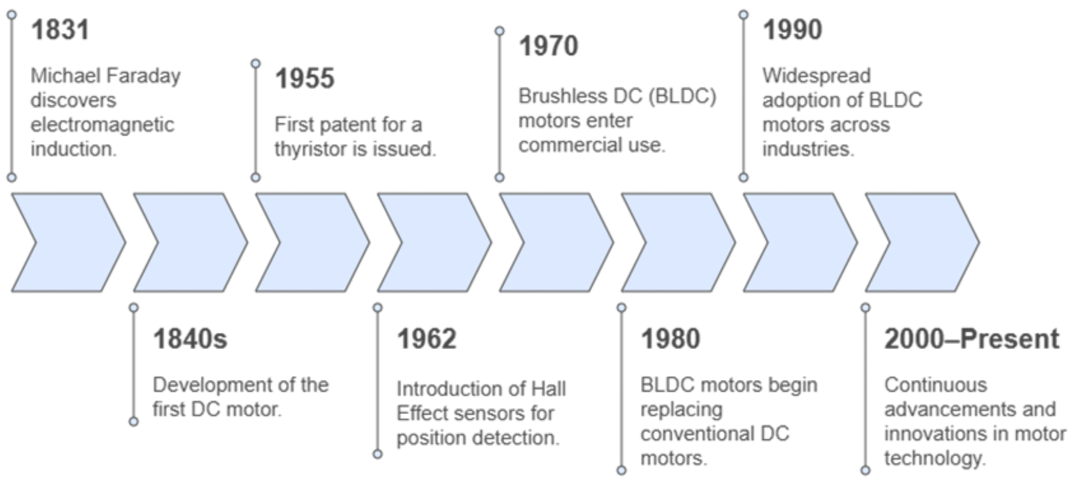


FIGURE – I.1 : Evolution of electric motor technology.

1.2.2 Control Methods for BLDC Motors :

Brushless DC (BLDC) motors are controlled by different methods based on the type of back electromotive force (EMF) and the intended performance behavior. Among these three, the most commonly employed are Trapezoidal Control, Sinusoidal Control, and Field-Oriented Control (FOC). All of them possess special strengths and trade-offs such that the selection is greatly subject to the application's requirements. This work adopted the Trapezoidal Control method for BLDC motor control[3][4][5].

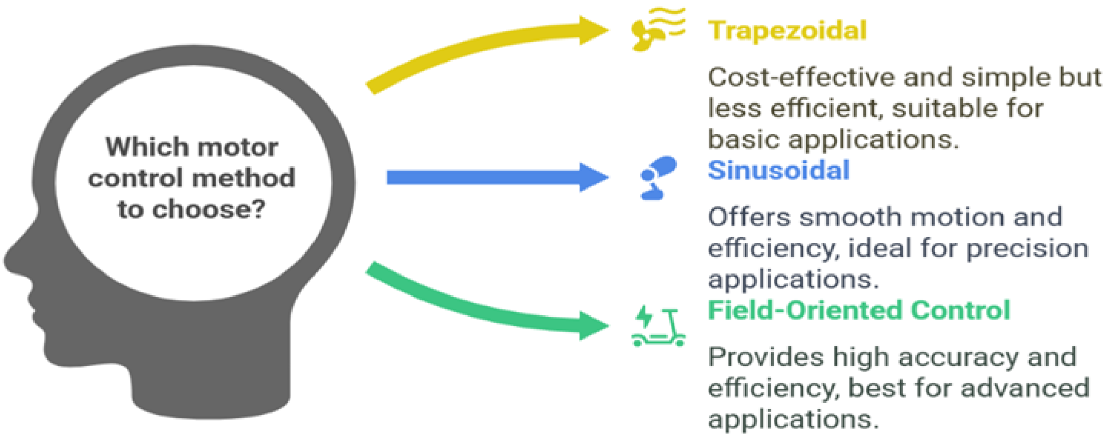


FIGURE – I.1 : Evolution of electric motor technology.

1.2.3 The difference between conventional DC motors (DC) and brushless DC motors (BLDC) :

Caracteristic	DC motor	Brushless motor
Size	More significance since it contains brushes and mechanical components.	Compact because it does not contain brushes and mechanical components.
Efficiency	Lower efficiency due to commutator losses and thermal dissipation. .	As there is no commutator, higher efficiency with reduced thermal losses.
Control	Less precise, relying on mechanical commutation for simple control	More precisely, electronic circuits with Hall effect sensors or sensorless control can be used for smoother control.
Starting Torque	Moderate, suitable for low-torque applications	High starting torque, suitable for machines requiring sudden movement.
Operational Components	Features brushes, a rotor, and a mechanical commutator that generate a magnetic field through electrical currents.	Uses Hall effect sensors to detect rotor position, electronically regulating current without moving parts.
Maintenance	High maintenance due to brush and commutator wear, requiring frequent maintenance.	High maintenance due to brush and commutator wear, requiring frequent maintenance.
Applications	Generally used in low-level applications like pumps and simple motors, where precision and speed control are not so crucial.	Preferred in high-performance applications like robotics, drones, medical devices, and automobile systems requiring efficiency and precision.

[2][6]

1.3 Fundamentals of BLDC Motor :

The figure below further illustrates these differences by showing the physical layout and design of both the stator and rotor in a typical BLDC motor [6].

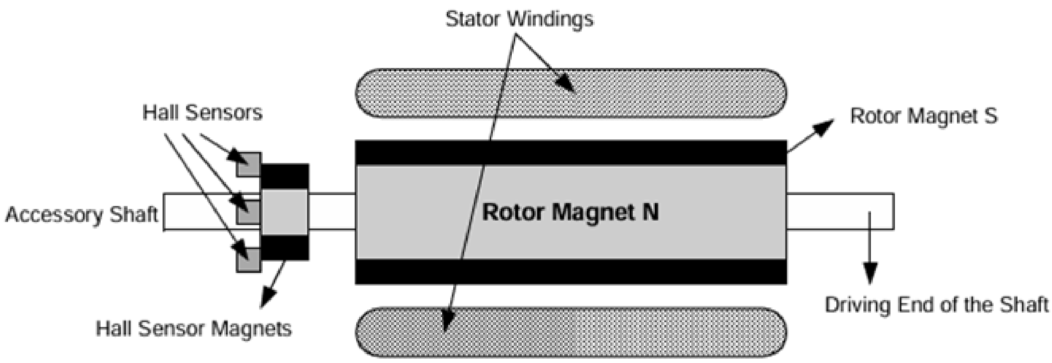


FIGURE – I.2. :BLDC motor.

To better understand how the BLDC motor works, it is important to analyze its main parts : the stator and the rotor. The table below shows a basic comparison between them in terms of structure, materials, and function [7].

Component	Stator	Rotor
Description	The stator is the stationary part of the BLDC motor, which generates a rotating magnetic field.	The rotor is the rotating part, made up of permanent magnets, and spins as a result of the stator field.
Material	Built with laminated silicon steel with copper windings in slots.	Has a steel core with rare-earth permanent magnets such as NdFeB.
Construction	Has three-phase distributed windings in slotted cores.	Two categories : 1-Surface-mounted magnets 2- Interior (buried) magnets (IPM).
Function	Generates a rotating magnetic field when current flows through the windings.	Follows the magnetic field of the stator and produces torque.
Types	Not categorized by type, but the winding configuration is mentioned.	Special types : Surface-mounted and IPM.
Cooling	The windings generate heat, and cooling is required.	Produces less heat but may require passive or active cooling for heavy power usage.

1.3.1 Mathematical Model of BLDC :

A low-order differential model of a three-phase, two-pole BLDC motor is developed in this section. The motor incorporates a full-pitch wound Y-connected stator and a non-salient rotor. Three Hall sensors, 120° apart, are used for position sensing. In modeling, the following are assumed : magnetic losses and armature reaction are ignored, the air-gap field is trapezoidal, cogging is ignored, and inverter devices are ideal. The resulting simplified motor schematic is shown in Figure I.3 [7].

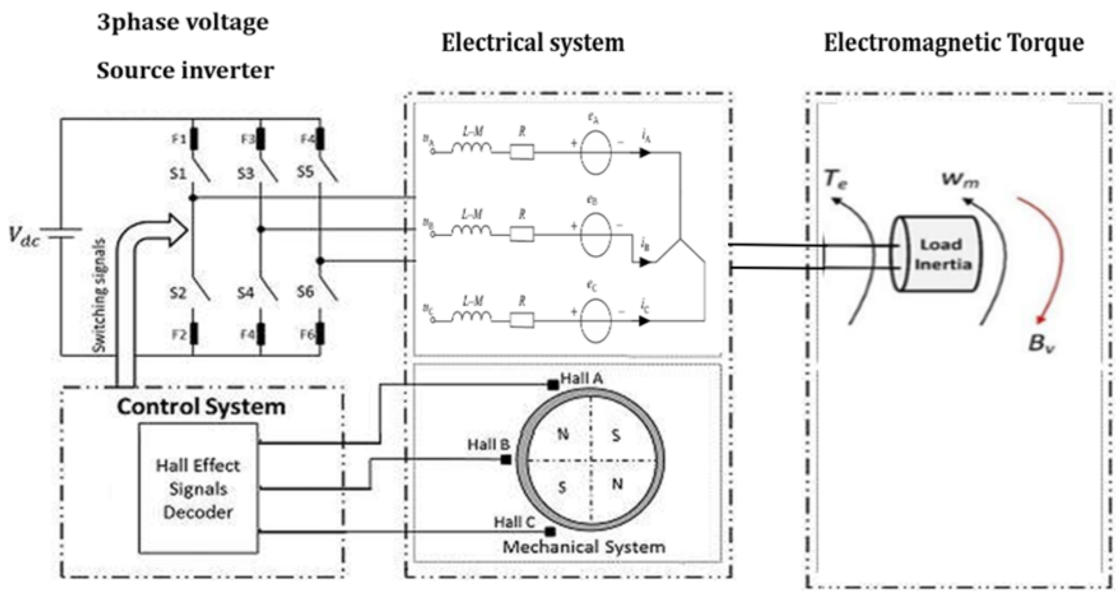


FIGURE – I.3. :BLDC Motor Control System and Mechanical Dynamics.

1.3.2 .Electrical equations :

The following set of equations gives the electrical behavior of the BLDC motor : Can be written in a simplified form as [7] :

$$u_A = Ri_A + (L-M)\frac{di_A}{dt} + e_A \tag{1}$$

Correspondingly, phase voltage equations can be written in the matrix form as :

$$\begin{bmatrix} u_A \\ u_B \\ u_C \end{bmatrix} = \begin{bmatrix} R & 0 & 0 \\ 0 & R & 0 \\ 0 & 0 & R \end{bmatrix} \begin{bmatrix} i_A \\ i_B \\ i_C \end{bmatrix} + \begin{bmatrix} L-M & 0 & 0 \\ 0 & L-M & 0 \\ 0 & 0 & L-M \end{bmatrix} \frac{d}{dt} \begin{bmatrix} i_A \\ i_B \\ i_C \end{bmatrix} + \begin{bmatrix} e_A \\ e_B \\ e_C \end{bmatrix} \tag{2}$$

Where :

- v_x : Phase voltage
- i_x : Phase current
- R : Phase resistance
- L : Phase inductance
- e_x : Back-EMF of each phase (non-sinusoidal in trapezoidal BLDC)

1.3.3 . Electromagnetic Torque :

The BLDC motor torque results from the magnet's placement, as show in Figure I.3. , the magnet is positioned where the electromotive force (EMF) is zero, for torque to be generated in the BLDC motor, current is to be applied to each phase when the electromotive force is available, the electrical torque produced by the BLDC motor is thus expressed as [1] :

$$T_e = \frac{e_a i_a + e_b i_b + e_c i_c}{\omega_m} \tag{3}$$

1.3.4 Dynamic equation :

The rotor's dynamic behavior is given by [1] :

$$\frac{dw_m}{dt} = \frac{1}{J}\left(T_L - T_e + B_v w_m\right) \tag{4}$$

Where :

- J : rotor inertia
- B_v : friction coefficient
- $Omega(w_m)$: rotor angular velocity
- Te : electromagnetic torque
- TL : load torque

1.4 Fundamentals of PID Controller

1.4.1 Historical Evolution of PID Controllers (From Ziegler-Nichols to modern optimization methods)

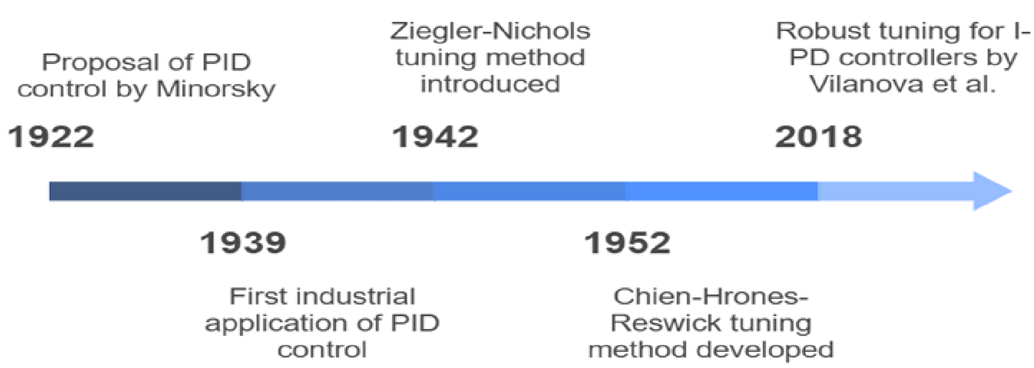


FIGURE – I.4. :Evolution and optimization of PID controller [8].

1.4.2 PID Control Theory and Structure

Figure I.5 shows the simulation model of the PID controller. The P, I, and D gains parameters are provided to the Motor. The PID controller is a three-term controller that stands for Proportional (P), Integral (I), and Derivative (D) controls, as shown in the above diagram. The controller includes concepts of Proportional, Integral, and Derivative terms by summing the product of error and respective gains, the output of the PID controller can be represented mathematically as follows [1].

$$u(t) = K_p\, e(t) + K_i \int_0^t e(\tau)\, d\tau + K_d \frac{d}{dt} e(t) \tag{5}$$

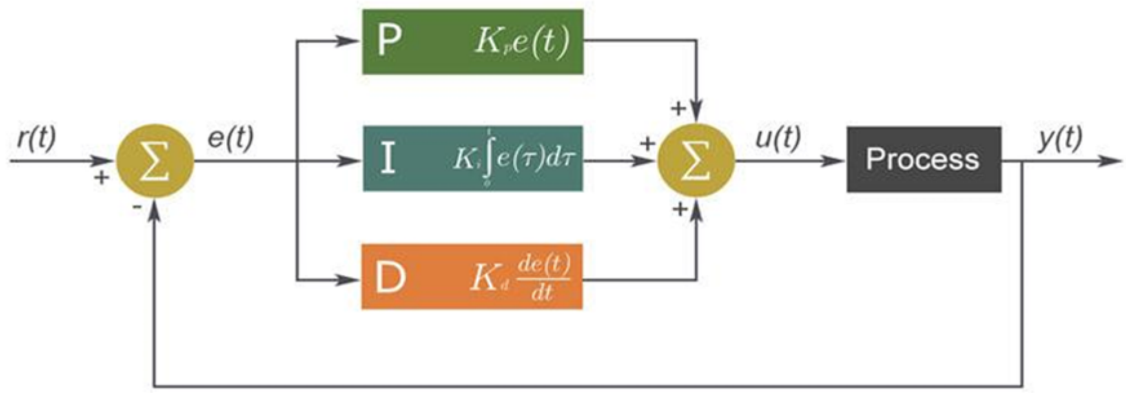
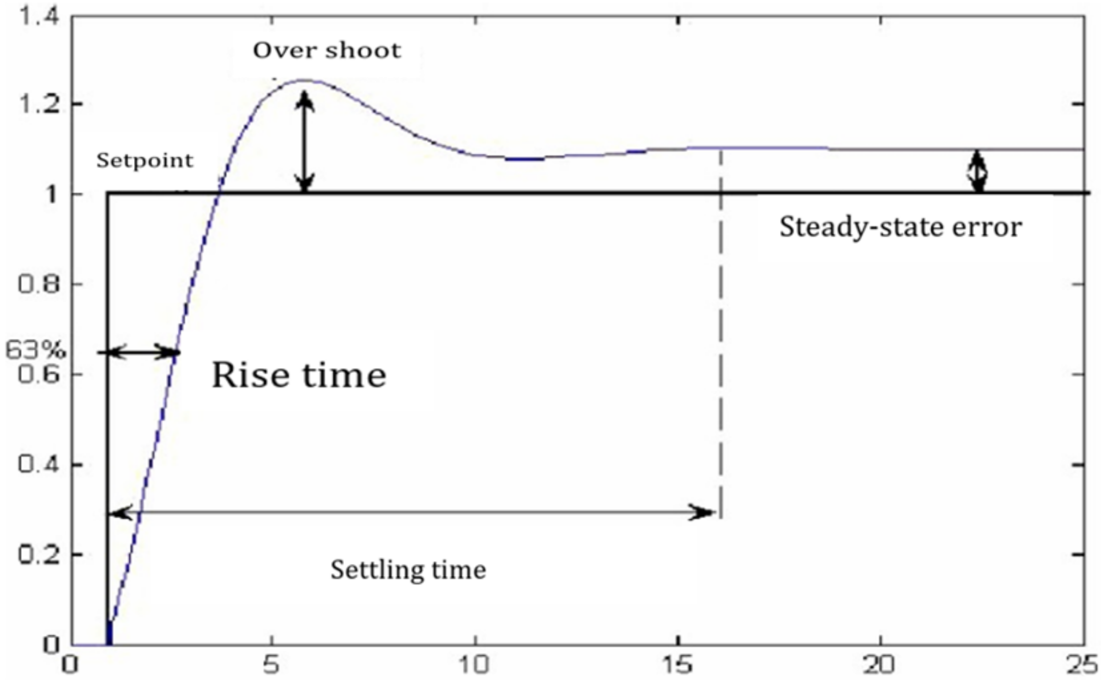


FIGURE – I.5. : The working principle of the PID controller

1.4.3 PID Tuning and parameter effect :

PID controller tuning is the process of selecting optimal values of the proportional (Kp), integral (Ki), and derivative (Kd) gains to generate the desired system response. The main objectives are robustness, quick response, and high precision with low overshoot, but robustness is the most important among these. A robust control system performs well irrespective of small changes in the model or operating conditions, giving the capability to handle uncertainties. The rate of the controller is determined by parameters such as rise time and settling time, which represent the rate at which the system reaches a steady state. Precision is quantified in terms of steady-state error, which characterizes how well the system output follows the reference desired [1].



