OPTIMIZATION OF PROPORTIONAL INTEGRAL DERIVATIVE PARAMETERS BASED ON IMPROVED PARTICLE SWARM OPTIMIZATION ALGORITHM FOR NON-LINEAR SYSTEMS

Patrick Ochieng Mboya Ogutu

A Thesis Submitted in Partial Fulfillment of the Requirements for the Degree of Master of Technology in Electrical and Electronics Engineering of Murang'a University of Technology

July, 2024

DECLARATION

Student Declaration

I hereby, declare that this Thesis is my original work and to the best of my knowledge has not been presented for a degree award in this or any other university. I understand what plagiarism is and am aware of the University's policy in this regard understand that any false claim in respect of this work shall result in disciplinary action as per the University Anti-Plagiarism Policy.

_____ _____

Patrick Ochieng Mboya Ogutu Date

EN400/5202/2018

Supervisors' Approval

The undersigned certify that they have read and hereby recommend for acceptance of the Murang'a University of Technology a thesis entitled "**Optimization of Proportional Integral Derivative Parameters Based on Improved Particle Swarm Optimization Algorithm for Non-Linear Systems**".

_____ _____

Dr.Nicholas Oyie, PhD Date

Department of Electrical and Electronics Engineering

Murang'a University of Technology

_____ _____

Dr.Winston Ojenge, PhD Date

Department of Electrical and Electronic Engineering

Technical University of Kenya

DEDICATION

I dedicate this thesis to my wife Lilian Auma Pondo, our son Pollen Seth Ogutu, and my parents Mr. Dickson Hezekiah Mboya Ogutu, Mrs. Mary Achola Mboya Ogutu and Kenya Universities Performing Arts Association -KUPAA who have shown me their unequivocal support for the entirety of this research period. It is through your backing, devotions and prayers that this has been possible will forever remain indebted.

ACKNOWLEDGEMENT

Special thanks and recognition go to the entire management of the Technical University of Kenya, for giving me time out of the university working schedule to conduct the research. I would also like to express my much appreciation to the internal and external supervisors, Dr. Nicholas Oyie and Dr. Winston Ojenge, for providing technical advice on areas where I thought I could not move ahead. Their contribution made a difference in the whole thesis, and without them, it could not have been possible. Finally my appreciation to all the staff at the Murang'a University of Science and Technology for their immense support.

ABSTRACT

The control of linear and nonlinear systems is of paramount importance when it comes to improvement of the plant performance. Many industries employ different modes of control when it comes to the proportional integral and derivative parameter tuning. Tuning control systems, whether for linear or nonlinear systems, can be a complex and challenging task. The challenges associated with tuning Linear Systems are Model Accuracy, Controller Selection, Parameter Tuning, System Variability while Challenges in Tuning Nonlinear Systems are Model Complexity, Multiple Equilibrium while for Nonlinear Control Strategies the challenges are Traditional linear control strategies may not be directly applicable to nonlinear systems, Input saturation, Uncertainty and Disturbances: Computational Complexity, Experimental Tuning: Safety Concerns: Overall, tuning control systems, whether for linear or nonlinear systems requires advanced tools and techniques such as particle swarm optimization. The effect on changing inertia weight so as to reduce the error was examined. The purpose of using Improved Particle Swarm Optimization is to understand its impact while tuning the Proportional Integral and Derivative parameters. The research used a standard nonlinear system depicting the real life situation, then using improved particle swarm optimization algorithm to analyze and compare the improved behavior on the MATLAB/Simulink tool box as applied to the proportional integral and derivative parameters. Finally it was logically realized that the improved particle swarm optimization algorithm system response was much better in comparison with the standard particle swarm optimization tuned system. It was noted that as the iteration was changed from 10, 50 all the way to 100, there was a significant reduction in integral time absolute error from 0.054806 to a minimum of 0.01900, which was far better than the standard particle swarm optimization algorithm. Standard particle swarm optimization reduced the error from 0.065143 to 0.020476.It is noted that the error is reduced from 5.4% to 1.9% which is significant achievement by the improved particle swarm optimization. The simulated system addressed the problem faced in dealing with the challenges in the conventional tuning of the proportional, integral and derivative parameters. The improved particle swarm optimization was proved to perform better compared to the standard particle swarm optimization and therefore is recommended for application in the industry.

TABLE OF CONTENTS

DECLARATION ii
DEDICATION iii
ACKNOWLEDGEMENT iv
ABSTRACT v
LIST OF TABLES ix
LIST OF FIGURES x
ACRONYMS AND ABBREVIATIONS xi
CHAPTER ONE: INTRODUCTION 1
1.1 Background of the Study 1
1.2 Statement of the Problem 4
1.3 Objectives of the Study 4
1.3.1 Main Objective 4
1.3.2 Specific Objectives 4
1.4 Research Questions 5
1.5 Significance of Study 5
1.6 Scope of the Study 6
1.7 Delimitation of the Study 6
1.8 Contributions of the Study 7
1.9 Organization of the Thesis 7
CHAPTER TWO: LITERATURE REVIEW 9
2.1 Introduction 9
2.2 Plant Controllability 9
2.2.1 Key points about controllability: 9
2.2.2 Non Linear systems 11
2.3 PID Tuning 12
2.3.1 PID mathematical presentation 13
2.3.2 History of tuning 14
2.3.3 Tuning Approaches 15
2.4 PSO PID system 19
2.4.1 PSO Background 19
2.4.2 PSO Algorithm 20

2.4.3 Advantages of PSO 21
2.4.4 Designing code for PSO 21
2.4.5 Procedure for optimizing PSO 22
2.4.6 PSO implementation procedure 23
2.4.7 Tuning of PID using Particle Swarm Optimization 24
2.4.8 Performance index 27
2.4.9 Evaluation of the Objective Function 28
2.4.10 Improved PSO 30
2.4.11 Empirical analysis of PSO characteristics 31
2.4.12 PSO Design analysis 32
2.4.13 Applications of PSO 36
2.5 PID Tuning works 36
2.6 Summary 42
CHAPTER THREE: METHODOLOGY 43
3.1 Introduction 43
3.2 Research Design 43
3.2.1 Data Collection 43
3.2.2 Study Hypothesis 43
3.3 Controllability status 44
3.4 Selection of Simulation software. 46
3.5 The plant system 47
3.5.1. MATLAB Simulink operation procedure 47
3.5.2. Plant design work 48
3.5.3. Study Algorithm 49
3.6 Ethical issues 53
3.7 Summary 53
CHAPTER FOUR: RESULTS AND DISCUSION 54
4.1 Introduction 54
4.2 Controllability 54
4.3 PID, SPSO and IPSO Simulation results 54
4.3.1 Simulation at 10 iterations 54
4.3.2 Simulation at 50 iterations 57
4.3.3 Comparison of performance indices at 10, 50 & 100 Iterations 59
4.4 Discussions 63

4.4.1 Effect of modified inertia weight 63
4.4.2 Effect of Iterations change on performance.....63
4.4.3 Effect of large number of particles.....63
4.5 Summary 64
CHAPTER FIVE: CONCLUSION, RECOMMENDATIONS AND FUTURE WORK.....65
5.1 Conclusion.....65
5.2 Recommendations 66
5.3 Future Works.....66
REFERENCES.....68
APPENDICES 74

LIST OF TABLES

Table 4.1: Comparison of performance indices at 10 Iterations at 30 Particles…..55
Table 4.2: Comparison of performance indices at 50 Iterations and 30 Particles...57
Table 4.3: Comparison of performance indices at 100 Iterations and 30 Particles.59
Table 4.4: Comparison of performance indices at 10, 50 & 100 Iterations…….61
Table 4.5: Comparison of performance indices at 15 and 30 particles……62