To find the values of the PID controller parameters kp, ki kd, there are many known methods. One of the most classical and commonly used is the Ziegler–Nichols method (both the first and second versions), as it is known for its simplicity and usability.

1.5 Ziegler–Nichols Tuning Method :

1.5.1 First Method :

The experimental or simulated unit step response of the system should show an S-shaped curve for the Ziegler–Nichols procedure to be applicable. If the response isn't of this shape, the procedure can't be used [9] .

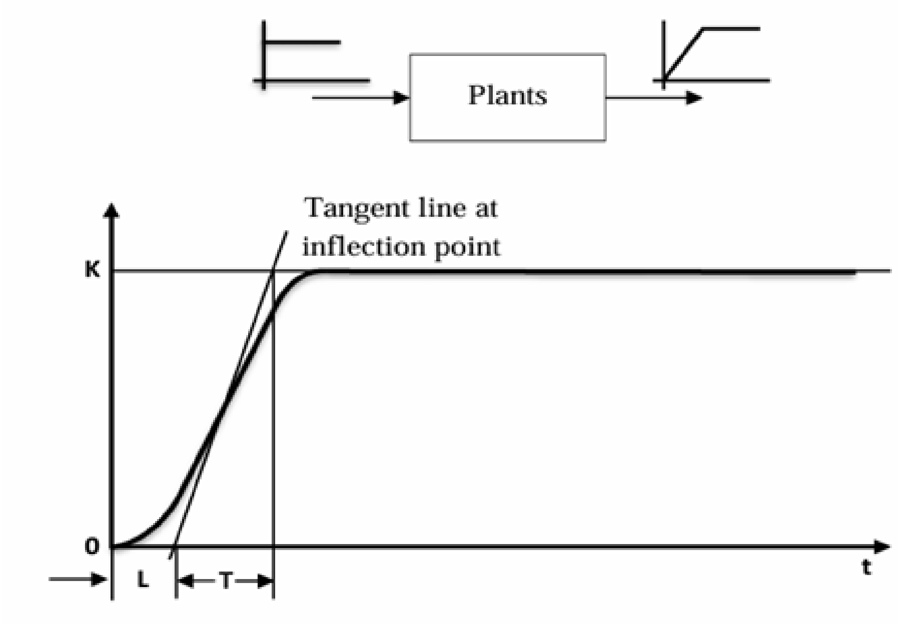


FIGURE – I.6. : Unit step response of a plant and its S-shaped curve.

• Determine the time delay (L) and time constant (T) from a tangent drawn at the inflection point of an S-shaped step response curve. Then, based on the resulting values.
• Utilize a typical Ziegler–Nichols tuning table to adjust PID parameters Kp, Ti, and Td for the utilized controller type.

Type of controller	$\mathbf{K_p}$	$\mathbf{T_i}$	$\mathbf{T_d}$
P	T/L	∞	0
PI	$0.9T/L$	$L/0.3$	0
PID	$1.2T/L$	$2L$	$0.5L$

TABLE 1 – 12 : The parameters of Kp, Ti and Td of the P,PI and PID by the first method of Ziegler Nichols.

1.5.2 Second Method :

• This method, similar to trial-and-error, begins by disabling the integral and derivative actions (Ti $=\infty$, Td $=0$).
• The proportional gain is increased until sustained oscillations exist, which sets the critical gain.
• If such oscillations are not feasible, the method is not effective. The final PID parameters are then selected using the critical gain and oscillation period, based on the Ziegler–Nichols tuning table.

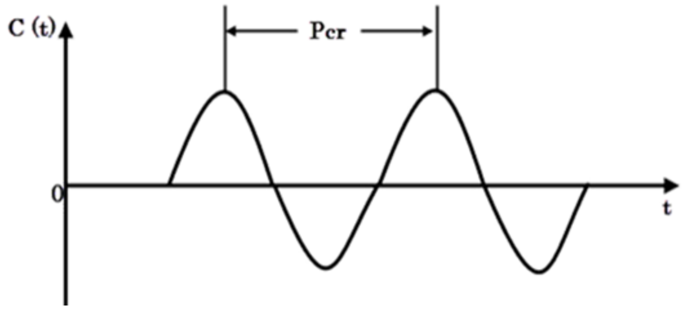
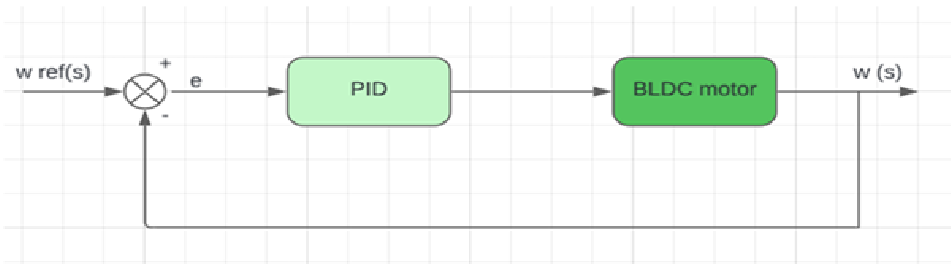


FIGURE – I.7. : The sustained oscillations with period Pcr. .

Type of controller	K_p	T_i	T_d
P	$0.5\,K_{cr}$	∞	0
PI	$0.45\,K_{cr}$	$1/1.2\,P_{cr}$	0
PID	$0.6\,K_{cr}$	$0.5\,P_{cr}$	$0.125\,P_{cr}$

TABLE 2 – 2– 14 :The parameters of Kp,Ti and Td of the P,PI and PID by the second method of Ziegler Nichols.

In this work, the second Ziegler–Nichols method was used to calculate the PID parameters, where Kp=365.8763, Ki=64.6501, and Kd=0.0946. These values were used in the closed-loop configuration of the BLDC motor. The figure below presents the model used in this setup.



1.6 SIMULATION RESULTS :

Before presenting the results of the simulation, several tests were conducted on the BLDC motorin MATLAB /SIMULINK under open-loop and closed-loop status, as seen in Figures 8 and 15. The motor was initially tested with no load (Load=0) to observe how it would react. A load of 5 was then applied (Load=5),We also changed the motor speed quickly from +1000 RPM to -1000 RPM only in the closed-loop configuration with no load to see if the motor can handle sudden changes and keep working smoothly, and the result was analyzed. We adopted the following parameters : The reference speed is 1000 RPM, the torque load value is TL=5 N.m, we applied it at the time T=0.15s,a voltage source of 300 V was used to supply the BLDC motor in all simulations.

1.6.1 Open-loop system :

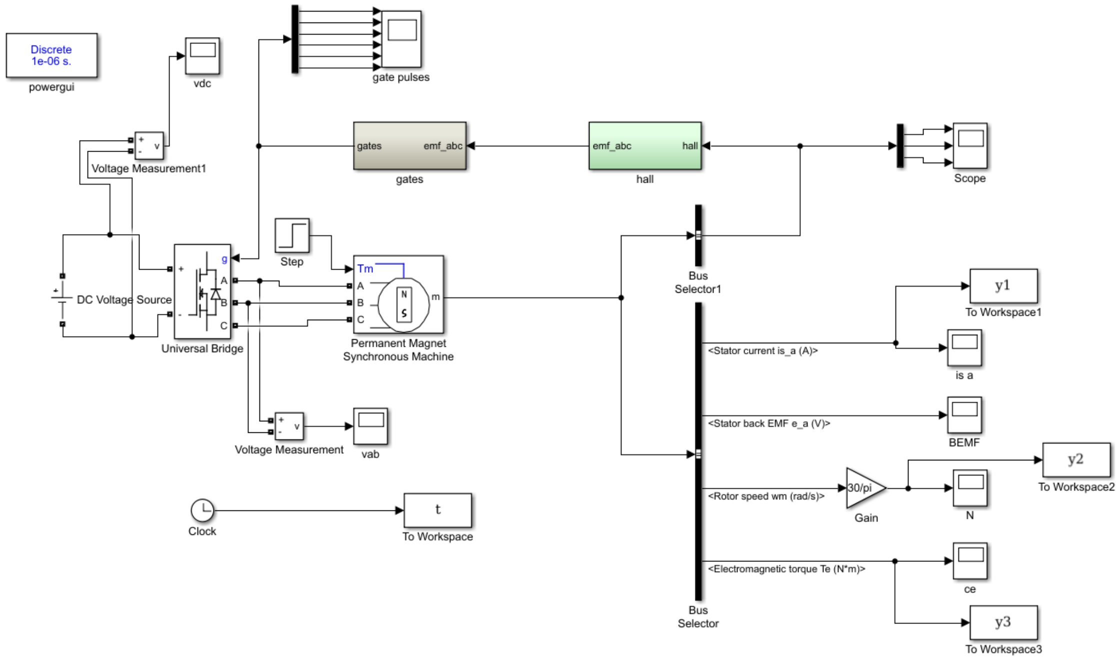


FIGURE – I.8. : Open-loop design of BLDC motor.

• with no load :

The simulation results are shown in the following figures :

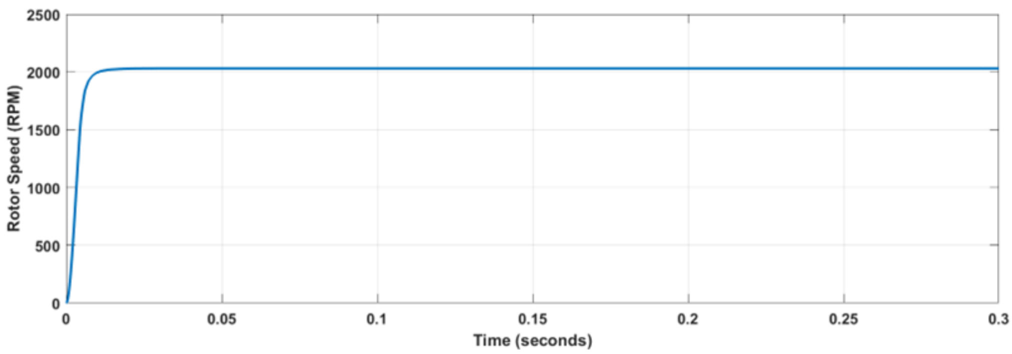


FIGURE – I.9.– Open-Loop Speed Response of the BLDC Motor (Load = 0 Nm)

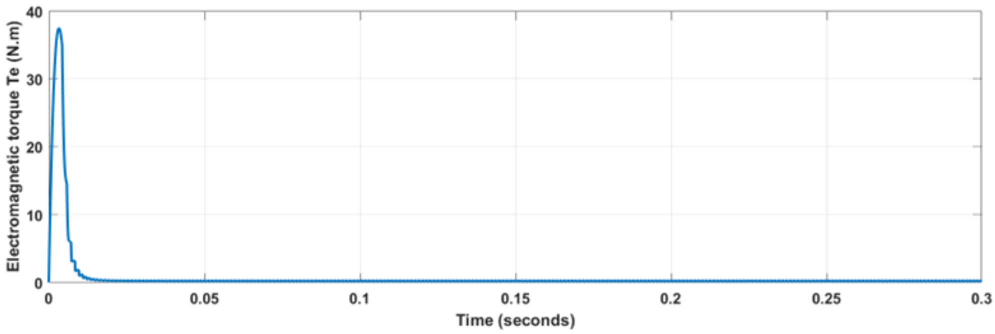


FIGURE – I.10.– Open-Loop Electromagnetic Torque of the BLDC Motor (Load =0 Nm)

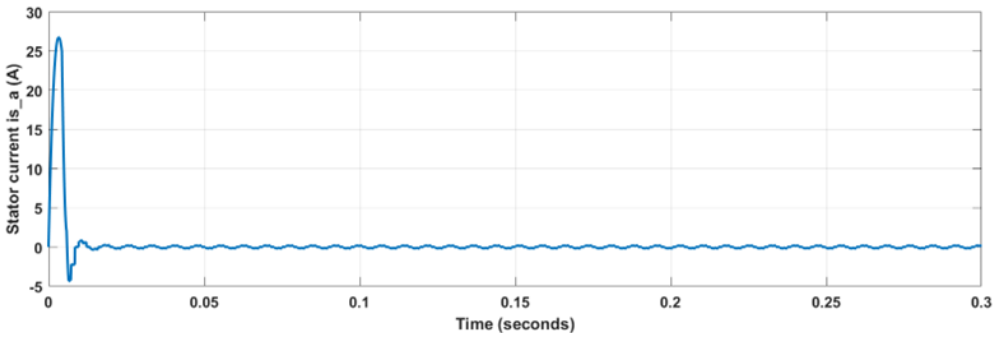


FIGURE – I.11. – Open-Loop Stator Current Response of the BLDC Motor (Load = 0 Nm)

• with Load = 5 Nm

The simulation results are shown in the following figures :

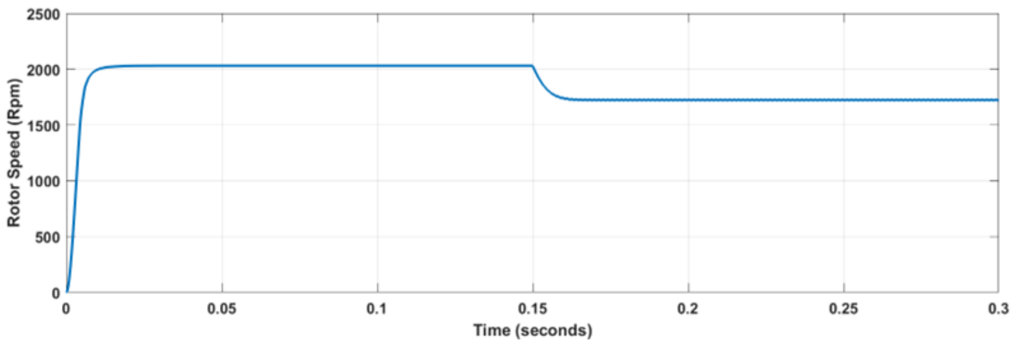


FIGURE – I.12.– Open-Loop Speed Response of the BLDC Motor (Load = 5 Nm)

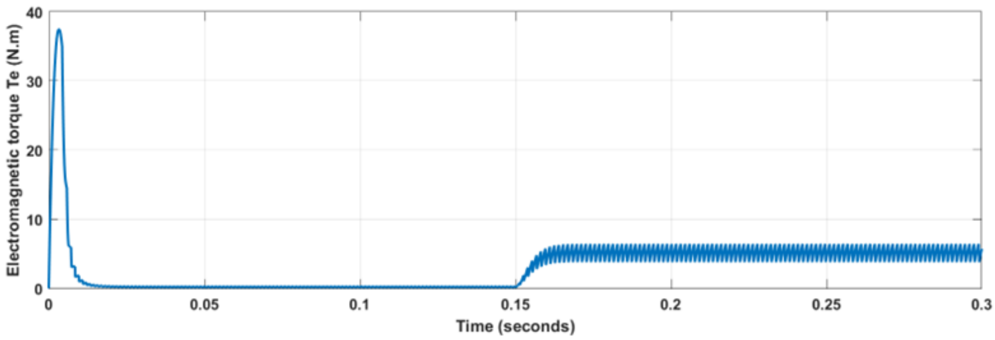


FIGURE – I.13.– Open-Loop Electromagnetic Torque of the BLDC Motor (Load = 5 Nm)

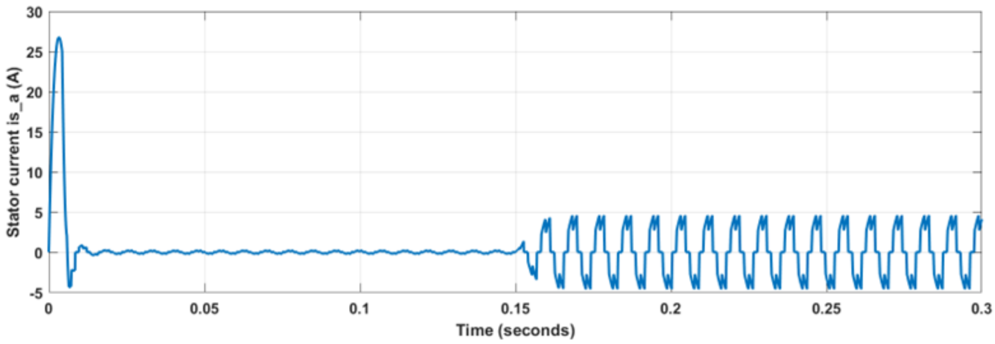


FIGURE – I.14.– Open-Loop Stator Current Response of the BLDC Motor (Load = 5 Nm)

• **Comments and interpretations :**

From Figure I.12, the BLDC motor starts and quickly attains a constant speed of about 2030 rpm. The 5 Nm load torque changes instantaneously at t = 0.15 s. As a result, the speed drops precipitously and levels off to about 1840 rpm. This type of speed drop is a natural phenomenon for an open-loop system because the initial speed has no feedback control to be maintained after the load increases.

According to Figure I.13, the electromagnetic torque increases initially to approximately 37 N·m in starting and reduces significantly and settles at an extremely small amount. The torque is surging at t = 0.15 s because of the change in load, and the action of the motor is to increase the torque. It varies with damped intervals at steady 5 N·m, which is a measure of the motor effort in maintaining the 5 Nm load in open-loop without control correction.correction.

Figures I.14 and I.11 show the stator current behavior when open-loop operation is done. On no-load, the current was peaking at about 26 A during startup and drops immediately to a steady 0.2 A with tiny ripples caused by phase switching. When the load of 5 Nm is applied, after which significant oscillations between +5 A and -5 A occur. This spike reflects the motor's additional effort in a loaded condition, as there is any feedback to regulate the current in open-loop control.

When the load is applied, the motor speed decreases and does not return to its original value. This is because the system operates in open-loop mode without any control. To solve this issue and maintain the desired speed, we will use a PID controller in the next step.

1.6.2. Closed loop system :

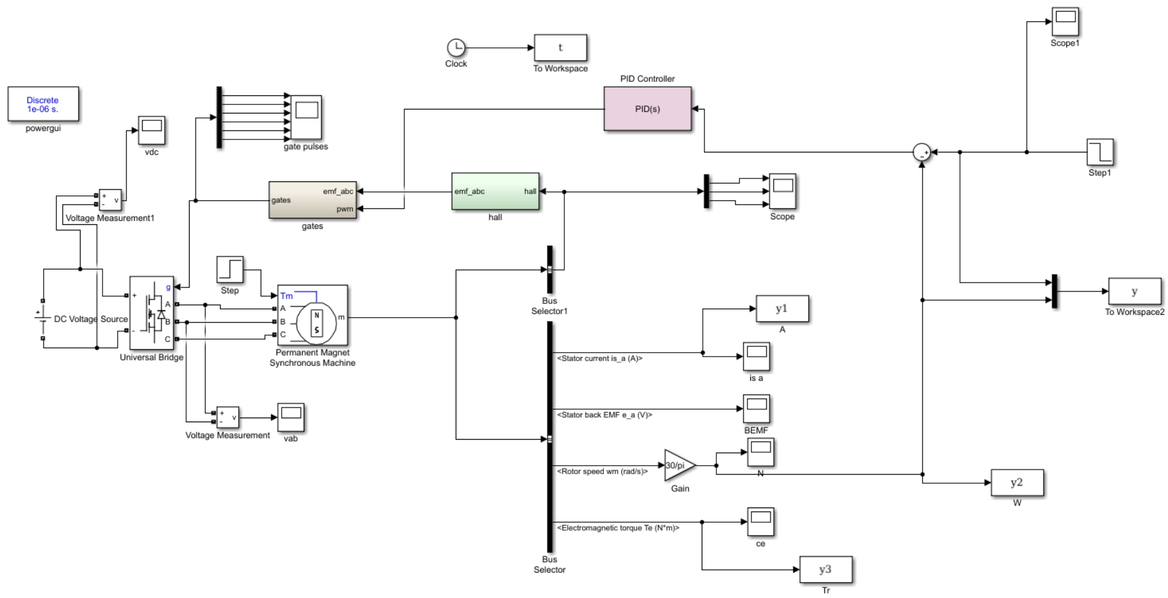


FIGURE – I.15. : closed-loop design of BLDC motor

The PID controller parameters are obtained (Kp=51.36, Ki=2153.45,kd=0.306)

• **with no load :**
The simulation results are shown in the following figures :

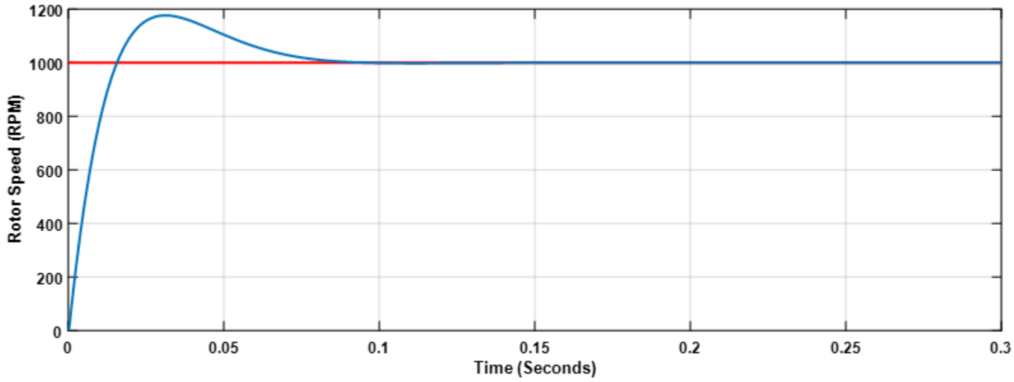


FIGURE – I.16.– Closed-Loop Speed Response of the BLDC Motor (Load = 0 Nm)

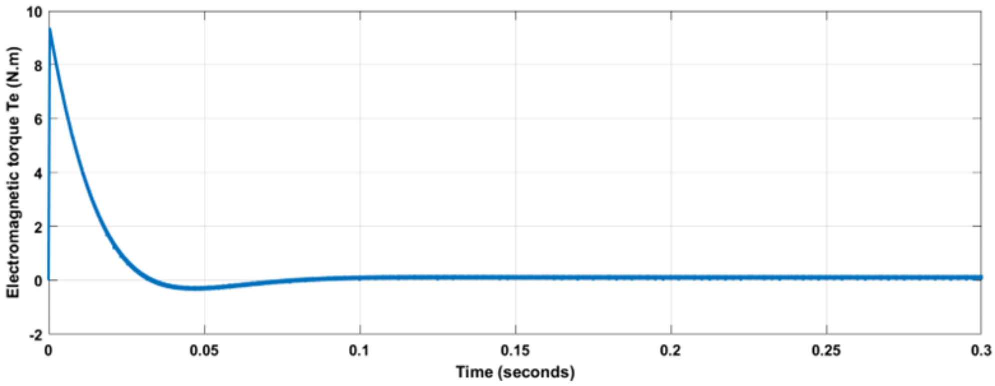


FIGURE – I.17.– Closed-Loop Electromagnetic Torque of the BLDC Motor (Load = 0 Nm)

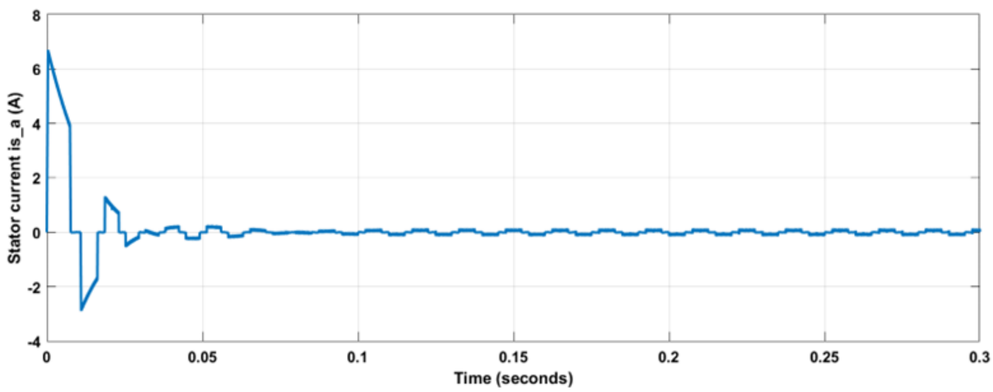


FIGURE – I.18.– Closed-Loop Stator Current Response of the BLDC Motor (Load = 0 Nm)

- **with Load = 5 Nm**

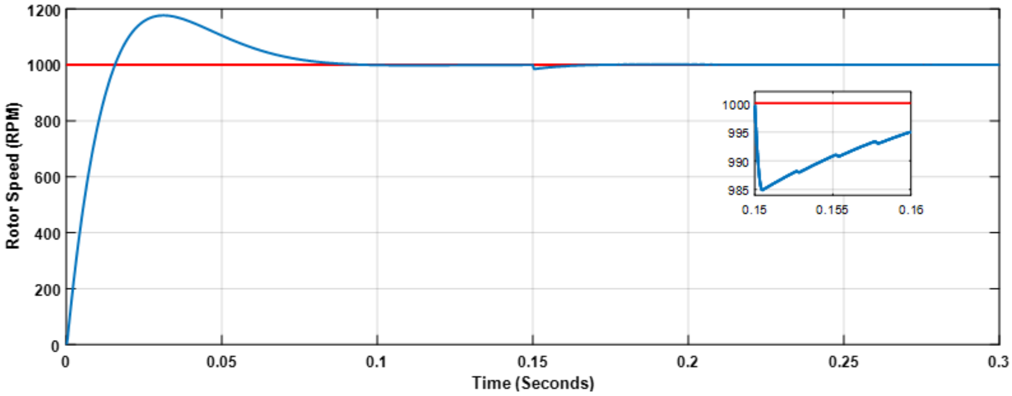


FIGURE – I.19.– Closed-Loop Speed Response of the BLDC Motor (Load = 5 Nm)

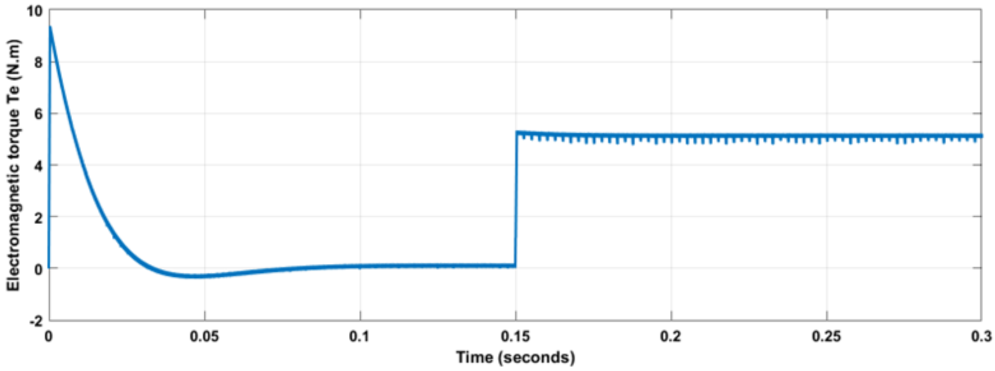


FIGURE – I.20.– Closed-Loop Electromagnetic Torque of the BLDC Motor (Load = 5 Nm)

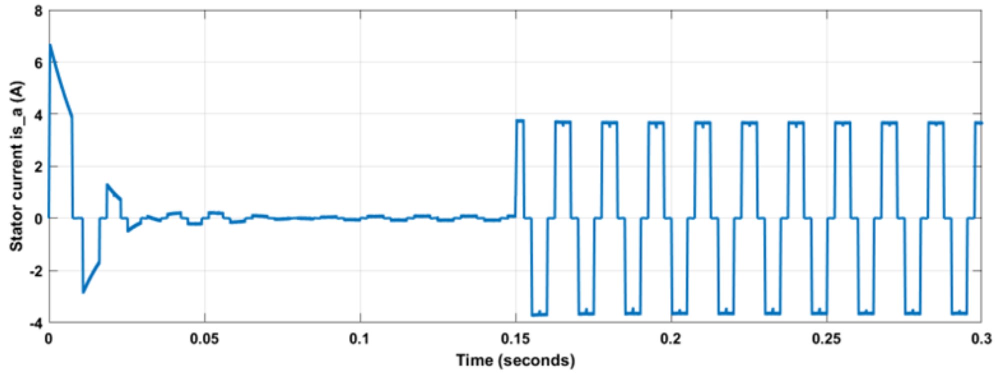


FIGURE – I.21.– Closed-Loop Stator Current Response of the BLDC Motor (Load = 5 Nm)

• **Comments and interpretations :**

Figure I.19. shows the closed-loop response of the BLDC motor shows the initial overshoot peaks at around 1180 RPM, decreasing followed by oscillations that gradually dampen,when the 5 Nm load is applied, the speed sharply drops to about 984 RPM, showing the strong impact of the load. The controller then acts to bring the speed back to the reference1000 RPM .

Figure I.20 shows the BLDC motor's torque under PID control, with an initial peak around 15.8 N·m, then settling near 0.6 N·m. When the 5 Nm load is applied, torque sharply rises to about 5.6 N·m to compensate. The response has a clear overshoot followed by damped oscillations. While the controller stabilizes the system, some residual oscillations suggest further tuning is needed to improve damping and performance.

As shown in Figure I.21, the BLDC motor's phase current quickly rises to about 6.8 A at the very beginning. Then, it drops sharply and settles, oscillating steadily between -4 A and 4 with stable oscillations over the long term

- **speed reversal :**

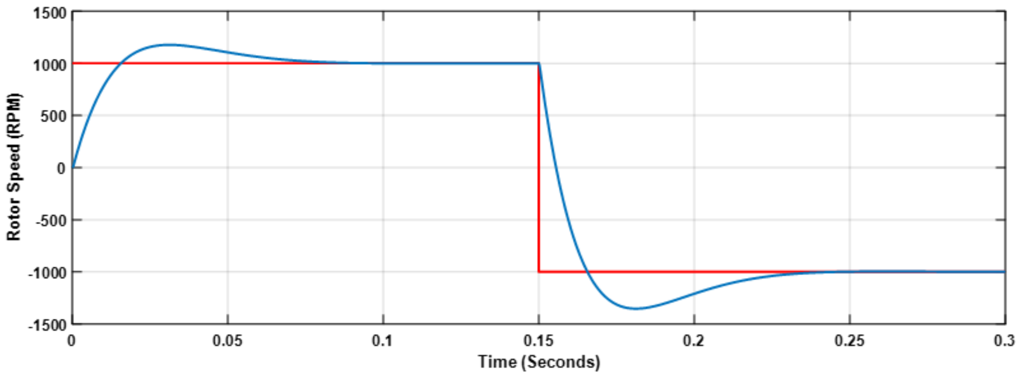


FIGURE – I.22. : Speed response of BLDC motor with inverter and PID)

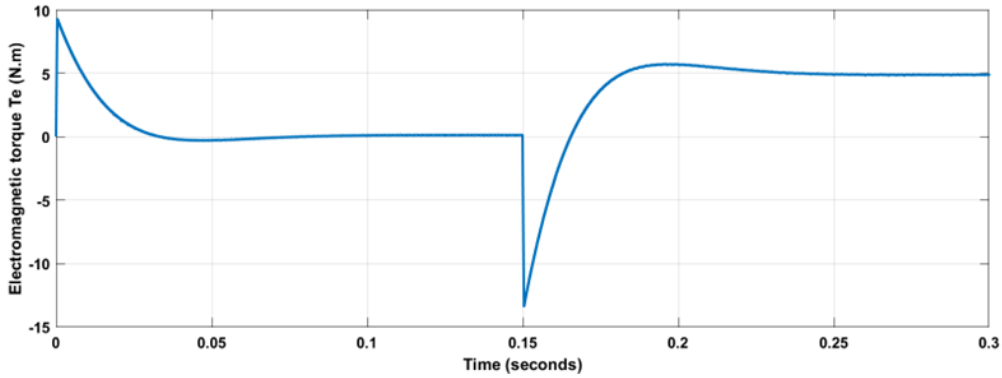


FIGURE – I.2.3 : : Torque response of BLDC motor with inverter and PID

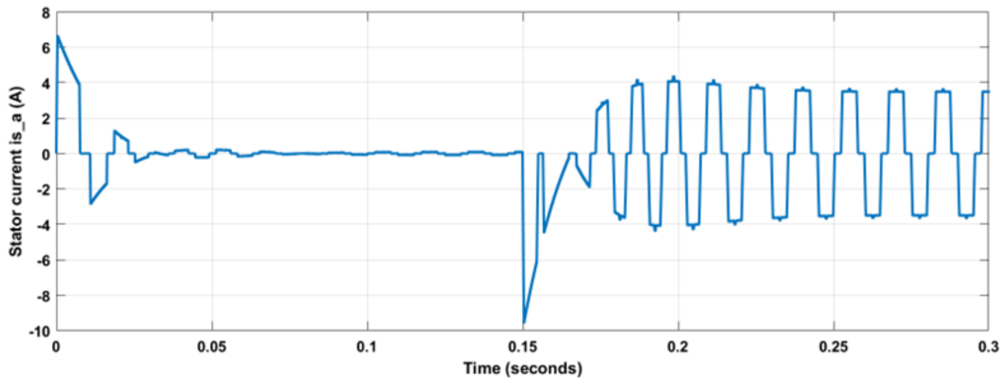


FIGURE – I.24. : Stator current response with inverter and PID

• Comments and interpretations :

In Figure I.22, the motor speed increases smoothly to +1000 rpm and then successfully reverses to -1000 rpm, showing stable and accurate control. Figure I.23 shows that the torque rises at the beginning to overcome inertia, then becomes stable. During speed reversal, a strong negative torque appears to slow down and reverse the motor, then it returns to steady state, which shows the PID controller's good response. In Figure I.24, the current rises sharply at startup, then becomes stable with small ripples. When the speed reverses, the current increases again before settling, which confirms that the controller maintains stability during changes in direction

I.7 Conclusion :

Based on the results, it is concluded that the speed response of the BLDC model with the PID controller possesses better dynamic characteristics than the BLDC model without a controller. The system with PID control reduces settling time and peak time, with greater stability in the steady state as well as the transient state. Furthermore, the improved speed response with reduced torque ripple contributes to making the system more stable and efficient.