LIST OF FIGURES

Figure 2.1 Feedback block diagram……..11
Figure 2.2 PID parameters block diagram……13
Figure 2.3 Trial and Error tuning method…….16
Figure 2.4 Cohen coon tuning method……..16
Figure 2.5 Software tuning technique……...17
Figure 2.6 Astrom-Hugglund technique……...17
Figure 2.7. PSO flow chart……...26
Figure 2.8 PSO/PID block diagram……..33
Figure 2.9 Particle swarm optimization design…….35
Figure 3.1 Study Flow chart……..44
Figure 3.2 ITAE, ISE and IAE circuit blocks……48
Figure 3.3 PID/ plant block diagram……48
Figure 3.4 Combined block diagram……..49
Figure 3.5 PSO swarm movement……..50
Figure 3.6 Velocity parameters……..50
Figure 3.7 Effect of acceleration coefficient on search ability……...51
Figure 3.8 Effect of inertia weight on search ability……..52
Figure 4.1 System response verses time for 10 iterations……..56
Figure 4.2 Convergence behavior for 10 iterations……57
Figure 4.3 System response verses time for 50 iterations……..58
Figure 4.4 Convergence behavior for 50 iterations……59
Figure 4.5 System response verses time for 100 iterations……60
Figure 4.6 Convergence behavior for 100 iterations……..61

ACRONYMS AND ABBREVIATIONS

ACO	Anti Colony Optimization
AD	Analogue to Digital Interface
AFC	Adaptive Fuzzy Control
AGC	Automatic Generation Control
ANF	Adaptive Neural Fuzzy
ANFIS	Adaptive Neuro-Fuzzy Inference System
ANN	Artificial Neural Network
ATM	Adaptive Tuning Method
BB-BC	Big Bang Big Crunch
BFT	Bacteria Forage Technique
BPSO	Binary Particle Swarm Optimization
CLCS	Closed Loop Control System
DCS	Distributed Control System
DE	Differential Evolution
FLC	Fuzzy Logic Controller
FOPDT	First Order Plus-Dead Time
FO	Fractional Order
FPID	Fuzzy Proportional Integral Derivative
FEIW	Flexible Experimental Control weights
GA	Genetic Algorithm
GWO	Grey Wolf Optimization
HVAC	Heating Ventilation Air Conditioning
IPSO	Improved Particle Swarm Optimization
ITAE	Integral Time Absolute Error
LCOA	Lozi Chaotic Optimization Algorithm
LDIW	Linear Decreasing Inertia weight
LOS	Line Of Sight
MF	Membership Function
MIMO	Multi-Input Multi-Output
MOPSO	Multi-Objective Particle Swarm Optimization
MSPSO	Multi Start Particle Swarm Optimization
MPC	Model Predictive Controller

MPPSO	Multi-Phase Particle Swarm Optimization
MSE	Mean Square Error
OFPID	Optimum Fuzzy Proportional Integral Derivative
OS	Over Shoot
PEMFC	Polymer Electrolyte Membrane Fuel Control
PBSO	PID Control-Based Strategy Particle Swarm Optimization
PID	Proportional Integral Derivative
PSO	Particle Swarm Optimization
PSOSCALF	Particle Swam Optimization Sine Cosine Algorithm and Levy Flight
PPSO	Perturbed Particle Swarm Optimization
PV	Present Value
SP	Set Point
SPSO	Standard Particle Swarm Optimization
SSE	Steady State Error
ST	Settling Time
STA	State Transition Algorithm
TF	Transfer Function
TITO	Two Input Two Output
TLBO	Teaching Learning Based Optimization
VEPSO	Vector Evaluated Particle Swarm Optimization
ZN	Ziegler Nichols

CHAPTER ONE

INTRODUCTION

1.1 Background of the Study

Real-world systems are inherently nonlinear in nature at least when a wide operating range is considered. Primary reasons behind the growing interest in nonlinear control include the improvement of linear control systems, analysis of hard non-linearity, and the need to deal with plant model uncertainties and design complications. A nonlinear controller through robustness or adaptability can handle the consequences due to model uncertainties. As the study moves from linear to nonlinear systems a more difficult situation is met [1]. The superposition principle no longer holds, and analysis tools necessarily involve more advanced mathematics. Most importantly since the superposition principle does not hold, we cannot assume that an analysis of the behavior of the system either analytically or via simulation may be scaled up or down to tell us about the behavior at large or small scales.

The first step when analyzing a nonlinear system is usually to make it linear about some nominal operating point and analyze the resulting linear model. However, it is clear that linearization alone will not be sufficient. We therefore must develop tools for the analysis of nonlinear systems. There are two basic limitation of linearization. First, since linearization is an approximation in the proximity of an operating point, it can only predict the local behavior of the nonlinear system in the vicinity of that point. Secondly, the dynamics of a nonlinear system are much richer than the dynamics of a linear system. There are essentially nonlinear phenomena that can take place only in the presence of non-linearity; hence they cannot be described or predicted by linear models. The following are examples of nonlinear phenomena [2]: -Finite escape time: The state of an unstable linear system can go to infinity as time approaches infinity. A nonlinear system's state, however, can go to infinity in finite time, Multiple isolated equilibrium points: A linear system can have only one equilibrium point, and thus only one steady-state operating point that attracts or repels the state of the system irrespective of the initial state. A nonlinear system can have more than one equilibrium point, Limit cycles: A linear system can have a stable oscillation if it has a pair of

eigenvalues on the imaginary axis. The amplitude of the oscillation will then depend on the initial conditions. A nonlinear system can exhibit an oscillation of fixed amplitude and frequency which appears independently of the initial conditions, Chaos: A nonlinear system can have a more complicated steady-state behavior that is not equilibrium or periodic oscillation. Some of these chaotic motions exhibit randomness, despite the deterministic nature of the system, Multiple Modes of behavior: A nonlinear system may exhibit very different forms of behavior depending on external parameter values, or may jump from one form of behavior to another autonomously.

These behaviors cannot be observed in linear systems, where the complete system behavior is characterized by the eigenvalues of the system matrix A. To achieve the stated control aspect described in background the use of Proportional Integral and Derivative (PID) control algorithm is introduced to the plant or process. PID is used due to its robustness, simplicity and very good and reliable control effect. Due to the stated factors it has become the most widely used control algorithm in many industries. However as far as PID is concerned, there is need to achieve better performance and for that matter the PID parameters need to be technically adjusted as per the real working environment and condition. This brought up the need for tuning either manually or automatically. [3]

The PID parameters is the core of this research work as applied in engineering design and application. Among the theoretical tuning there is root locus and frequency response method attenuation curve, critical proportional band among others. As far as performance and application is concerned some tuning methods require long term experience and far too many experimental analysis in order to succeed in the implementation, [4]to overcome the above teething problem researchers embarked on research on the possible use of automatic tuning of the PID Algorithm controller parameters. Some of the soft tuning methods which have been undertaken are back propagation neural network and the fuzzy logic self-tuning PID controller .Study revealed that there is improvement in the over shoot and response time compared to the conventional PID.

The study came up with a standard plant model as the objective system to be controlled using improved Particle swarm optimization (IPSO) Algorithm technique. The system step response is IPSO Algorithm. While comparing the manual tuning methods of conventional PID controllers, it is noted and concluded that the former is more reliable and efficient.

The manual PID controllers' tuning methodology suffers from non-linearity and also has very large settling time and overshoot [5].The core area of the research is to analyze the overall performance of an IPSO algorithm as utilized in the process of tuning the PID controller parameters with the sole objective of optimizing plant behavior [6].

The use of Ziegler Nichols process control technique tuning mechanism was found to be too cumbersome and unpredictable hence a more efficient technique was required to ensure good response and for that matter improved PSO algorithm was used. PSO came as a result of the study of bird movement in nature, the process or plant optimization problem is tackled by the simple behavior of individual and their interaction within the group set up.

The PSO algorithm is basically used in various areas as function optimization, image processing and other fields, this is due to the fact that it is simple in operation and have very fast convergence time. In order to prove the effectiveness of the proposed method, a standard plant or process is used as the control object, then simulation models used on MATLAB Simulink platform to compare the performance effect with the conventional PID.