Although the performance obtained by using the PID controller is a clear betterment over that of the uncontrolled model, it is still short of realizing the desired optimum level of performance. Therefore, the concern in the next chapter is to attempt to build upon and improve these results by using yet another type of control.

CHAPITRE II

Metaheuristic Algorithms for PID Optimization

II.1 Introduction :

Against the limitations of traditional PID controllers, which are highly reliant on machine parameters and may alter with operating changes, there is a growing need to use advanced optimization techniques. Among them, metaheuristic algorithms have been found to be successful tools for improving control systems. Optimization in a control system aims to achieve optimal efficiency and effectiveness by improving performance, minimization of specified criteria, and reduction in resource consumption and operating costs. Optimized systems also enhance accuracy, reliability, and safety with minimal waste and risk. Optimization techniques are specially crucial for real-time systems where stability must be maintained at all times. In this chapter, an enhancement by the application of GNDO, EO, and SMA metaheuristic algorithms for the optimization of a PID controller for BLDC motor speed control is elaborated and an analytical comparison between them is provided [10] [11].

II.2 Overview of Metaheuristic Algorithms :

Metaheuristic-based approaches tackle complex problems by offering solutions that are nearly optimal in a reasonable amount of time. Their effectiveness and efficiency in solving large-scale and complex problems have led to an enormous increase in their popularity over the last few years. Metaheuristics are advanced techniques to achieve superior performance than basic heuristic techniques. The prefix "Meta" indicates "beyond" or "higher level," which indicates their enhanced search capability. Each metaheuristic algorithm uses both global and local search methods so that it can optimally search for the solution space. Randomization is often employed to produce diversified solutions. No consensus definition, though the nomenclature is established, has been found in academic research regarding heuristics and metaheuristics, though they are being used more and more, even though they are used interchangeably by most researchers. However, present trends show that any stochastic algorithms utilizing randomization and global search methods belong to the class of metaheuristics in broad terms. This will also be the methodology followed in this assessment [12].

II.3 Classification :

This part shortly discusses and outlines a wide literature review of metaheuristic algorithms and their creation. Metaheuristic algorithms may be grouped in several forms, depending on selected features or properties that make them different. Several authors and researchers divide Metaheuristic algorithms by various features or properties. Classification might also be reliant or founded on objectives or study aims [12] [13] [14] [15] .

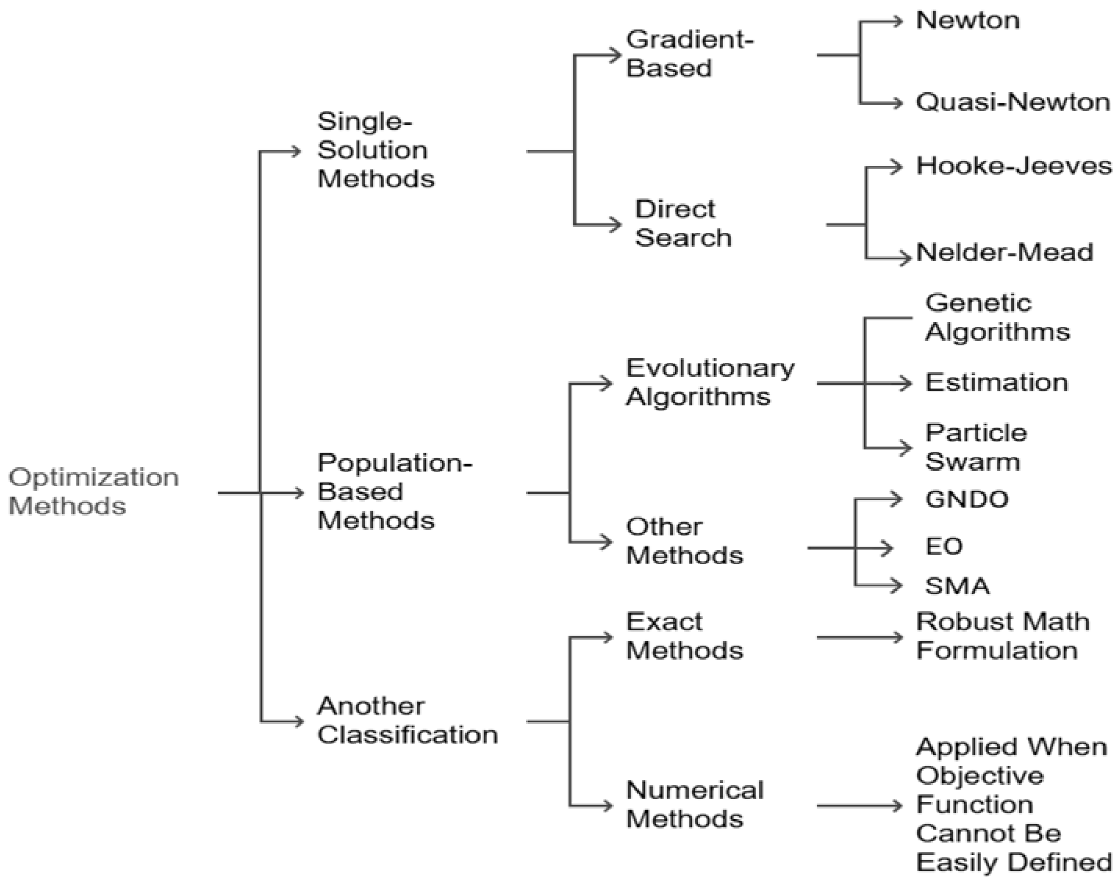


FIGURE – II.1 : Optimization Methods Classification)

II.4 Comparative analysis of Conventional and Metaheuristic Optimization Approaches :

We aim to compare traditional optimization methods with metaheuristic optimization methods based on several key aspects, Figure (II.2) illustrates this comparison in a simplified manner, highlighting the fundamental differences between the two approaches[16] [17][18] .

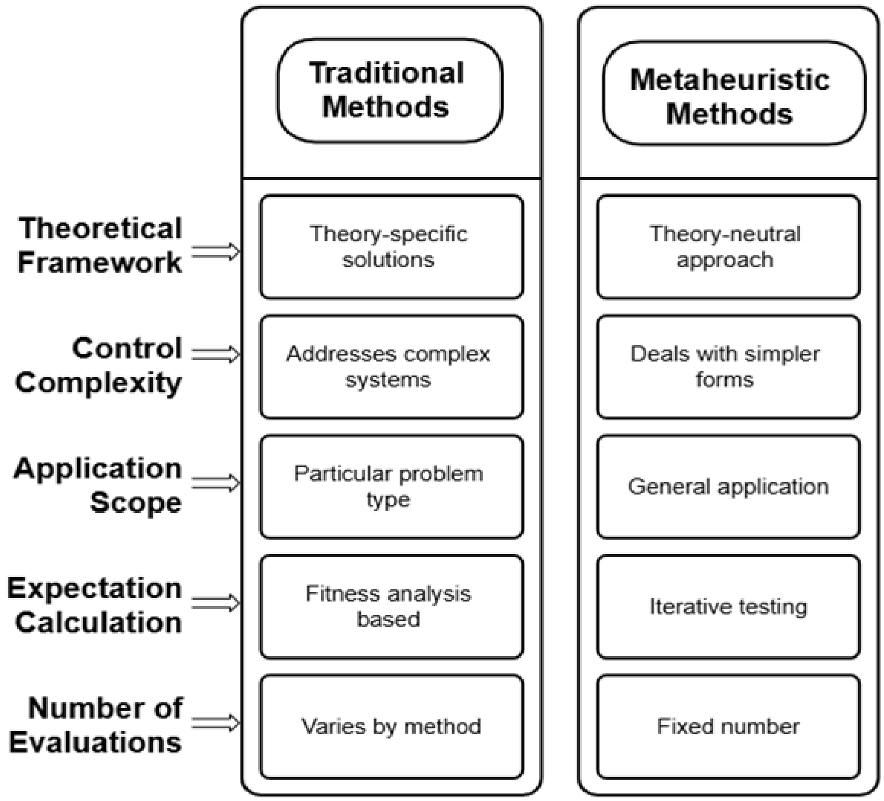


FIGURE – II..2 : Differences Between Classical and Metaheuristic Optimization)

II.5 Selected Modified Metaheuristic Algorithms for PID Optimization :

There are many metaheuristic algorithms used to optimize PID controllers, such as Particle Swarm Optimization (PSO), Genetic Algorithm (GA), and others. However, for the purpose of our work, we have selected the following algorithms : Generalized Normal Distribution Optimization (GNDO), Equilibrium Optimizer (EO), and Slime Mould Algorithm (SMA).

II.5.1 Generalized Normal Distribution Optimization (GNDO) :

The GNDO algorithm is based on the Generalized Normal Distribution model, which forecasts dependent variables employing various parameterized distributions such as normal, exponential, Rayleigh, and Weibull. It incorporates local exploitation through action by the ideal and mean positions as well as global exploration through random selection of three agents from the population. Such integration enhances the optimization process efficiency. In addition, GNDO is also GPU-enabled and able to solve black-box and brown-box optimization problems depending on the relationship between data distributions [19] [20] [21] [22] [23] .

A detailed explanation of both learning strategies is provided as follows :

population is initialized by :

$$x_{i,j}^t = l_j + (u_j - l_j) \times \lambda_5 \tag{6}$$

where : $i = 1, 2, 3, \ldots, N$, $j = 1, 2, 3, \ldots, D$.

where the number of design variables is defined by D, the number of population size is defined by N the lower and upper boundary of the jth design variable is described by lj and uj, respectively.5 is a random number in the interval of [0, 1].

Global Exploration :

The proposed method used an equation for exploration as follows :

$$v_i^t = x_i^t + \underbrace{\beta \times (|\lambda_3| \times v_1)}_{\text{Local learning term}} + \underbrace{(1-\beta) \times (|\lambda_4| \times v_2)}_{\text{Global learning term}} \tag{7}$$

where two random distributions were created 3 and 4 from the normal distribution. The parameter is a random number between 0 and 1. The following are the two trail vectors v1 and v2 :

$$v_1 = \begin{cases} x_i^t - x_{\text{p1}}^t, & \text{if } f(x_i^t) < f(x_{\text{p1}}^t) \\ x_{\text{p1}}^t - x_i^t, & \text{otherwise} \end{cases} \tag{8}$$

$$v_2 = \begin{cases} x_{\text{p2}}^t - x_{\text{p3}}^t, & \text{if } f(x_{\text{p2}}^t) < f(x_{\text{p3}}^t) \\ x_{\text{p3}}^t - x_{\text{p2}}^t, & \text{otherwise} \end{cases} \tag{9}$$

- **p1, p2, p3** : Three distinct integers randomly selected from 1 to N, such that

$i \neq p1 \neq p2 \neq p3$.

- **i** : Represents the current individual in the population.
- λ_3, λ_4 : Random variables that follow a standard normal distribution.
- β : Adjustment parameter used to balance between local exploitation and global exploration.

Local Exploitation :

The proposed method used an equation for exploitation as follows :

μ_i :The generalized mean position of individual i, calculated using variables such as :

$$v_i^t = \mu_i + \delta_i \times \eta, \quad i = 1, 2, 3, \ldots, N \tag{10}$$

$$M = \sum_{i=1}^{N} x_i^t \tag{11}$$

$$\mu = \frac{1}{3}\left(x_i^t + x_{\text{Best}}^t + M\right) \tag{12}$$

$$\delta = \sqrt{\frac{1}{3}\left[(x_i^t - \mu)^2 + (x_{\text{Best}}^t - \mu)^2 + (M - \mu)^2\right]} \tag{13}$$

$$\eta = \begin{cases} \sqrt{-\log(\lambda_1)}\cos(2\pi\lambda_2), & \text{if } a <= b \\ \sqrt{-\log(\lambda_1)}\cos(2\pi\lambda_2 + \pi), & \text{otherwise} \end{cases} \tag{14}$$

- M : The average position of the population.
- x_{Best}^t : The current best position in the population at time t.
- δ_i : The generalized standard deviation.
- v_i^t : The trailing (or velocity) vector of individual i at time t.
- η : The penalty factor.

Random numbers (a, b, λ_1, and λ_2) between 0 and 1 are also used in computing these values.

To ensure that improved solutions are carried forward into the subsequent generation's population, a selection mechanism has been devised, which can be described as follows :

$$x_i^{t+1} = \begin{cases} v_i^t, & \text{if } f(v_i^t) < f(x_i^t) \\ x_i^t, & \text{otherwise} \end{cases} \tag{15}$$

The proposed GNDO algorithm's flowchart is shown in Figure II.3.

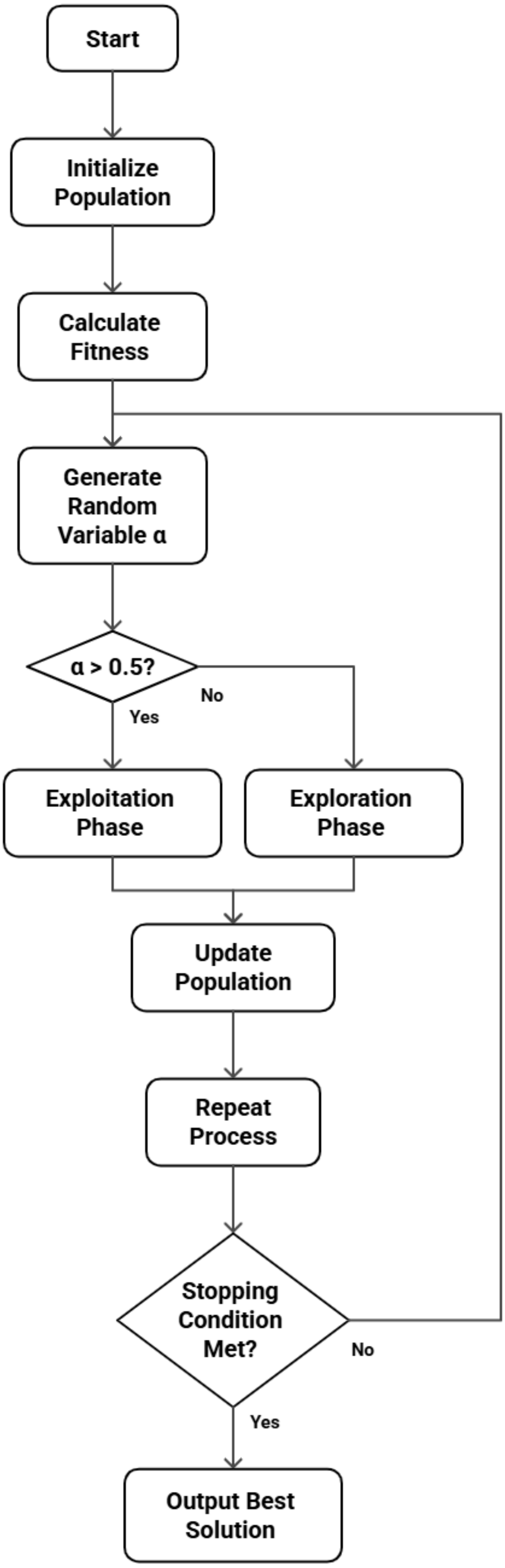


FIGURE – II.3 : Flowchart of GNDO

II.5.2 Equilibrium Optimizer (EO) :

Equilibrium Optimizer (EO) is a relatively new intelligent algorithm drawing inspiration from physical phenomena such as diffusion, heat exchange, osmosis, and chemical reactions. The principle of EO is to find the point of equilibrium—where there are no more changes along any axis. EO achieves this by balancing concentration within the search space, basically driving particles towards optimal positions and maintaining them there. One of the virtues of EO is its simplicity ; its algorithmic representation is concise and simple, typically reduced to a one-page pseudocode. The EO process consists of three main mathematical phases : (1) initialization of the population, (2) building the equilibrium pool and candidates, and (3) updating concentration levels [24] [25] [26] [27].

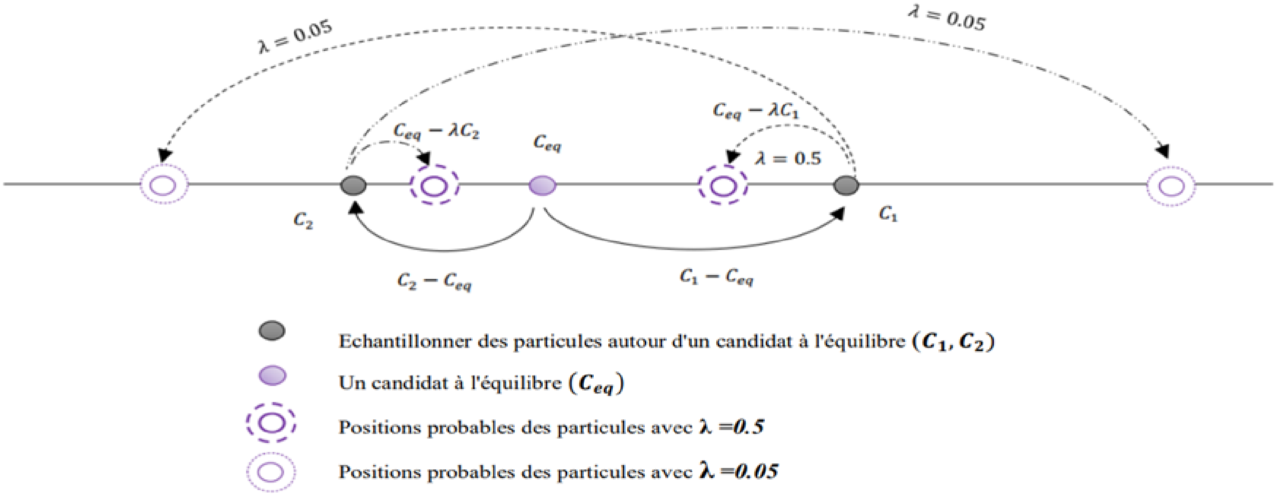


FIGURE – II.4 : Methode EO

Initialization :

Similar to evolutionary algorithms based on population development, the EO algorithm relies on the random generation of an initial population. This population consists of particles that are uniformly distributed within the search space, where each particle is represented by a concentration vector that defines its position and behavior in the optimization process. The initial population is generated from

$$P_i^{\text{initial}} = P_{\text{min}} + \text{rand}_i \left(P_{\text{max}} - P_{\text{min}}\right) \quad i = 1, 2, ..., n \tag{16}$$

Where, **Pi** initial is the vector corresponding to the initial concentration of particle **i**, **Pmax** and **Pmin** are the upper and lower bounds, respectively, **n** is the number of particles in the population, and **randi** generates a random value [0,1].

Equilibrium pool and candidates :

EO algorithm uses a pool of five particles, consisting of the best four so far for diversification, and their average is used for the exploitation process to reach the optimal solution.The pool is defined as

$$P_{eq} = [P_{eq}(1), P_{eq}(2), P_{eq}(3), P_{eq}(4), P_{eq}(avg)] \tag{17}$$

Concentration update :

At each iteration, the EO updates the particle population through the following equation :

$$\overrightarrow{P} = \overrightarrow{P_{\text{eq}}} + (\overrightarrow{P} - \overrightarrow{P_{\text{eq}}})\overrightarrow{F} + \frac{\overrightarrow{R}}{\overrightarrow{\lambda}}(1 - \overrightarrow{F}) \tag{18}$$

- $\overline{F}$: Influences the balance between exploration and exploitation.
- $\overline{\lambda}$: A random value within the range $[0, 1]$.
- t : A value that decreases as the iteration number $(iter)$ increases.
- $\overline{R}$: Referred to as the generation rate.

[28] .

flow chart of the EO are presented in Figure II.5

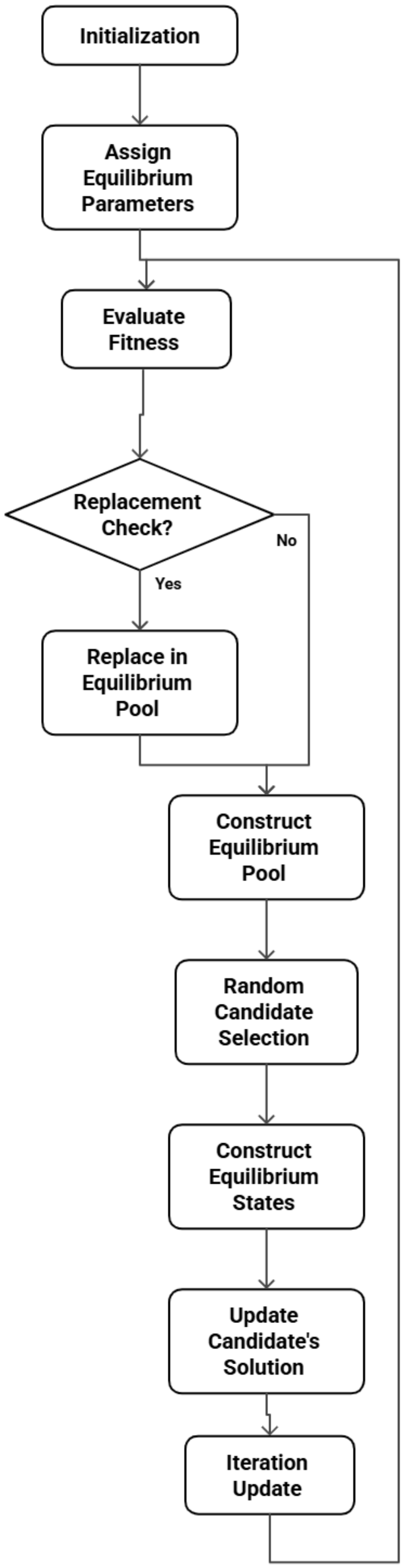


FIGURE – II.5 : Flowchart of EO

II.5.3 Slime Mould Algorithm (SMA) :

The slime mould Physarum polycephalum is a plasmodial amoeboid eukaryotic organism that can take on so many varied forms throughout its life history. They range from an amoeboid single-cell stage, through a tubular network with foraging capability, to a mature cable structure with tip-closed lumens on its branches. These networks exhibit intelligent reactions to environmental stimuli and display complex adaptive behavior, particularly in their foraging behavior. Its.-large, multinucleated plasmodial cells that span large areas render P. polycephalum a superb model system for studying emergent behavior and the dynamics of active matter. Its environmentally controlled behavior presents rich opportunities for scientific study in a relatively simple organism [29] [30] [31] [32].

flow chart of the EO are presented in Figure II.5

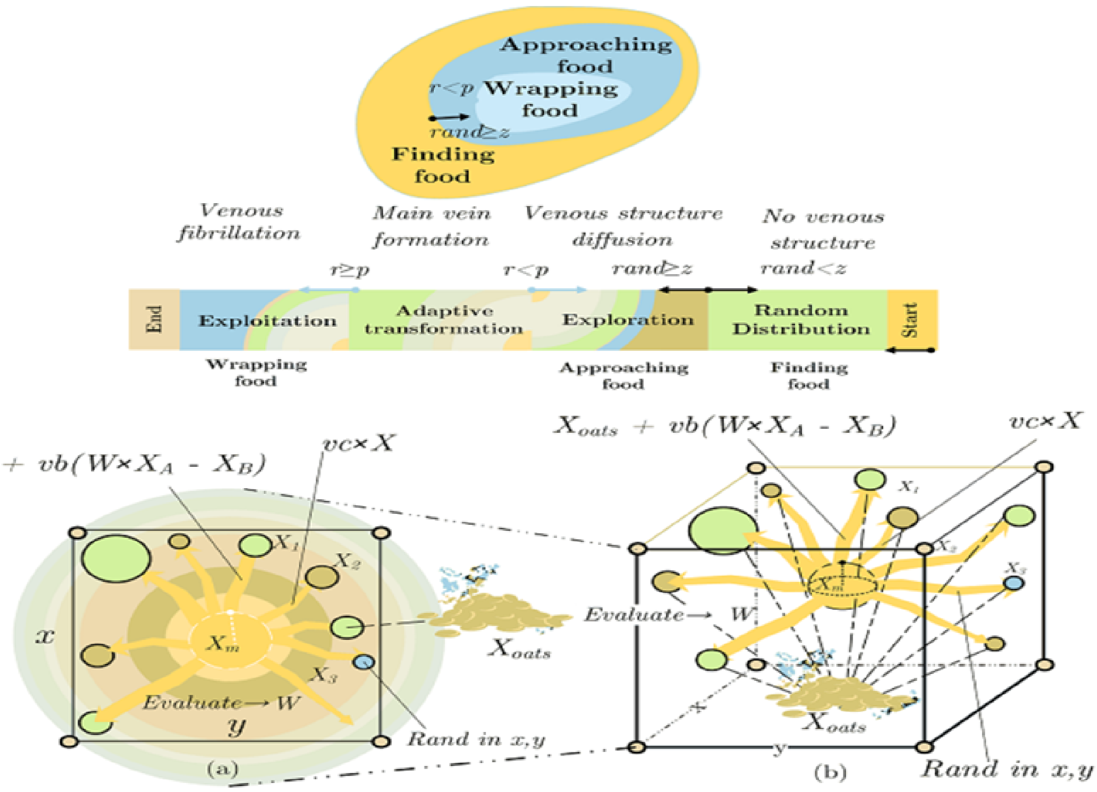


FIGURE – II.6 : Methode SMA

The mathematical model used to update the slime mould's position is represented as follows :

$$\overrightarrow{X(t+1)} = \begin{cases} \text{rand} \cdot (UB - LB) + LB & \text{rand} < z \\ X_b(t) + \vec{vb} \cdot \left(\vec{W} \cdot \overrightarrow{X_A(t)} - \overrightarrow{X_B(t)} \right) & r < p \\ \vec{vc} \cdot \overrightarrow{X(t)} & r \geq p \end{cases} \tag{19}$$

- LB and UB : Represent the lower and upper bounds of the search space.
- rand and r : Random values within the range $[0, 1]$.
- $z = 0.03$: A fixed parameter used in the model.
- $X(t)$: Position of the strongest food odor at the current moment.
- $\overline{vb}$: A parameter that varies within the range $[-a, a]$.
- $\overline{vc}$: A parameter that decreases linearly from 1 to 0 with iterations.
- W : The weight of the slime mould.
- $X(t)$ and $X_B(t)$: Positions of two randomly selected agents in the population.
- (t) : The current position of the slime mould.

$$p = \tanh |S(i) - DF| \tag{20}$$

where $\underline{i} \in 1, 2, \ldots, n$, $S(i)$ denotes the fitness $\overline{X}$ and DF represents the best fitness obtained to date.

$$vb = [-a, a] \tag{21}$$

$$a = \left(- \left(\frac{t}{max\_t} \right) + 1 \right) \tag{22}$$

where max_t indicates the maximum number of iterations.

$$vc = [-b, b] \tag{23}$$

$$b = 1 - \left(\frac{t}{max\_t} \right) \tag{24}$$

The slime mould's weight ${}^2W$ is calculated using the following equation :

$$W\left(\text{SmellIndex}(i)\right) = \begin{cases} 1 + r_1 \cdot \log\left(\frac{bF - S(i)}{bF - wF} + 1\right) & \text{condition} \\ 1 - r_1 \cdot \log\left(\frac{bF - S(i)}{bF - wF} + 1\right) & \text{other} \end{cases} \tag{25}$$

- **condition** : Represents whether $S(i)$ is ranked in the top half of the population.
- F : A random number in the range $[0, 1]$.
- DF : The best fitness value obtained in the current iteration.
- wF : The worst fitness value obtained in the current iteration.
- **SmellIndex** : Indicates the result of sorting fitness values in ascending order (in a minimization problem) [33].

flow chart of the SMA are presented in Figure 2.7

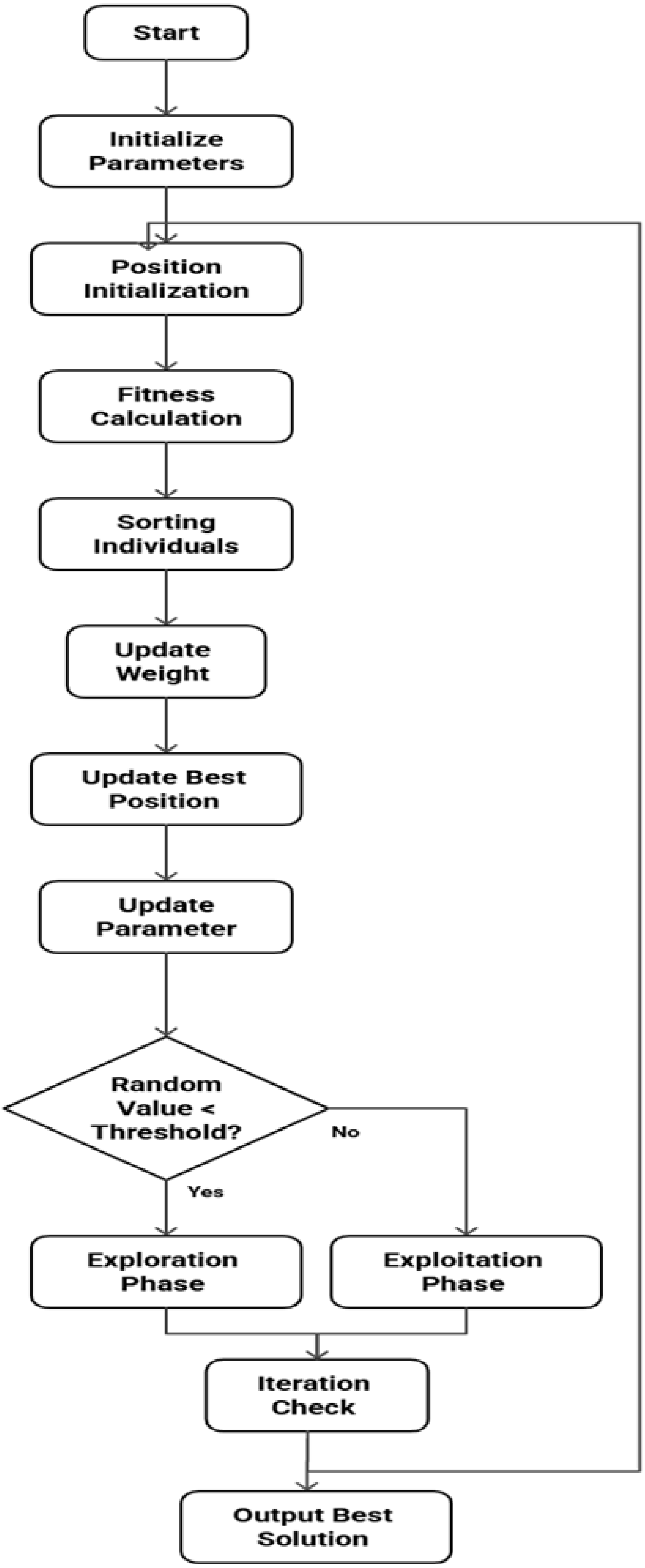


FIGURE – II.7 : Flowchart of SMA

II.6 : Comparative Analysis of Modified GNDO, EO, and SMA :

We compared these three metaheuristic algorithms in several key aspects. Table II.1 illustrates this comparison.

Parameter	GNDO	Equilibrium Optimizer (EO)	Slime Mould Algorithm (SMA)
Scientific Foundation	Based on the Generalized Normal Distribution	Based on physical principles of chemical equilibrium	Inspired by slime mould foraging behavior
Operational Mechanism	Focuses on parameter optimization of probability distributions	Regulates particle density within the search space	Generates adaptive networks in response to environmental stimuli
Key Strengths	- Highly mathematically precise - Uses GPU acceleration for faster calculation	- Light and fast - Simple to deploy	- Highly adaptive - Leverages biological intelligence
Major Limitations	Requires statistically significant distributed data	Best suited for complex optimization problems	Needs careful tuning of control parameters

TABLE II.1 – Comparative Overview of GNDO, EO, and SMA

II.7 Control of BLDC Motor with GNDO, EO, and SMA :

Figure II.8 presents a flowchart illustrating the optimization strategy applied to the tuning of PID controller parameters. In this approach, advanced metaheuristic algorithms) GNDO, EO, and SMA(were employed to enhance the control performance of a BLDC motor.