In this research, the problem statement was stated, the general and specific objectives were specified well, the research question guiding the research which forms the basis of the study was well adhered to, the significance and scope of the study were tackled in detail. The delimitation of the study and theory conceptual frame work clearly illustrated.

1.2 Statement of the Problem

In many industrial control systems, the performance of Proportional-Integral-Derivative (PID) controllers plays a critical role in achieving precise and stable control of various processes. The challenge often lies in selecting appropriate PID controller parameters (Kp, Ki, and Kd) to optimize the system's performance. Conventional tuning methods, such as trial-and-error or heuristic approaches, can be time-consuming and may not always yield optimal results.

This research aims to address the problem of PID controller tuning by employing the Improved Particle Swarm Optimization (IPSO) algorithm. The goal is to design and implement an efficient and automated PID tuning framework that leverages IPSO's capabilities to find optimal or near-optimal PID parameters for a given control system.

Particle Swarm Optimization (PSO) is a popular optimization algorithm that has been successfully applied to various control problems. However, the standard particle swarm optimization (SPSO) algorithm has limitations in handling non-linear systems with complex dynamics. Therefore, there is a need to develop IPSO algorithm that can effectively optimize PID parameters for non-linear systems.

The IPSO method aims to improve control performance, reduce human effort, and increase the robustness of the control system. The results of this study will contribute to the development of efficient and effective methods for PID tuning in non-linear systems.

1.3 Objectives of the Study

1.3.1 Main Objective

The objective of this research is to develop a novel method for optimizing PID parameters in non-linear systems using an IPSO algorithm.

1.3.2 Specific objectives

The specific objectives of this research are:-

i. To determine Controllability status of the non-linear plant

ii. To design a controller system based on PID control in the MATLAB/Simulink platform
iii. To optimize the PID parameters using the standard PSO and improved PSO algorithms
iv. To validate the findings of the improved PSO algorithm by comparing the simulation results with the PID and the standard PSO algorithm results

1.4 Research Questions

The research questions tackled were as follows:-

i. What makes a nonlinear plant controllable?
ii. How can the nonlinear plant be designed based on PID control in the MATLAB/Simulink platform?
iii. When Optimizing the PID parameters using the standard PSO and improved PSO algorithms, what parameters are of importance and actual values?
iv. How do you compare the improved PSO algorithm, the PID and the standard PSO in terms of performance?

1.5 Significance of Study

Significance of studying the optimization of PID parameters based on the improved Particle Swarm Optimization (IPSO) algorithm for non-linear systems are as follows: Improved control performance: The optimization of PID parameters using the PSO algorithm can lead to improved control performance of non-linear systems. By fine-tuning the PID parameters, the system can achieve better stability, faster response, and reduced overshoot, which can result in more accurate control of the system, Cost savings: Optimization of PID parameters can lead to significant cost savings in industrial applications. By achieving better control performance, the system can reduce the waste of resources, improve energy efficiency, and increase production output, which can result in lower operational costs, increased reliability: Optimizing the PID parameters using the PSO algorithm can increase the reliability and robustness of the control system. By fine-tuning the parameters, the system can adapt to changes in the system dynamics, environmental conditions, and other disturbances, which can result in more consistent and reliable control, Easy implementation: The PSO

algorithm is easy to implement and can be applied to various non-linear systems. By using a simple and efficient optimization method, engineers and researchers can quickly optimize the PID parameters for different applications, which can save time and effort. Advancement of research: Studying the optimization of PID parameters using the PSO algorithm can contribute to the advancement of research in control theory and optimization. By exploring new optimization techniques, researchers can develop more efficient and effective methods for controlling non-linear systems, which can lead to new discoveries and innovations in various fields.

Finally studying the optimization of PID parameters based on an IPSO algorithm for non-linear systems can lead to significant improvements in control performance, reduced human effort, increased robustness, and wider adoption of the PID controller. The application of the improved PSO Algorithm in tuning the PID controller is highly relevant and significant to the study of control systems in Kenya. Conducting this research will significantly contribute to realizing the most efficient and cost-effective method of tuning the PID control parameters using the auto-tuning PSO method. One of the major advantages of tuning the PID parameters using the proposed method is its specificity, precision, and accuracy in operation. The selected approach of using the IPSO Algorithm to tune the PID parameters makes this application in process control to be of great importance in ensuring optimal performance.

1.6 Scope of the Study

The scope of the research on the optimization of PID parameters based on the IPSO algorithm for non-linear systems involved:-.Development of IPSO algorithms: The research focused on developing new and IPSO algorithms that are specifically designed for optimizing the PID parameters of non-linear systems. The algorithms was modified to account for different types of non-linearity's and system dynamics, leading to better performance and faster convergence. Implementation of the PSO algorithm: The research involved the implementation of the PSO algorithm on real-world non-linear systems in this case chicken .Sensitivity analysis: The research involved conducting sensitivity analysis to study the effect of different inertia weight on the PID controller's parameter tuning, this will helped in identifying critical system

parameters that can significantly impact the control performance and can aid in selecting appropriate PID parameters for different operating conditions.

1.7 Delimitation of the Study

The research concentrated on the performance analysis of a plant or process using both PID, SPSO and IPSO PID parameter tuning. The following factors will not be used in the research: Study of linear systems due to their simplicity and low application, Meta rustics Algorithm such as Genetic Algorithm, Anti Colon Optimization and Fire Fly techniques, and third order functions and above

1.8 Contributions of the Study

The study made the following contributions:-Improved Control Performance: By optimizing the PID parameters using an IPSO algorithm, the control performance was significantly improved, leading to better stability, faster response, and reduced overshoot and settling time, Reduction in Human Effort-by using the IPSO, the tuning process is automated, saving time and effort while improving the overall control performance, Robustness: by using an improved PSO algorithm, the PID controller was tuned to a more robust to disturbances leading to better control performance under different operating conditions, IPSO algorithm was used to optimize the PID parameters for a wide range of non-linear systems making it a versatile and general approach for PID tuning.

Finally studying the optimization of PID parameters based on an IPSO algorithm for non-linear systems led to significant improvements in control performance, reduced human effort, increased robustness, and wider adoption of the PID controller. Further the research will contribute to advancing the field of control theory and optimization and can have practical applications in various industries where non-linear systems are prevalent

1.9 Organization of the Thesis

The general organization of a thesis on the optimization of PID parameters based on an improved PSO algorithm for non-linear systems includes the following sections:

Introduction: This section provide an overview of the research problem, the motivation for the study, and the objectives and scope of the thesis. It provide a brief overview of the PID controller, the PSO algorithm, and their applications in control systems.

Literature Review: This section present a comprehensive review of the existing literature related to the optimization of PID parameters using PSO algorithm for non-linear systems. It highlight the gaps in the literature and identify the research questions that will be addressed in the thesis.

Methodology: This section describes the methodology used in the study, including the mathematical models used to describe the non-linear system, the PID controller, and the PSO algorithm. It also provide details on the experimental setup, simulation tools, and performance metrics used to evaluate the performance of the optimized PID controller.

Results and Discussion: This section present the results of the study, including the performance of the optimized PID controller under different operating conditions and disturbance levels. The results should be analyzed and interpreted to provide insights into the effectiveness of the proposed method in improving the control performance of the non-linear system.

Conclusion: This section summarize the key findings of the study, including the contributions to the field, the limitations of the study, and the future research directions. It also provides recommendations for the application of the proposed method in practical control systems.

CHAPTER TWO
LITERATURE REVIEW

2.1 Introduction

In this section, the literature review on different tuning techniques is discussed with emphasis on the theory behind the particle swarm optimization, plant controllability principles, PID theory and the effect of improved particle swarm optimization in solving system response for nonlinear system.

The section also focuses on related literature review from various researchers conducted on different tuning techniques. The chapter collects relevant information from the past literature in order to narrow the research gap left by the past researchers. The current trends in PID tuning are also discussed and gaps identified. Most processes or plant in either domestic or industrial set up require some control element and if the control component cannot be effected stably, it will end up causing serious damage to the electronics circuits and the entire equipment. The same can result into injury and fatality hence it is very significant to ensure the performance of the process up to the expected level. To achieve the desired performance a detailed explanation is done on the Particle swarm optimization (PSO) Proportional Integral Derivative (PID) system and various tuning approaches.

2.2 Plant Controllability

Controllability is a fundamental concept in control theory and engineering, particularly in the context of dynamic systems. It refers to the ability to control a system's state (or trajectory) from an initial state to a desired final state using control inputs or actuators. A system is said to be controllable if it can be driven from any initial state to any desired state within a finite amount of time, given appropriate control inputs.

2.2.1 Key points about controllability:

The following are Key points about controllability (a) State-Space Representation**:** Controllability analysis is often performed in the state-space representation of a dynamic system. Where the system's behavior is described by a set of first-order differential equations, typically represented as: $\dot{x}(t) = Ax(t) + Bu(t)$

Where:

- $x(t)$ represents the state vector of the system at time t.
- A is the system's state matrix, describing how the system evolves autonomously.
- B is the control input matrix, describing how the control inputs affect the system.

(b) Controllable vs. Uncontrollable Systems**:** A system is considered controllable if, for any initial state x_o and any final state x_f , there exists a control input u (t) over a finite time horizon that can drive the system from x_o to x_f , Conversely, if there are some initial and final states for which there is no control input that can connect them within a finite time, the system is considered uncontrollable,(c) Controllability Matrix**:** Controllability analysis often involves calculating the controllability matrix, denoted as C. [7] The controllability matrix is a function of the system's state and input matrices A and B. A system is controllable if and only if the controllability matrix has full rank, meaning its rank is equal to the dimension of the state vector x.,(d) Applications**:** Controllability analysis is crucial in control system design, especially for ensuring that a system can be effectively controlled to meet desired performance specifications. It is used in the design of controllers, such as state feedback controllers and optimal control strategies. (e) Physical Systems: Controllability is applicable to a wide range of physical systems, including mechanical systems (e.g., robotic arms), electrical circuits, chemical processes, and more. Understanding whether a system is controllable is essential for designing control strategies for these systems. (f) Observability: Controllability is related to observability, another important concept in control theory. Observability refers to the ability to estimate the internal states of a system based on the system's outputs. Controllability and observability are often considered together when designing control systems. [8]

In summary, controllability in this study is a key property of dynamic systems that shows whether a system can be effectively controlled from any initial state to any desired state. It plays a critical role in chosen plant design, helping the findings so as to ensure that control strategies achieve desired performance objectives.

2.2.2 Non Linear systems

Non-linear systems are systems whose mathematical equations that do not follow a straight line. These equations are complex and can be difficult to solve. Non-linear equations examples can be found in various fields such as physics, engineering, economics, and biology. In this section we will explore some non-linear equations in relation to different applications.

2.2.2.1 Examples of nonlinear system

Nonlinear System of Equations: A nonlinear system of equations is a set of equations that do not follow a straight line. These equations can have multiple solutions or no solution at all. Nonlinear systems of equations can be solved using numerical methods such as Newton's method or the bisection method, Differential Equation: A differential equation is a non-linear equation that involves derivatives. These equations are used to model physical phenomena such as motion, heat transfer, and fluid flow. Differential equations can be solved using numerical methods or analytical methods, Nonlinear Optimization: Nonlinear optimization is a mathematical technique used to find the maximum or minimum value of a function that does not follow a straight line. Nonlinear optimization can be used in various fields such as economics, engineering, and physics, Chaos Theory is a branch of mathematics that studies the behavior of nonlinear systems. Chaos theory is used to understand complex systems such as weather patterns, stock markets, and biological systems. [9].For this study the nonlinear transfer function is derived from [10] and for the plant application is from [11] .

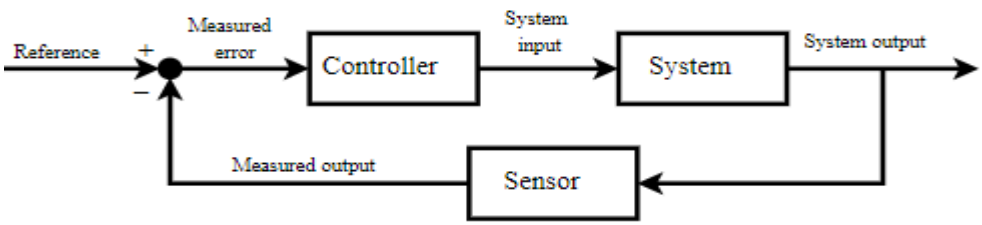


Figure 2.1 Feedback block diagram

2.3 PID Tuning

PID parameter tuning is an active research area in process control applications. Proper tuning of the three parameters (proportional, integral, and derivative) is essential for effective performance improvement as applied to industrial processes or biomedical instrumentation systems.

When selecting a PID system the following assumptions should be stated: the plant must be controllable and observable, and there should be a wide range of knowledge-including basic engineering logic, a data set for input/output, and the controller analytical model (which can easily be implemented). Additionally, a solution should be available for the task at hand, however, the control engineering designer should work towards an appropriate solution not necessarily an optimized one. The controller used in the design should be able to perform as per specifications within an acceptable range of precision.

It should be well noted that stability and optimal are ongoing problems in PID controller. The purpose of parameter tuning is to achieve desired performance and meet set design requirements as well as to set the PID values to get the best possible control for a particular process [5]. The following terminologies will be used in achieving the research objectives: Proportional Band – the band expressed in % of scale within which the controller proportional action takes place. The wider the PB, the greater the area around the set point in which the proportional action takes place, sometimes referred to as the gain 1/PB, Integral – reset adjusts the PB with respect to the set point to compensate for offset droop from the set point. It adjusts the controller temperature to the set point after the system stabilizes, Derivative – rate senses the rate of rise or fall of system temperature and automatically adjusts the PB to minimize the overshoot or undershoot

In order to obtain the required output the PID controller must be properly tuned. This means that the optimal gains of the proportional, integral, and derivative terms are adjusted to achieve the ideal system response. The device is tuned to prevent disturbances which means that when the set point is changed the controller's output adjusts to the new set point. When the PID controller is appropriately tuned the system output will automatically adjust to a new set point with minimal oscillations and damping.

2.3.1 PID mathematical presentation

A Proportional-Integral-Derivative (PID) controller is a widely used feedback control algorithm that is used to regulate a system by adjusting its output based on the error between a desired set point and the actual output. The amplitude and period of the output waveform are then calculated using the mathematical equation for a controller in Eq 2.1 [12].

$$u(t) = K_p e(t) + K_i \int edt + K_d \frac{de(t)}{dt} \quad (2.1)$$

This equation is written in the continuous/analog domain whereas many of the controllers implemented these days are in digital form. To compute the output at time t it takes the error value multiplying it by a coefficient (Kp), and that is it. When applying it to Integral control action the proportion is very straightforward though prone to problematic situations. Finally, the Derivative control action is the least frequently used component. Most systems can make do with just P and I with the derivative bringing only slight improvements. In most industrial situations the derivative component is the only one that would never reach the set point on its own. The proportional and integral control actions respond to an error with a minimum output at the same time the error doesn't change the derivative which is set at zero.