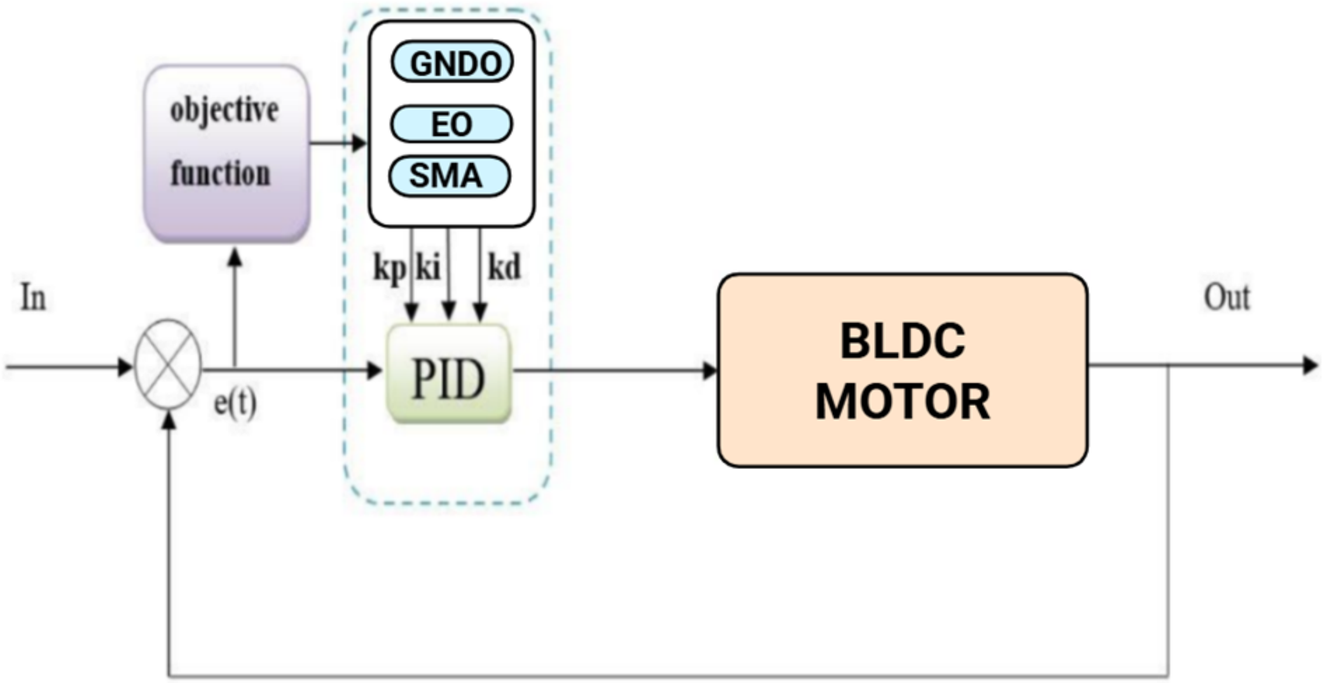


FIGURE – II.8 : optimization shema of PID controller with GNDO, EO, and SMA

The error e(t). where the objective function is ITAE.

$$\text{ITAE} = \int_0^t t|\epsilon(t)|\, dt \tag{26}$$

II.8 SIMULATION RESULTS :

In this section, a Brushless DC (BLDC) motor was tested under various operating conditions, including no-load, loaded, and reverse speed scenarios. These tests aimed to assess the robustness and stability of the control system when optimized using metaheuristic algorithms such as GNDO, EO, and SMA, and to compare their performance against conventional tuning methods. To ensure optimal PID controller performance, multiple simulations were carried out using MATLAB/SIMULINK, through which the most suitable parameter values were determined :

Parameter	**GNDO**	**EO**	**SMA**	**PID Con**
Population Size	20	20	20	/
Iteration	20	20	20	/
higher Bound	[500, 3000 0.5]	[500, 3000 0.5]	[500, 3000 0.5]	/
Lower Bound	[0.1, 0.1, 0.01]	[0.1, 0.1, 0.01]	[0.1, 0.1, 0.01]	/
Best fitness	0.00530301217	0.00529845521	0.00530785413	/
$\mathbf{K_P}$	358.43654472	479.17941996	23.50577310	51.36
$\mathbf{K_I}$	64.56986104	87.11396150	4.41693252	2153.45
$\mathbf{K_D}$	0.15586214	0.21386748	0.01008437	0.306

TABLE II.2 – The choice of GNDO, EO, SMA and PID parameter

- The fitness curve is shown below for each of the GNDO, EO and SMA methods :

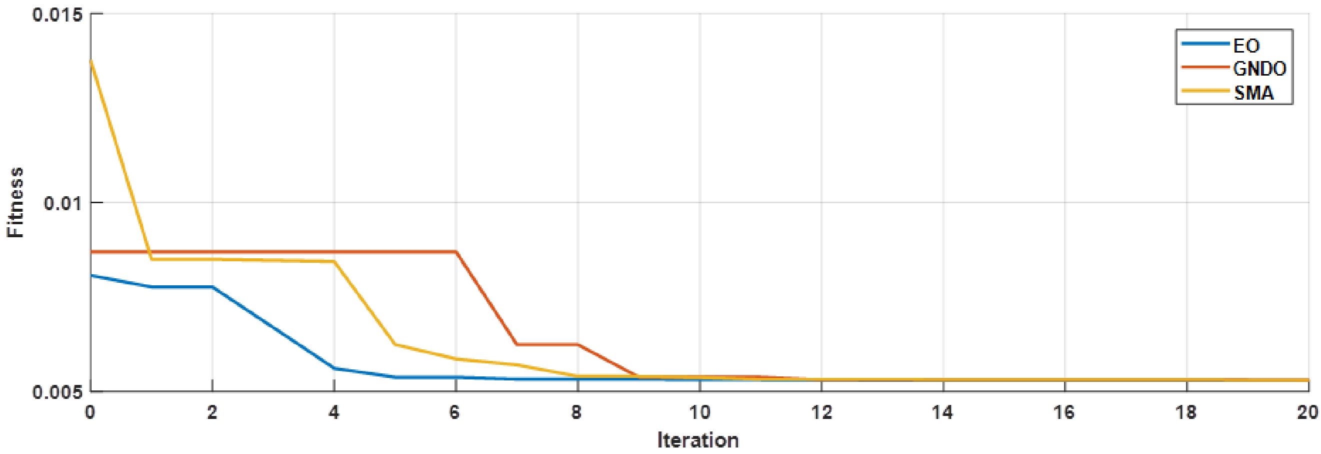


FIGURE – II.9 Fitness function curve of GNDO ,EO and SMA

No load :

• Empty test :

Value the rotation of the speed is : 1000 rad/s

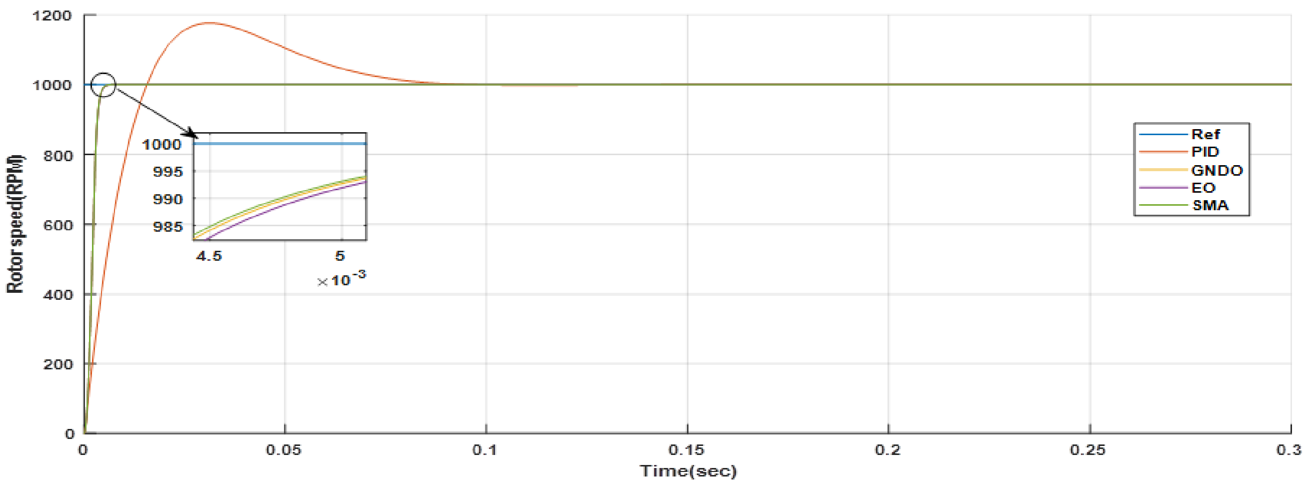


FIGURE – II.10 : speed response obtained by using GNDO, EO, SMA and Z-N PID in Empty test

• Reverse speed test :

To carry out this test, the direction of rotation of the speed is reversed from 1000 rad/s to -1000 rad/s at $t = 0.15$, s

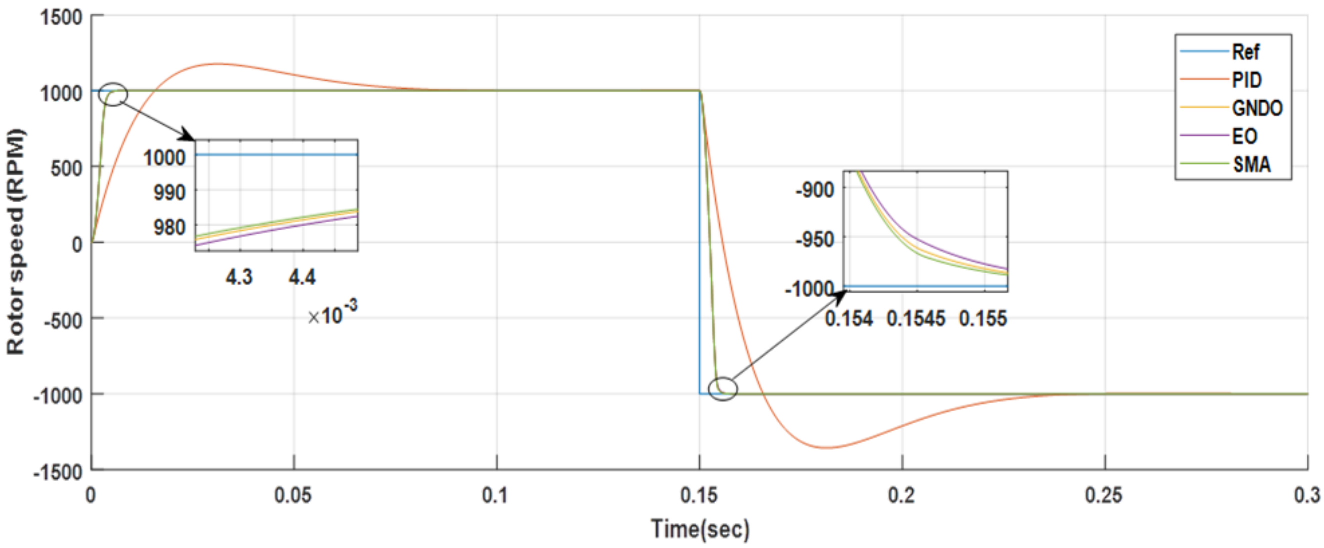


FIGURE – II.11 : speed response obtained by using GNDO, EO, SMA and Z-N PID in Reverse speed test

With load :

To carry out this test, $T_m = 5\,\mathrm{N{\cdot}m}$ at $t = 0.1\,\mathrm{s}$.

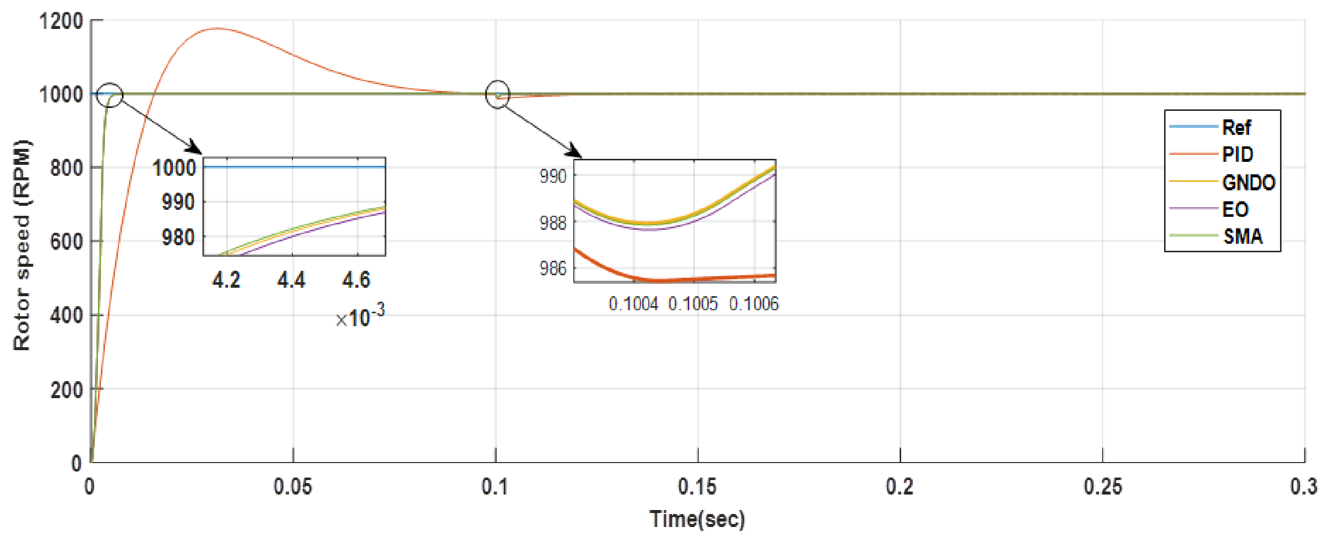


FIGURE – II.12 : speed response obtained by using GNDO, EO, SMA and Z-N PID in With load operation

• Comments and interpretations :

The figure illustrates II.9 the evolution of the fitness value over several iterations for three metaheuristic optimization algorithms : EO, GNDO, and SMA.The EO algorithm demonstrated the fastest convergence rate, with a sharp drop in fitness value during the initial iterations. This indicates high efficiency in quickly reaching a good solution.The SMA and GNDO algorithms also performed well but with a slower convergence rate compared to EO. Both eventually stabilized near the minimum fitness value, but only after more iterations.All three algorithms reached nearly equal results in terms of final fitness value, highlighting their effectiveness in solving the optimization problem.

No load :

• Empty test :
In Figure II.10, the GNDO, EO, and SMA techniques demonstrate better performance through faster response to the reference speed, minimal overshoot, and very short settling time. In contrast, the PID controller exhibits significant overshoot, noticeable oscillations, and a long settling time, making it the least efficient in motor speed control. The SMA technique, in particular, shows a good initial response, being slightly faster than the other techniques at the beginning.

• Reverse speed test :
Figure II.11 shows that in Reverse speed test, the three techniques (GNDO, EO, and SMA) prove to be more effective in achieving fast and stable control compared to the PID controller, due to its slower response and larger oscillations. Additionally, the SMA technique appears to be slightly faster than the other techniques.

With load :

Similarly ,in Figure II.12, the techniques (GNDO, EO, and SMA) appear to be superior in terms of reaching the reference speed and settling time compared to the PID controller. However, the effect of the load disturbance at 0.1 seconds causes a slight drop in speed across all techniques, but stability is quickly regained afterward. It is also observed that the SMA technique was slightly faster than the other techniques in returning to the reference speed.

• To evaluate the capability of the selected metaheuristic optimization algorithms in enhancing the performance of the PID controller, a comparative analysis was conducted between the algorithms (GNDO, EO, and SMA) and the conventional PID controller using different population sizes. Their effectiveness was assessed based on key control performance criteria. Table (II.3) presents the results of this analysis.

Criteria	**GNDO**	**EO**	**SMA**	**PID**
Rise time (s)	0.0024	0.0025	0.0024	0.0116
Overshoot percentage (%)	0.0414	0.0675	0.0492	17.6184
Settling time (s)	0.0043	0.0044	0.0043	0.0744

Table II.3 – Parameters and performance for metaheuristic algorithms and PID

The table clearly shows the superiority of the GNDO, SMA, and EO algorithms over the traditional PID controller in almost all performance criteria.

II.9 Conclusion

In this chapter, we presented the GNDO, EO, and SMA algorithms, which were used to tune and improve the gains of the PID controller. During the optimization process, proposed solutions were evaluated based on the quality of the speed response, and the simulation results showed that these algorithms are powerful and efficient in quickly and accurately finding optimal solutions. What distinguishes these stochastic methods is their ability to avoid the complexity of traditional arithmetic operations. Compared to conventional methods, metaheuristic algorithms proved to be superior in both solution speed and control efficiency. In the following, we will explore a practical application through a project aimed at controlling a BLDC motor.

CHAPITRE III

Application and Real-Time Control of BLDC Motor

III.1.Introduction

Here, emphasis has been given on experimental implementation of a control strategy of a Brushless DC (BLDC) motor by open-loop methods. Supplementing the theoretical background and simulation findings placed within the earlier parts of this thesis , this part aims to transfer those concepts to real practice. Implementation was carried out in two ways : a simulation in the Proteus software and a hardware implementation with an Arduino Uno, an Electronic Speed Controller (ESC), and a potentiometer.