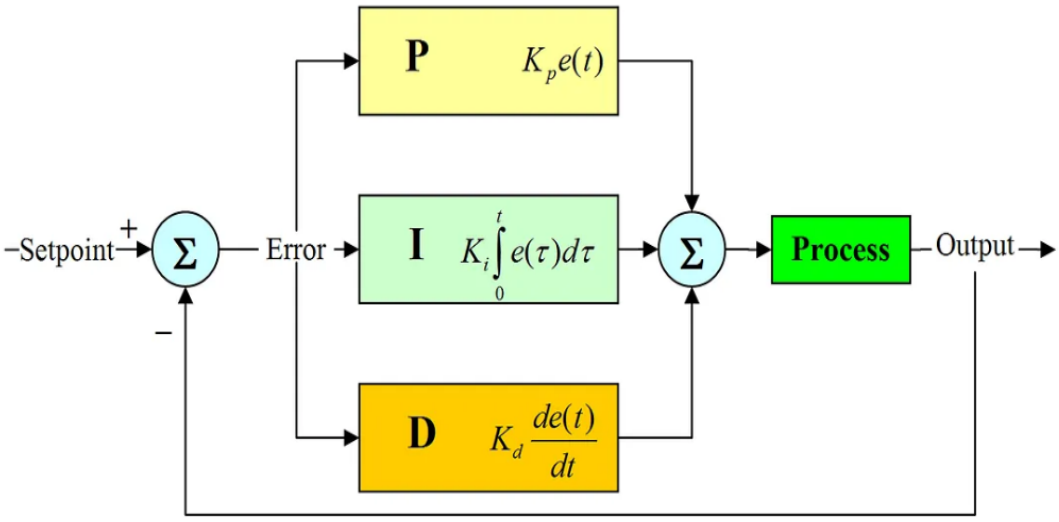


Figure 2.2 PID parameters block diagram

The block diagram of a Proportional-Integral-Derivative (PID) controller in Fig 2.2 describes the various parameters as illustrated with the following constants consisting of several components that work together to regulate a system. The block diagram includes the following components:

Error signal: The error signal is the difference between the set point and the actual output of the system. The error signal is fed into the PID controller, which calculates the control output based on the error, Proportional (P) block: The P block multiplies the error signal by a proportional gain, Kp, to produce a proportional term output. The proportional term output is directly proportional to the error signal, and it determines the immediate response of the system to changes in the error. A high proportional gain will produce a larger control output for a given error, resulting in a more responsive system, integral (I) block: Integrates the error signal over time to produce an integral term output. The integral term output accumulates the past errors and compensates for any steady-state error that may exist in the system. The integral gain, Ki, determines the rate at which past errors are accumulated. A high integral gain will produce a larger integral term output for a given error, resulting in a more aggressive response to steady-state errors, Derivative (D) block: The D block calculates the rate of change of the error signal over time to produce a derivative term output. The derivative term output dampens the system response to rapid changes in the error signal, reducing overshoot and oscillations. The derivative gain, Kd, determines the sensitivity of the derivative term output to changes in the error rate. A high derivative gain will produce a larger derivative term output for a given change in the error rate, resulting in a more sensitive system response, Summing junction: The summing junction combines the proportional, integral, and derivative term outputs to produce the control output, u (t). The control output is the sum of the three term outputs and it represents the corrective action that needs to be applied to the system to reduce the error, the tuning of the PID gains K_p, K_i, and K_d, is crucial to achieve optimal system performance. The tuning process involves adjusting the gains to minimize the error and achieve the desired system response characteristics, such as settling time, overshoot and stability.

2.3.2 History of tuning

The history of tuning began with Ziegler-Nichols in 1942, followed by Cohen-Coon in 1953, Astrom and Hagglow in 1984, De Paor and Umally in 1989, Zhuang and

Atherton in 1993, Venkatashanker and Chidambaram in 1994, Pouline and Pomerleav in 1996 and finally Huang and Chan in 1996. The main focus of the above tuning methods was the dynamical behavior under closed-loop conditions. Even though novel, effective theory and design methodologies are continually being developed in the digital control field the PID automatic tuning remains the most preferred technique used to solve problems both in the domestic and industrial sectors [13]. Linear control systems can be easily tuned using conventional tuning techniques such as Ziegler-Nichols and Cohen-Coon tuning formulae, but the results often show unsatisfactory performance and involve more trial and error. To overcome these tuning obstacles for meta-heuristic approaches is embraced

2.3.3 Tuning Approaches

Proportional-Integral-Derivative (PID) Tuning is very common in any engineering application and the methods used has strength and weaknesses but generally the Strengths are :Simplicity: PID controllers are relatively simple to implement and understand, Versatility: They can be applied to a wide range of systems and are effective for many control tasks, Stability: Properly tuned PID controllers can provide stable control while weakness noted are :-Trial-and-Error: Tuning PID controllers can be time-consuming and often relies on trial-and-error methods, Limited for Complex Systems: PID controllers may struggle with highly nonlinear or complex systems, Sub-optimal for Some Systems: They may not provide optimal control for systems with significant time delays or varying dynamics. There are various tuning methodologies available to make a PID controller operate with proper tuning. A few of these methods are explained below: [14].

a. Trial & Error – This is a manual tuning approach that involves increasing the proportional value until the controller reaches the point of oscillating response without losing stability, with integral and derivative terms set to zero. Next, the integral value is adjusted to end the oscillation, followed by adjusting the derivative value to provide a quick response. as shown in Figure 3

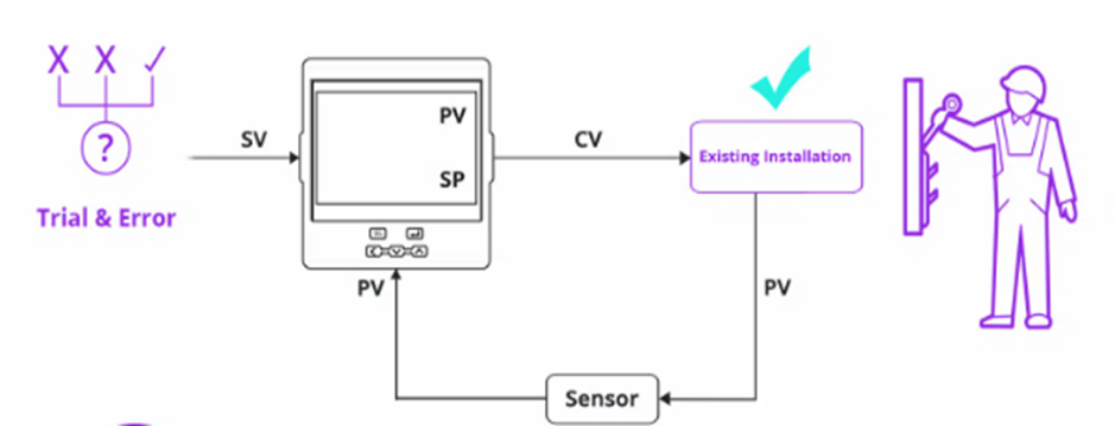


Figure 2.3 Trial and Error tuning method

b. Ziegler-Nichols – In this method, the integral and derivative terms are set to zero initially, and the proportional term value is gradually increased until the point of ultimate gain is reached (the point where the output loop begins to oscillate). Using the ultimate gain and oscillation period, the PID controller's gain can be determined as in Figure 2.3.

c. Process Reaction Curve – Also known as the Cohen-Coon tuning methodology, this approach involves generating a process reaction curve with respect to disturbance. The curve provides information on the controller gain, derivative, and integral times. This curve can be observed when the system operates in an open-loop step test, and the slope, dead, and rise time of the curve correspond to the proportional, integral, and derivative values as shown in Figure 2.4

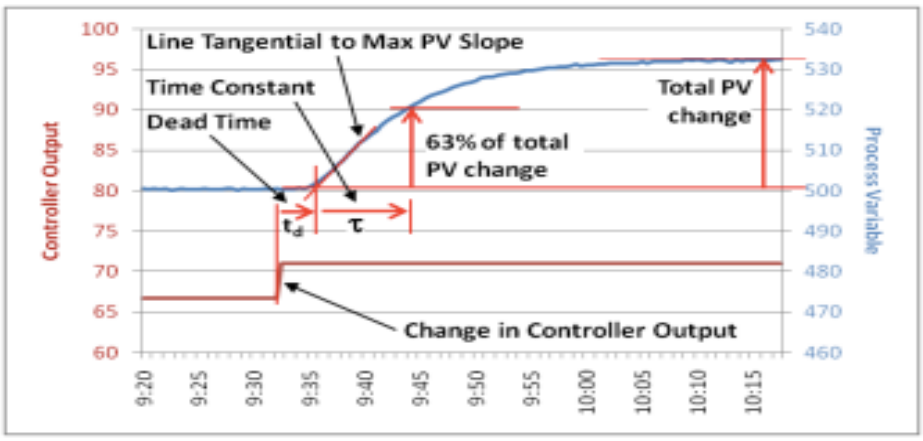


Figure 2.4 Cohen coon tuning method

d. Software – Various software packages are available for loop optimization and controller tuning. These software packages collect information and prepare corresponding mathematical models. PID Tuner provides a fast and widely applicable single-loop PID tuning method for the Simulink® PID Controller blocks. With this method, you can tune PID controller parameters to achieve a robust design with the desired response time as described in Figure 2.5.

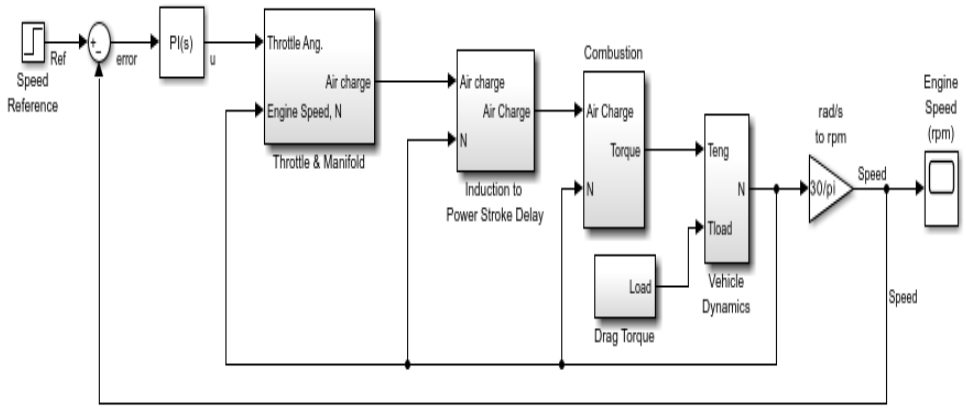


Figure 2.5 Software tuning technique

e. Relay – Also known as the Astrom-Hugglund technique of tuning the PID controller, [15] this method involves switching the output between two control variable values, which exceed the set point values. The output wave-form's amplitude and period are calculated to determine the oscillation period and ultimate gain. as described in Figure 2.6

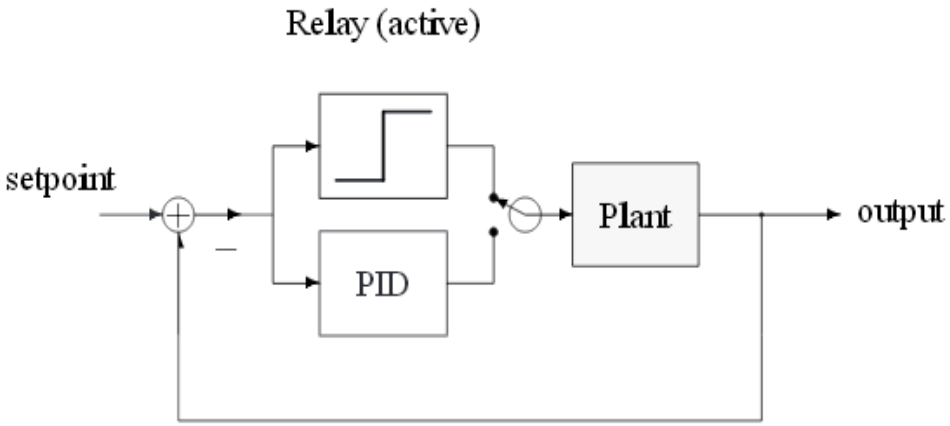


Figure 2.6 Astrom-Hugglund technique

2.3.4 Strength and weakness of tuning methods

Ziegler-Nichols Method (PID Tuning Method):has the following Strengths: Quick Estimation: Ziegler-Nichols provides a quick initial estimate of PID controller parameters, Applicable to Various Systems: It can be applied to various processes without detailed system knowledge while the weaknesses are :-Conservatism: The Ziegler-Nichols method can lead to conservative (overshoot) control settings, which may not be suitable for all systems, Lack of Optimal: It does not guarantee optimal performance or robustness

Model-Based Tuning (e.g., Pole Placement): has the following Strengths: Optimal: Model-based methods can achieve optimal control performance when the system model is accurate, Customization: Controllers can be tailored to specific performance criteria, Applicable to Complex Systems: Effective for complex systems with known models while it has the following weaknesses: Model Requirement: Requires an accurate system model, which may not always be available, Sensitivity: Sensitivity to model inaccuracies and parameter uncertainties ,Complexity: Model-based methods can be computationally complex and may require specialized software.

Frequency Response Analysis (e.g., Bode Analysis): Has the following Strengths: Frequency Domain Insight: Provides insights into system behavior at different frequencies, Robustness Analysis: Useful for robust controller design. While the weaknesses are as follows: Limited for Nonlinear Systems: Less effective for highly nonlinear systems and Labor-Intensive: Requires system measurements and frequency response experiments.

Optimization-Based Tuning (e.g., Gradient Descent, Genetic Algorithms):this is the mostly used currently and has several strengths :Optimal: Can find optimal or near-optimal controller parameters, Customization: Controllers can be tailored to meet specific performance criteria, Applicable to Complex Systems: Effective for complex and nonlinear systems while the major weaknesses are: Computationally Intensive: Optimization-based methods can be computationally expensive and time-consuming, Model Requirement: May require an accurate system model for optimization, sensitivity to Initialization: Results may depend on the initial guess or population in optimization algorithms. [16]

2.4 PSO PID system

The use of IPSO algorithm in tuning PID parameters in a domestic or industrial plant is suitable for any selected environment. For this study the plant or process will be considered standard, as has been previously studied in different research work. PSO is a computational optimization technique that can be used to tune the parameters of a PID control system. A PID controller is a feedback control system that is commonly used in industrial control systems to regulate the behavior of a dynamic system. It works by continuously measuring the output of the system, comparing it to a desired set point and adjusting the input to the system to minimize the difference between the set point and the actual output.

PSO is a population-based optimization method first proposed by Eberhard and colleagues. Some of the attractive features of PSO include the ease of implementation and the fact that no gradient information is required. It can be used to solve a wide array of different optimization problems like evolutionary algorithms the PSO technique conducts a search using a population of particles corresponding to individuals. Each particle represents a candidate solution to the problem at hand. In a PSO system, particles change their positions by flying around in a multidimensional search space until computational limitations are exceeded. This new tuning approach features many advantages such as simplicity, fast in response and coding require few lines. Moreover, this approach is advantageous over evolutionary and genetic algorithms in many ways: PSO memory. That is, every particle remembers its best solution (global best), Initial population of the PSO is maintained and so there is no need for applying operators to the population a process that is time wastage and memory storage-consuming, PSO is based on constructive cooperation between particles in contrast with the genetic algorithms which are based on the survival of the fittest. [17]

2.4.1 PSO Background

The nature-inspired algorithm can be based on either evolution or particle swarm optimization. Evolution can be either genetic algorithm or differential evolution while for swarm optimization we have particle swarm optimization (PSO) for birds or ant colony optimization (ACO). PSO works based on the collective behavior of animals

is given by Qii = Σiwi*f = Σiwif/ Σiwi, A swarm of n particles flies over a system's space to locate a global optimum. It is best designed to communicate directly or indirectly with one another using search direct gradient descent. During an iteration of PSO, each particle updates its position according to its previous exposure, as well as the experience of its neighbors.