The primary aim of this chapter is to demonstrate the ability to manually control the motor speed by modifying the duty cycle of a PWM signal, generated by the Arduino and fed into the ESC. The analog input of the potentiometer is the controlling variable in this configuration. While the system is open-loop—there is no live position or speed sensor—the action and speed variation of the motor can still be observed meaningfully both through serial observation and direct visual inspection.
This chapter is organized in the following way : it initially depicts the hardware and software infrastructures utilized in experimental and simulated settings. Then, it presents the implementation steps and results from both environments. The final sections are devoted to speed behavior analysis based on input-output relations, supplemented with graphical figures and important discussion. Through this working example, the ability and limitations of open-loop BLDC motor control are explored as a stepping stone towards more sophisticated closed-loop systems in the ensuing research.

III.2.Simulation of BLDC Motor Control in Proteus

III.2.1 Simulation Setup

A simulation was developed using the Proteus software to model an open-loop BLDC motor control system. The control logic was implemented using an Arduino Uno which managed the six-step commutation of the motor by sending control signals to IR2104 MOSFET driver ICs. Additional input components such as a potentiometer and two push buttons were integrated to manage speed and start/stop operations. The circuit also included a virtual oscilloscope for signal observation.

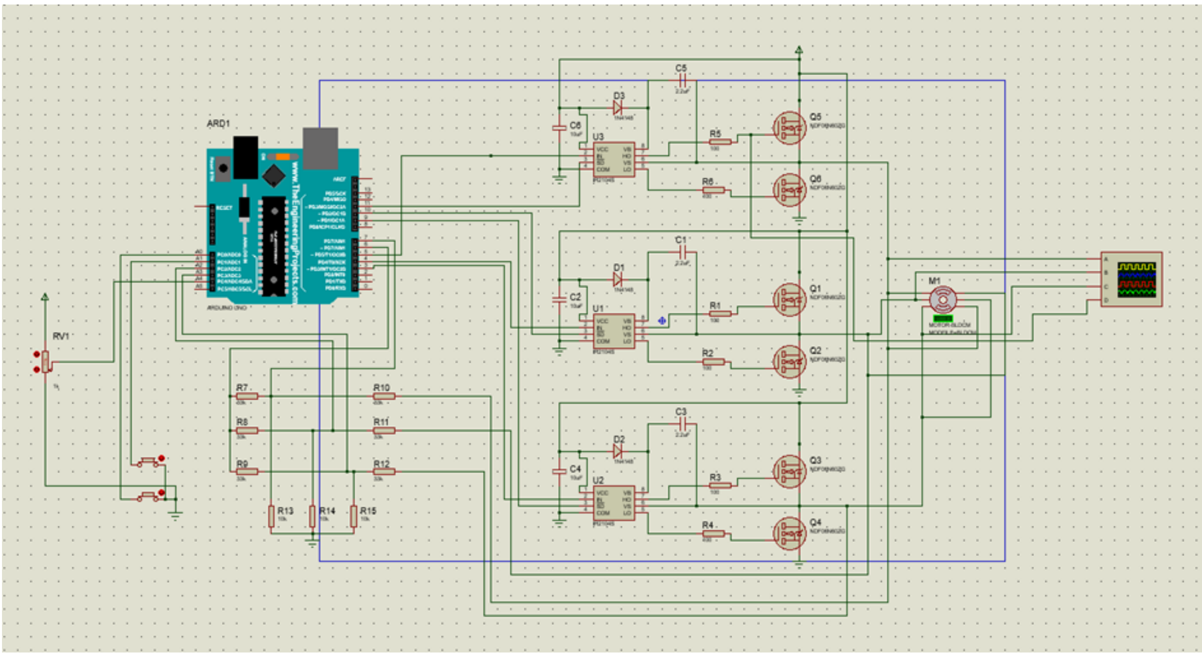


FIGURE – III.1 Arduino-Based Sensorless BLDC Motor Control Circuit in proteus

III.2.2 Arduino Code for Proteus Simulation

The Arduino code controlled the six-step commutation via sequenced phase energization within the motor. Speed was controlled by varying the delay between steps based on potentiometer input. Start and stop were handled via the use of buttons.

```
// Phase high-side control (connected to IN of IR2104)
const int PHASE_UH = 3;
const int PHASE_VH = 4;
const int PHASE_WH = 5;

// SD pins (enable) of each IR2104
const int SD_U = 9;
const int SD_V = 10;
const int SD_W = 11;

// Possibly low-side enable signal
const int LOW_ENABLE = 6;

// Input controls
const int SPEED_POT = A4;
const int START_BUTTON = A0;
const int STOP_BUTTON = A1;

bool motorRunning = false;

void setup() {
  // Outputs for high-side signals
  pinMode(PHASE_UH, OUTPUT);
  pinMode(PHASE_VH, OUTPUT);
```

```
  pinMode(PHASE_WH, OUTPUT);

  // Outputs for SD (enable) of IR2104
  pinMode(SD_U, OUTPUT);
  pinMode(SD_V, OUTPUT);
  pinMode(SD_W, OUTPUT);

  // Possibly common enable for low-side drivers
  pinMode(LOW_ENABLE, OUTPUT);

  // Set SD and low-side enable HIGH to allow switching
  digitalWrite(SD_U, HIGH);
  digitalWrite(SD_V, HIGH);
  digitalWrite(SD_W, HIGH);
  digitalWrite(LOW_ENABLE, HIGH);

  // Inputs
  pinMode(START_BUTTON, INPUT_PULLUP);  // active LOW
  pinMode(STOP_BUTTON, INPUT_PULLUP);   // active LOW

  Serial.begin(9600);
}

void loop() {
```

```
  if (digitalRead(START_BUTTON) == LOW) {
    motorRunning = true;
  }

  if (digitalRead(STOP_BUTTON) == LOW) {
    motorRunning = false;
    allOff();
  }

  if (motorRunning) {
    int potValue = analogRead(SPEED_POT);
    int delayTime = map(potValue, 0, 1023, 30000, 3000); // microseconds

    step1(); delayMicroseconds(delayTime);
    step2(); delayMicroseconds(delayTime);
    step3(); delayMicroseconds(delayTime);
    step4(); delayMicroseconds(delayTime);
    step5(); delayMicroseconds(delayTime);
    step6(); delayMicroseconds(delayTime);
  }
}

void allOff() {
  digitalWrite(PHASE_UH, LOW);
```

```
  digitalWrite(PHASE_UH, LOW);
  digitalWrite(PHASE_VH, LOW);
  digitalWrite(PHASE_WH, LOW);
}

void step1() { allOff(); digitalWrite(PHASE_UH, HIGH); digitalWrite(PHASE_VH, LOW); }
void step2() { allOff(); digitalWrite(PHASE_UH, HIGH); digitalWrite(PHASE_WH, LOW); }
void step3() { allOff(); digitalWrite(PHASE_VH, HIGH); digitalWrite(PHASE_WH, LOW); }
void step4() { allOff(); digitalWrite(PHASE_VH, HIGH); digitalWrite(PHASE_UH, LOW); }
void step5() { allOff(); digitalWrite(PHASE_WH, HIGH); digitalWrite(PHASE_UH, LOW); }
void step6() { allOff(); digitalWrite(PHASE_WH, HIGH); digitalWrite(PHASE_VH, LOW); }
```

III.2.3 Simulation Observations

The Arduino control signals were used to switch the motor phases sequentially through the IR2104 drivers. The delay between steps was varied with the potentiometer, adjusting the rotation speed. Push buttons were used to allow manual control of motor running. The MOSFETs worked as expected, and protection diodes enabled safe operation in back EMF simulation.

III.2.4 Oscilloscope Analysis of Motor Phases

For verification of correct signal behavior, a virtual oscilloscope was connected to phase outputs of the motor. The waveforms exhibited the expected six-step trapezoidal commutation pattern with 120° phase separation confirming right logic and timing.

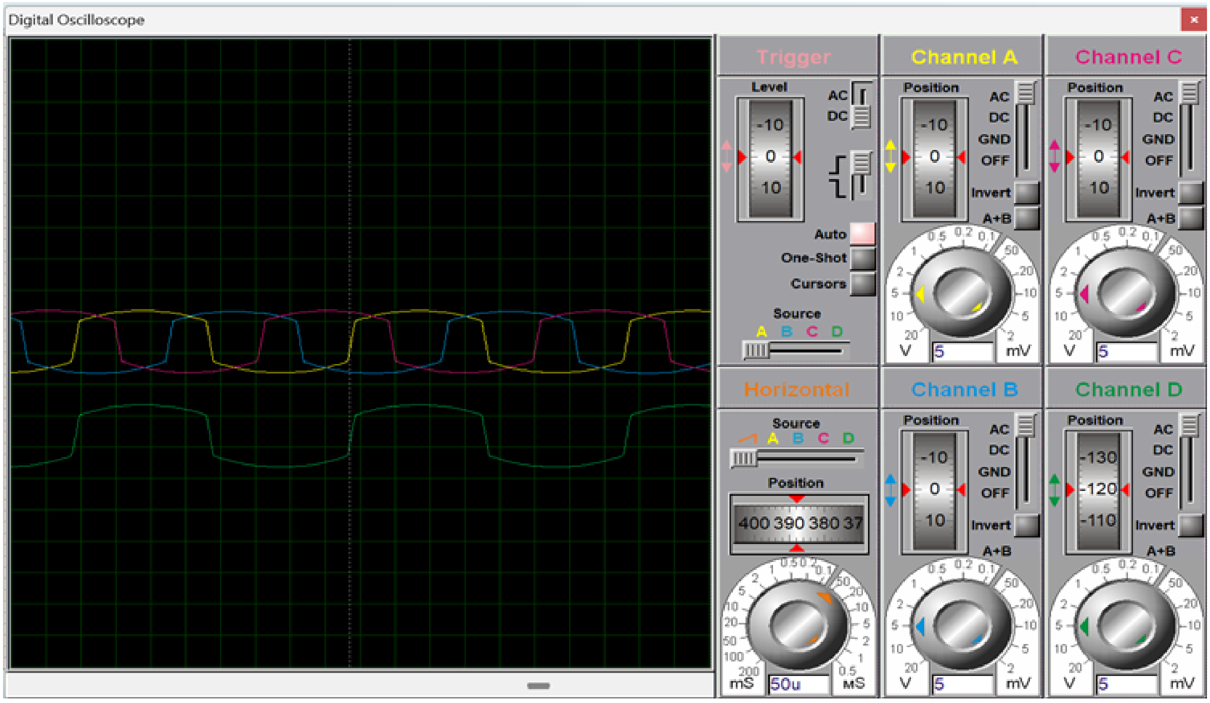


FIGURE – III .2 Proteus Oscilloscope Waveforms of Three Phases (A, B, C)

III.3 Real Hardware Implementation :

FIGURE – III.3 –Real Hardware Setup of BLDC Motor

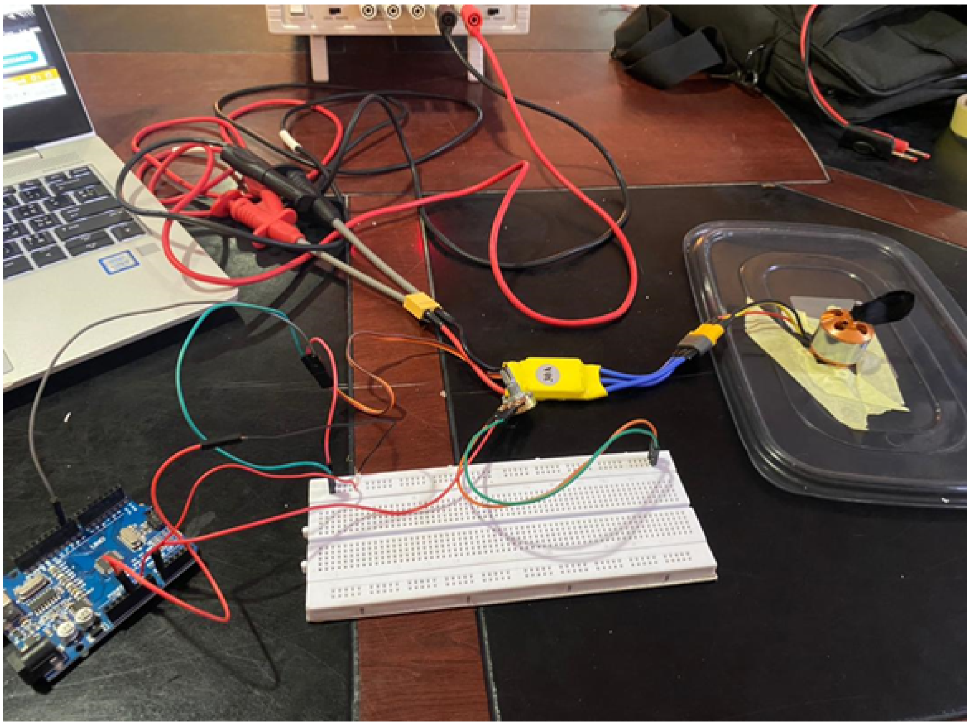


FIGURE – III.4 Real Hardware Setup of BLDC Motor with Arduino

III.3.1 Components Used :

- Arduino Uno microcontroller
- A2212/15T 930KV BLDC motor
- Electronic Speed Controller (ESC) compatible with the motor
- Potentiometer for speed input(connected to analog pin A0)
- DC Power supply (11.77 V and 5.0 V tested)
- Connecting cables and breadboard

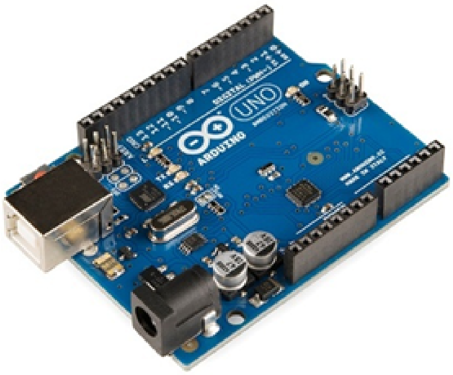
III.3.2 Arduino Code for ESC Control :

The Arduino uses the Servo library to generate PWM signals to the ESC. The potentiometer input is read via analog input pin A0, mapped from 0–1023 to 0–180 (PWM range), and written to the ESC on pin 9 to control the motor speed.

```
#include <Servo.h>

Servo ESC;

int Speed;

void setup() {
  ESC.attach(9, 1000, 2000);    // Attach ESC to pin 9 with min/max pulse width
  Serial.begin(9600);           // Start serial communication at 9600 baud
}

void loop() {
  Speed = analogRead(A0);                 // Read analog value from pin A0
  Speed = map(Speed, 0, 1023, 0, 180);    // Map it to 0-180 range
  ESC.write(Speed);                       // Write value to ESC

  // Print to Serial Monitor and Plotter
  Serial.print("Mapped Speed: ");
  Serial.println(Speed);                 // This will plot on Serial Plotter too

  delay(100);  // Small delay to make data easier to read
}
```

III.4.Real Hardware Implementation

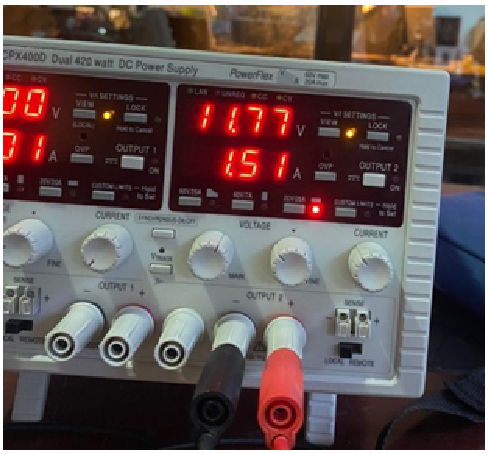
In this setup, the Arduino Uno reads analog input from a potentiometer and maps it to a PWM signal using the Servo library. This signal is sent to the ESC to control the speed of the BLDC motor. Additionally, it outputs data to the Serial Monitor for real-time monitoring.



III.4.1 BLDC Motor (A2212/15T 930KV)

The motor is powered through an ESC controlled by Arduino PWM. Tests were conducted at two voltages :

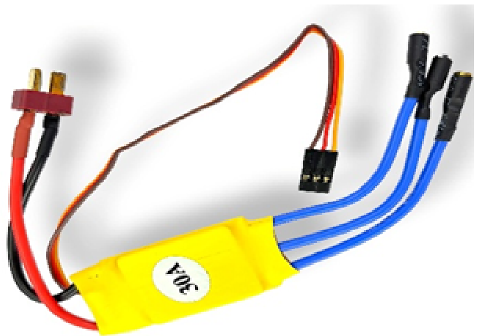
- 11.77 V supply : Current approximately 0.51 A ; motor runs faster.



• 5.0 V supply : Reduced speed and current, illustrating voltage's impact.

III.4.2 Electronic Speed Controller (ESC)

The ESC receives PWM signals from Arduino to adjust motor speed. It eliminates the need for complex phase switching in the Arduino code by managing motor phases automatically.



III.4.3 Potentiometer

A potentiometer serves as a user input device to control the motor speed by varying the voltage sensed by the Arduino analog pin.