A particle is composed of write-up vectors, modeling records, the current position of the particle in the search space, the vector is known as best (records the location of the best solution found so far by a particle), and the vector containing a gradient direction in which a particle will travel if not disturbed.
In PSO each particle moves towards the best position found by it, that is the pbest so that the best global (gbest) is obtained so far by any particle with a random weighted acceleration at each time step [18].
This is done by adding a v-vector to the x vector to get another x vector (xi= xi+ vi). Once a particle computes the new xi it then evaluates its new location. If x fitness is better than p-fitness then pbest =xi and p-fitness. Factors that swarm depends on are: Swarm size, Position of particle, Velocity of the particle, Maximum number of iterations

2.4.2 PSO Algorithm

The PSO algorithm is a stochastic approach based on the principles of natural selection and search algorithms. There is much evidence to support the theory behind this investigation such as animal movement and plant growth including ants foraging, birds flocking, fish schooling and bacterial chemistry.

The main goal of PID tuning is to control parameters that meet the closed-loop system performance specifications and improve them over a wide range of operating conditions. PID adjustments result in better transient response to set point changes, while poor adjustments result in a sluggish response to disturbances [19].Choosing conservative values of the control parameter can result in slow closed-loop response to set point changes.

PSO is a stochastic optimization algorithm based on a social simulation model whose development is inspired by the concept and rules of collision-free, synchronized moves that govern socially organized populations in nature such as bird flocks, fish

schools and animal herds. The traditional tuning method provides data for the tuned parameters which are then used with the PSO system [20].

2.4.3 Advantages of PSO

The following are major advantages related to PSO: The algorithm does not use the gradient information of objective function but uses its real value, Ease of implementation, Can be applied to solve nonlinear, multiple optimum, and high-dimensional problems. Solutions hardly depend on the initial state of the particles, Low a computational time, High global convergence speed, Does not need gradient information and therefore can minimize naturally defined cost functions without the use of complex mathematical operations, Does not require gradient information and basic mathematical operations, Uses very few parameters, making analysis easy, Easily applicable to nonlinear, non-convex, continuous, discrete, and integral variables, Review and discussion of the most established results in PSO algorithms are well tackled.

However, the method suffers from the difficulty in the evaluation of the first derivative for many functions in discontinuous spaces. It also suffers from partial optimism which degrades the regulation and the lack of diversity thus injecting random chaos. A swarm can be defined as an apparently disorganized collection of moving particles or a population that tends to cluster together while each particle seems to be moving in a random direction [18].

2.4.4 Designing code for PSO

As [21] referred in the above statements the assumption are:-

No of particles swarm = swarm (n)

Movement in space = inertia (w)

Response in group = correction factor (c_1/c_2)

Velocity and position update equation Eq 2.2, Eq2.3 and Eq 2.4 is given by:

$$Vij = wvij + c_1r_1(pbest - xij) + c_2r_2(gbest - xij) \quad (2.2)$$

$$Xij = vij + xij \quad (2.3)$$

$$W = wmax - (iter/max\,iter)\,(wmax - wmin) \quad (2.4)$$

Where

W_{max} = final weight

W_{min} = initial weight

Max_{iter} = maximum iteration number

Iter = current iteration number

r_1 /r_2 = random numbers (0-1)

V_{ij} = previous velocity

P = personal best

G = global best

2.4.5 Procedure for Optimizing PSO

The standard procedure for optimizing a plant process typically involves the following steps: Define the objective: Clearly define the goal of the optimization process, which could be to maximize production, minimize energy consumption, reduce waste, or improve product quality, among others., Identify the variables: Identify the key variables that affect the process and determine the optimal set points for these variables, Gather data: Collect data on the current performance of the process, including measurements of the variables and other relevant parameters, such as flow rates, temperatures, pressures, and compositions, Develop a model: Develop a mathematical model of the process using the collected data and other relevant information. The model should be able to predict the process behavior under different operating conditions, Simulate the process: Use the model to simulate the process behavior and optimize the set points of the variables to achieve the defined objective, Validate the model: Validate the model by comparing the simulation results with the actual performance of the process. Adjust the model as necessary to improve its accuracy, Implement the optimized process: Implement the optimized set points and monitor the process to ensure that it is running smoothly and meeting the defined objective, Continuously monitor and improve: Continuously monitor the process performance and make adjustments as necessary to maintain or improve the optimization.

[20] Particle swarm optimization is inspired by social behavior of birds flocking where it searches for optimum by updating generations. The Swarm refers to populating the moving particle as per the number of iterations, it further solve problems by having population called swarms of candidate solutions. Each particle movement is influenced by its local best known position guided by known position in the search

space, therefore when any particle exceeds the search space, the value will be re initialized so as to arrive at the best optimum value.

The controlled parameters are in a range in which the algorithm computes the optimal control variable. As per practical experimental work the particle swarm optimization technique Algorithm do not guarantee an optimal solution however it is Easy to implement and utilizes very few parameters. The Fitness function is used in meta-heuristic algorithm for optimization where conditions are best and most favorable. Most systems use objective function so as to obtain the maximum or minimum value of the controlled conditions. Each particle searches optimum value by updating the generation changes effected by the process. In each iteration every particle is updated by following the two best values, First best one is the best solution fitness and the second best value tracked by PSO.

Objective function is a multi-modal that is more than one function, this implies that a process can have more than one optimal but can either have a single or more than one global optima .Finally the local optimum lack global perspective as will be seen in the optimization results. Particle swarm optimization usually provide better results however Genetic Algorithm are better it is only that the results are not definite [22].With a modified particle swarm optimization algorithm it can be used to tune the PID parameters [23]**.**

2.4.6 PSO implementation procedure

For effective implementation of the particle swarm optimization the following sequence of activities were strictly followed: Initialize the swarm P(f) of particles such that the position x_i (t) of each particle $P_i \epsilon P(t)$ is random within the hyperspace with t-0,Evaluated the performance F of each particle using its current position x_i (t),Compared the performance of each individual to its best performance thus far : if $F(x_1)$<pbest then this results into $F(x_i(t)$ then $X_{pbest} = X_i(t)$,

Compared the performance of each individual to the global best particle :if F(xi(t) < gbest then gbest is equal to F(xi(t) hence Xgbest = Xi(t),Change the velocity vector for each $vi(t)=vi(t-1)+\rho_1(Xpbest- Xi(t))+ \rho_2(Xgbest- Xi(t))$ whereby ρ_1/ ρ_2 are random variable and second term(t) is the cognitive component while the last term is the social component, Move each particle to a new position $x_1(t) =x (t-1) +v_1 (t)$ and

t=t+1,Procedure reverted to (b) and action done and re done until convergence is achieved.

The random variable ρ_1/ ρ_2 are defined as $\rho_1=r_1c_1$ while ρ_2 is given by r_2c_2 with r_1, r_2 lies between 0 and 1 and normally c_1 and c_2 are positive acceleration constant .NB: c_1 +c_2 must be less than or equal to four to guarantee stability in the use of particle swarm optimization. Parameters are initialized randomly in first step and then are being updated using particle swarm optimization algorithm, in each iteration one of the parameters set are being updated on each iteration sequence signified by a_1 and the second iteration is by b_1 and the sequence repeated.

2.4.7 Tuning of PID using Particle Swarm Optimization

Particle swarm optimization (PSO) can be defined as an optimization technique based on evolutionary computation. It was developed through the simulation of simplified social systems and is robust in solving continuous nonlinear optimization problems. PSO is a population-based stochastic optimization technique where individuals, referred to as particles, are grouped into a swarm. Every particle in the swarm represents a candidate solution to the optimization problem. Each particle is flown through a multidimensional search space, adjusting its position in search space according to its own experience and that of neighboring particles. Therefore, a particle makes use of the best position encountered by itself and that of its neighbors to position itself toward an optimal solution as described in Figure 7.