III.5 Serial Monitoring and Speed Control Analysis

To observe the system behavior, the Arduino Serial Monitor was used to display real-time analog values from a potentiometer and their corresponding mapped speed values sent to the ESC. The analog values (ranging from 0 to 1023) were read using the analogRead function and converted into PWM speed values (0 to 255) using the map function. This process shows how the potentiometer input is translated into PWM signals that control the motor speed. A screenshot of the Serial Monitor is shown in Figure III.5. Based on the recorded data, a graph Figure III.6 was created to analyze the relationship between the analog input and the resulting speed values. This graphical representation illustrates the linearity of the control and confirms that the mapping and motor response are functioning correctly.

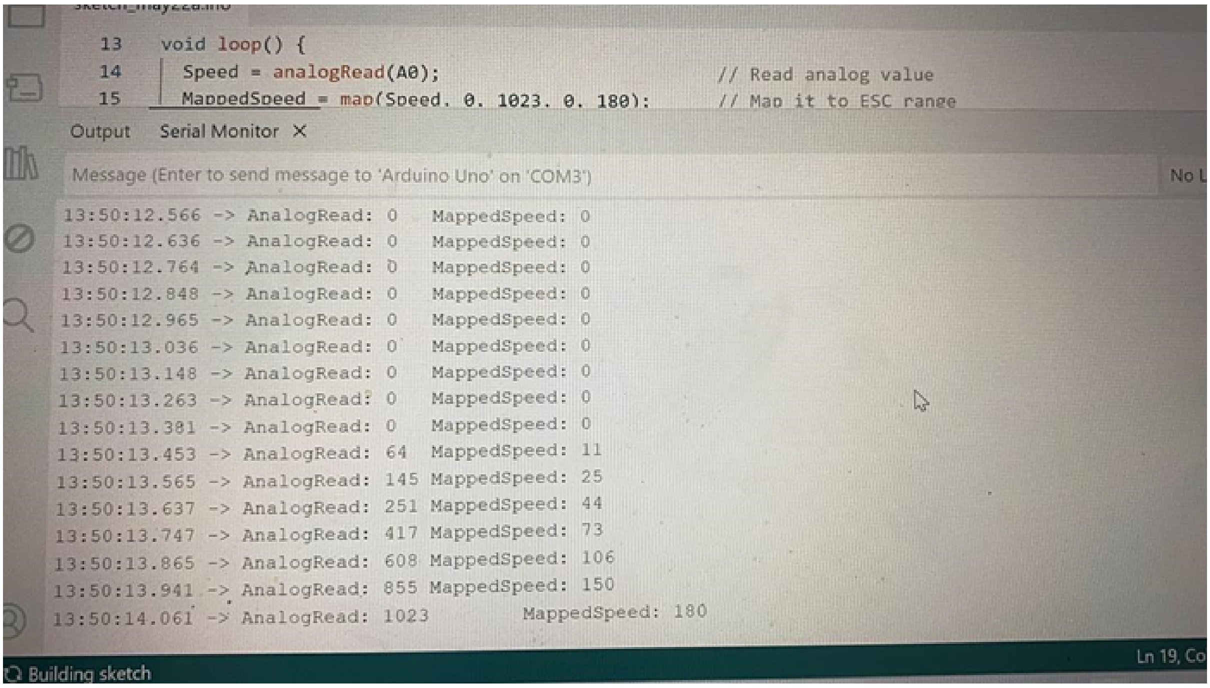


FIGURE – III.4 Serial Monitor Output with 11.77 V Power Supply

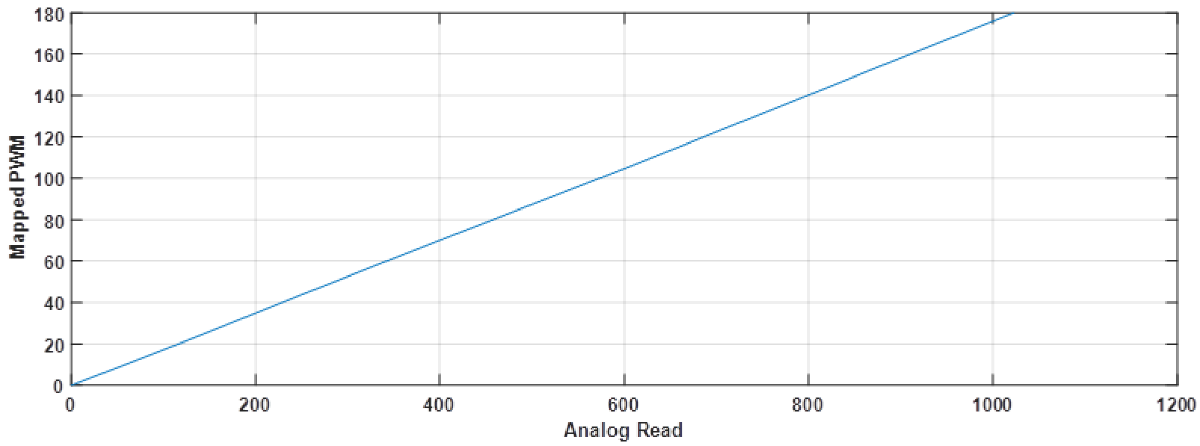


FIGURE – III.5 Graph of AnalogRead Input and PWM Speed Command

This graph confirms the proportional speed control using the potentiometer in an open-loop configuration.

III.6 Observations and Discussion

- The motor speed increases proportionally with potentiometer rotation.
- Higher power supply voltage leads to higher speed and current draw.
- The ESC simplifies implementation by handling motor commutation internally.
- Real-time serial output makes debugging and behavior analysis easier.
- The entire system operates in open-loop, with no feedback sensor.

III.7 Conclusion

This chapter successfully demonstrates open-loop control of a Brushless DC (BLDC) motor through both simulation and real-world implementation. The Proteus simulation provided valuable insights into phase commutation and waveform patterns using IR2104 drivers and an Arduino Uno. In the hardware setup, the BLDC motor was controlled using a standard Electronic Speed Controller (ESC), with speed variation achieved through a potentiometer and PWM signals generated by Arduino.

The results confirmed that even without feedback sensors, reliable motor speed control is possible by adjusting the PWM duty cycle in an open-loop system. The motor speed responded proportionally to the analog input, and the behavior aligned well with theoretical expectations.

However, the system has clear limitations, primarily due to the absence of real-time feedback, which restricts its ability to adapt to load changes or achieve precise speed regulation. Therefore, future work is proposed to enhance the system by integrating a closed-loop control approach using sensors such as Hall effect sensors or rotary encoders for real-time speed measurement and feedback.

Furthermore, an important focus of future development will be improving the PID controller performance using metaheuristic algorithms such as the Generalized Normal Distribution Optimization (GNDO), Equilibrium Optimizer (EO), and Slime Mould Algorithm (SMA). These algorithms help automatically tune PID parameters to achieve smoother and more accurate speed control, reduce overshoot, and improve system stability, especially under dynamic conditions.

By combining sensor feedback with optimized PID tuning using these metaheuristic algorithms, the system will be capable of delivering high-performance motor control suitable for advanced applications like robotics, electric vehicles, and industrial automation.

General Conclusion

This thesis presented a comprehensive study on the speed control of a Brushless DC (BLDC) motor using an optimized PID controller. The work was structured across three major axes : modeling, classical and intelligent control design, and real-time application.

In the first phase, we developed a detailed mathematical model of the BLDC motor and implemented a classical PID controller. The initial tuning was performed using the Ziegler-Nichols method, which provided baseline control. However, simulation results revealed limitations such as slower response, overshoot, and sensitivity to disturbances.

To address these issues, we introduced metaheuristic optimization techniques to automatically tune the PID gains. The optimized controller significantly improved the system's dynamic performance, including faster response, better speed tracking, and enhanced robustness under variable load and speed conditions.

In the final phase, the controller was implemented in a real-time environment using Proteus simulation, with practical integration that demonstrated the feasibility of applying the proposed system beyond MATLAB/Simulink. Although the real-time application was carried out in an open-loop configuration, the results provided important insights into system behavior and control effectiveness.

Overall, this work demonstrated the effectiveness of combining classical control theory with modern optimization algorithms, both in simulation and practical environments. It opens promising perspectives for intelligent, adaptive control of BLDC motors in real-world applications.

Future Work :

Although the results of this study are promising, several key areas remain to be explored and developed :

Closed-Loop Real-Time Control : Future work will focus on transitioning from open-loop to closed-loop control in the real-time implementation. This will allow dynamic feedback correction and ensure better stability and accuracy under varying operating conditions.

Metaheuristic Integration in Real-Time : While metaheuristic algorithms were applied successfully in simulation, the next step is to integrate them into the real-time closed-loop system. This will enable automatic tuning and adaptive control based on real-time motor behavior.

Implementing metaheuristic algorithms in real-time would also make it possible to develop an online adaptive PID controller, where the parameters are constantly updated based on system performance and external disturbances. This would be particularly useful in applications where system dynamics are non-linear or vary with time

BIBLIOGRAPHY

[1] Bokhari, N., Drissi,G., Rahoua , N. (2023). *Speed Control of BLDC Motor using Fuzzy Logic* (Mémoire de Master, University Mohamed Khider Biskra).

[2] Tahri, A., Benyahia, A. (2022). *Modélisation et réglage optimal du gain du variateur de vitesse PID pour moteur BLDC en utilisant la technique PSO* (Mémoire de Master, Université Kasdi Merbah Ouargla). Consulté sur DSpace Ouargla.

[3] Ministère de l'Éducation nationale, de la Jeunesse et des Sports, France. *Modélisation des moteurs Brushless* https ://sti.eduscol.education.fr/sites/eduscol.education.fr.sti/files/ ressources/pedag ogiques/9005/9005-moteurs-brushless-v3.pdf

[4] S,Rambabu., B, D, Subudhi. (2007). *MODELING AND CONTROL OF A BRUSHLESS DC MOTOR* (Mémoire de Master, Department of Electrical Engineering National Institute of Technology Rourkela).

[5] DIVI ,B., BODAPATI, V ,R., NAGESWARA ,R ,K (2022). *SPEED CONTROL OF BLDC MOTOR USING ADAPTIVE FUZZY PID CONTROLLER* (SCHOOL OF ELECTRICAL SCIENCES HINDUSTAN INSTITUTE OF TECHNOLOGY AND SCIENCE).

[6] Zhao, Jian ., Yu, Yangwei (2011). *Brushless DC motor fundamentals* organization = Monolithic Power Systems.

[7] Chang-liang Xia (2012). *Permanent Magnet Brushless DC Motor Drives and Controls* ,publisher = John Wiley & Sons Singapore Pte. Ltd.

[8] Alfaro, V. M. , Vilanova, R.(2022)*PID control : Resilience with respect to controller implementation* ,journal =Frontiers in Control Engineering,pages = Article 1061830.

[9] GHOUL, M, AIMENE ,M., AMIEUR ,T (2022). *Modelling and Fractional Order Control of DC motor* (Mémoire de Master, Larbi Tébessi University -Tébessa).

[10] Carpanzano, E., Knüttel, D. (2022). Advances in Artificial Intelligence Methods Applications in Industrial Control Systems : Towards Cognitive Self-Optimizing Manufacturing Systems. Applied Sciences, 12(21), 10962. https ://doi.org/10.3390/ app122110962

[11] Elakkiya, R., Rajasekar, N. (2022). An overview of smart irrigation systems using IoT. Nexus, 1(3), 100124. https ://doi.org/10.1016/j.nexus.2022.100124

[12] Stephen, B,J. , Emmanuel ,G ,d., Afeez,A. (2022)*Metaheuristic algorithms for PID controller parameters tuning : review,approaches and open problems* ,journal =Heliyon e09399.

[13] Patil, S. S., Deshmukh, R. S., Holambe, R. S., Dudul, S. V. (2021).*A* review on smart irrigation system using IoT. In S. Rajasekaran, G. Balasubramanian, P. K.

Srijith, K. Satheesh Kumar (Eds.), Intelligent Communication, Control and Devices (pp. 13–24). Springer. https ://doi.org/10.1007/978-981-33-4295-8_2

[14] Gokul, S.T., Mehdi. S., Elmira ,J. Ben, Horan., Saad ,M.,Alex, S. (2022).*R*ole of optimization techniques in microgrid energy management systems—A review.journal =Energy Strategy Reviews.

[15] K.K ,Ramachandran, (2023). *OPTIMIZING IT PERFORMANCE : A COMPREHENSIVE ANALYSIS OF RESOURCE EFFICIENCY* (International Journal of Marketing and Human Resource Management (IJMHRM)).

[16] Adam ,Z., , Xin ,Yao., Jacek ,M (2025). *Migrant Selection in Island-Based Optimization* (Warsaw University of Technology, Faculty of Mathematics and Information Science,School of Data Science, Lingnan University, Hong Kong SAR,Faculty of Computer Science, AGH University of Krakow, Krakow, Poland).

[17] Xiangzhou ,G., Shenmin ,S. (2024).*A* switching competitive swarm optimizer for multi-objective optimization with irregular Pareto fronts.journal =Expert Systems with Applications.

[18] Das, S., Dhang, N. (2022). *D*amage identification of thin plates using multi-stage PSOGSA and incomplete modal data. Applied Mathematics in Science and Engineering, 30(1), 397–439. https ://doi.org/10.1080/27690911.2022.2080206

[19] Khodadadi, N., Khodadadi, E., Abdollahzadeh, B. et al. *M*ulti-objective generalized normal distribution optimization : a novel algorithm for multi-objective problems. Cluster Comput 27, 10589–10631 (2024). https ://doi.org/10.1007/s10586-024-04467-7

[20] Zhang, Y. *A*n improved generalized normal distribution optimization and its applications in numerical problems and engineering design problems. Artif Intell Rev 56, 685–747 (2023). https ://doi.org/10.1007/s10462-022-10182-9

[21] Ahmed, S., Sheikh, K. H., Mirjalili, S., Sarkar, R. (2022).*B*inary Simulated Normal Distribution Optimizer for feature selection : Theory and application in COVID-19 datasets. Expert Systems with Applications, 200, 116834. https ://doi.org/10.1016/j.eswa.2022.116834

[22] Hoque S, Islam MR, Shafiullah M, Adnan S, Azam MS.*G*eneralized Normal Distribution Optimization Algorithm for Economic Dispatch with Renewable Resources Integration. Journal of Energy and Power Technology 2023 ; 5(3) : 030 ; doi :10.21926/jept.2303030.

[23] Mohamed, A. A., El-Sayed, A. H. (2024).*A*n application of the multi-objective generalized normal distribution optimizer for the design optimization of a brushless direct current motor. Heliyon, 10(3).

[24] Rai, R., Dhal, K.G. *R*ecent Developments in Equilibrium Optimizer Algorithm : Its Variants and Applications. Arch Computat Methods Eng 30, 3791–3844 (2023). https :// doi.org/10.1007/s11831-023-09923-y

[25] DINA ,A. E., MOHAMED ,M. S.,D ESSAM ,A. R (2021).Improved Equilibrium

Optimizer Algorithm for Features Selection :Methods and Analysis.

[26] Lan, P., Xia, K., Pan, Y., Fan, S. (2021). An Improved Equilibrium Optimizer Algorithm and Its Application in LSTM Neural Network.*A*n Symmetry, 13(9), 1706. https ://doi.org/
10.3390/sym13091706

[27] Benyoucef, H., Harchaou, K. (2021).*E*tude comparative entre l'EO, PSO et FA (Mémoire de Master, Université de Bordj Bou Arreridj, Faculté des mathématiques et de l'informatique). https ://dspace.univ-bba.dz/items/c49d2e4d-05e7-4947-b17a-9c2ba4b7b6d2

[28] Houssein, E.H., Hassan, M.H., Mahdy, M.A. et al. *D*evelopment and application of equilibrium optimizer for optimal power flow calculation of power system. Appl Intell 53, 7232–7253 (2023). https ://doi.org/10.1007/s10489-022-03796-7

[29] Boussard, A., Fessel, A., Oettmeier, C., Briard, L., Döbereiner, H.-G., Dussutour, A. (2021). *A*daptive behaviour and learning in slime moulds : The role of oscillations. Philosophical Transactions of the Royal Society B : Biological Sciences, 376(1820), 20190757. https ://doi.org/10.1098/rstb.2019.0757

[30] Reid CR.*T*houghts from the forest floor : a review of cognition in the slime mould Physarum polycephalum. Anim Cogn. 2023 Nov ;26(6) :1783-1797. doi : 10.1007/s10071-023-01782-1. Epub 2023 May 11. PMID : 37166523 ; PMCID : PMC10770251.

[31] Awad, A., Pang, W., Lusseau, D., Coghill, G. M. (2023). *A* survey on physarum polycephalum intelligent foraging behaviour and bio-inspired applications. Artificial Intelligence Review, 56(1), 1–26. https ://doi.org/10.1007/s10462-021-10112

[32] Li, S., Chen, H., Wang, M., Heidari, A. A., Mirjalili, S. (2020). *S*lime mould algorithm : A new method for stochastic optimization. Future Generation Computer Systems, 111, 300–323. https ://doi.org/10.1016/j.future.2020.03.055

[33] Sun, Q., Wang, C., Chen, Y. et al.*A*n efficient weighted slime mould algorithm for engineering optimization. J Big Data 11, 141 (2024). https ://doi.org/10.1186/s40537-024-01000-w