The effect is that particles fly toward a minimum, while still searching a wide area around the best solution. The performance of each particle is measured using a predefined objective (fitness) function, which encapsulates the characteristics of the optimization problem [24].

Each particle is treated as a point in a j-dimensional space. The i^{th} particle is represented as $X_I = (xi_1,xi_2 \ldots xij)$. The best previous position (giving the minimum fitness value) of any particle is recorded and represented as $PI = (pi_1,pi_2 \ldots pij)$, this is called pbest. The index of the best particle among all particles in the population is represented by the symbol g, the velocity for the particle i is represented as $V_I = (V_{i1} \ldots V_{i2 \ldots} Vij)$.The Particle Swarm Optimization (PSO) algorithm is a computational optimization technique that can be used to find the optimal solution to a problem by

simulating the behavior of a swarm of particles moving in a multidimensional search space. The following is a brief description of the flow chart for the PSO algorithm:
Initialization: The algorithm starts by initializing a population of particles, each representing a possible solution to the problem. The particles are randomly initialized within the search space, and their position and velocity are defined.

Evaluation: Each particle's fitness is evaluated based on its position and velocity in the search space. The fitness function measures how well the particle satisfies the problem constraints and is used to determine the quality of the particle's solution.

Update particle's personal best: Each particle maintains its personal best position, which is the best position it has found so far. If the particle's current position is better than its personal best, the personal best position is updated.

Update swarm's global best: The swarm's global best position is the best position found by any particle in the swarm. If a particle's personal best is better than the current global best, the global best is updated to the particle's personal best.

Update velocity and position: The particles' velocity and position are updated based on the current position, velocity, personal best, and global best. The new position and velocity are calculated using mathematical equations that determine the direction and magnitude of movement.

Check for termination: The algorithm checks if the termination criterion has been met. If the criterion is met, the algorithm stops, and the best solution found by the swarm is returned. Otherwise, the algorithm goes back to step 2 and repeats the process until the termination criterion is met.

Output: Once the algorithm has converged, the best solution found by the swarm is returned as the output of the algorithm.

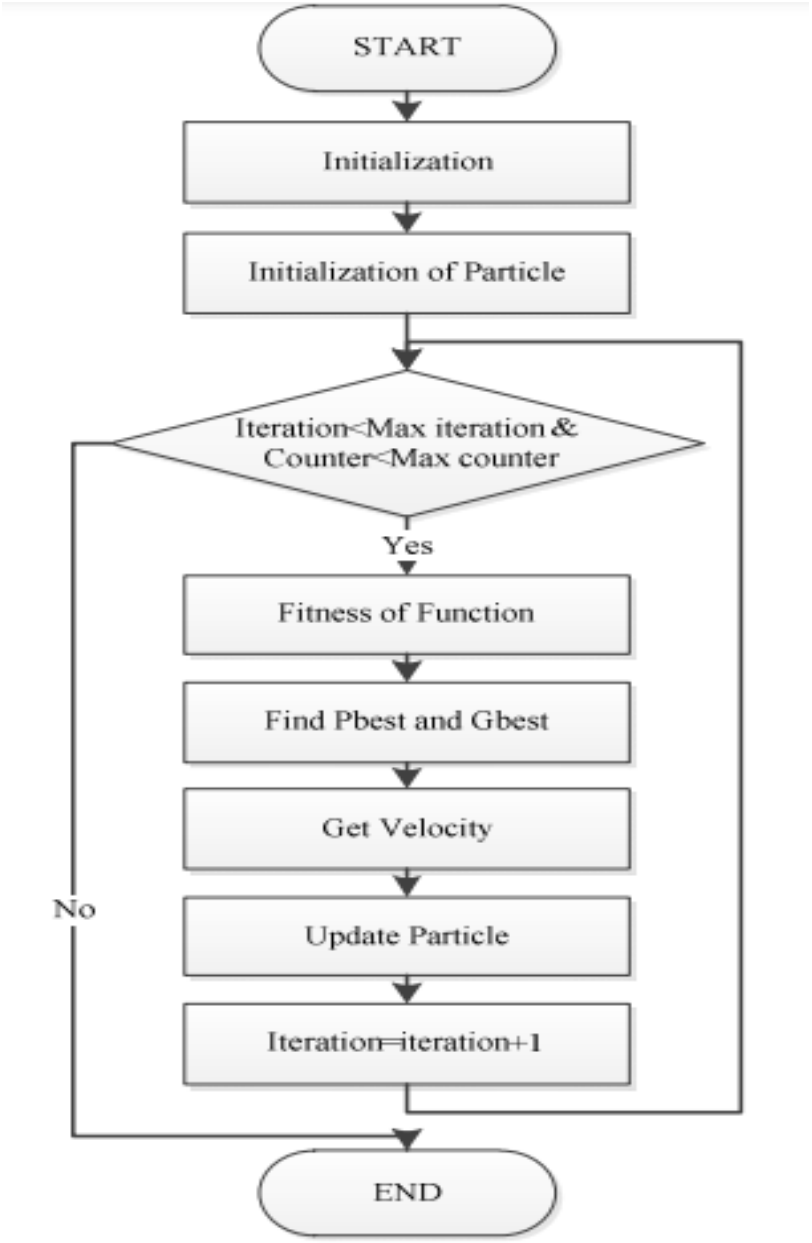


Figure 2.7. PSO flow chart

The particles are updated according to the following equations [22] Eq 2.5 and Eq 2.6

$$V_{ij}^{n+1} = w * V_{ij}^{n} + r_1 c_1 (P_{ij}^{n} - X_{ij}^{n}) + r_2 c_2 (P_{gj}^{n} - X_{ij}^{n}) \quad (2.5)$$

$$X_{ij}^{n+1} = X_{ij}^{n} + V_{ij}^{n+1} \quad (2.6)$$

Where w is the inertia weight, c_1 and c_2 are acceleration coefficients, r_1 and r_2 are random numbers uniformly distributed between 0 and 1, and n represents the iteration number.is used to calculate particle's new velocity according to its previous velocity and the distances of its current position from its own best experience (position) and the group's best experience. Then the particle flies toward a new position.

The performance of each particle is measured according to a defined objective function (performance index), which is related to the problem to be solved. c_1 and c_2 represent the weighting of the stochastic acceleration terms that pull each particle toward pbest and gbest positions. Low values, allow particles to roam far from the target regions before being tugged back. On the other hand, high values result in abrupt movement toward, or past, target regions. Hence, the acceleration constants c_1 and c_2 were set to be 2.0 according to past experiences. Suitable selection of inertia weight w provides a balance between global and local explorations thus requiring less iteration on average to find a sufficiently optimal solution [19].

As originally developed often decreases linearly from about $w_{max} = 0.9$ to $W_{min} = 0.4$ during a run. In general, the inertia weight w is set according to the following Eq2.7.

$$w = w_{max} - \frac{wmax - wmin}{itermax} X_{iter} \quad (2.7)$$

Where iter max is the maximum number of iterations (generations), and iter is the current number of iterations, the termination criterion used here is when the maximum number of iterations is reached which is 100 iterations. Considering that there are 15 and 30 individuals (particles) in the population. Since there are three parameters (K_p , $K_{i,}$ and K_D)to control, position and velocity are represented by matrices with the dimension of 2x2. The values of (K_p , $K_{i,}$ and K_D)are assumed to be from 1 to 100. The position represents the PID controller gains ((K_p , $K_{i,}$ and K_D) [25]. Normally to reduce the overshoot problem derivative action is added and mostly when solving an optimization problem the PSO Algorithm always searches for the optimum solution by adjusting the past moments and current search direction. However, if the optimizer needs to frequently change the descending direction and the present error is insignificant the history momentum will cause the overshoot problem.

2.4.8 Performance index

Quantitative measure to depict the system performance of the designed PID control. Using the PID optimum system can be designed and set of parameters adjusted to meet the desired specification. The major objective is to achieve minimum performance index. Performance index is the quantitative measure to depict the system performance of the designed PID controller which is tuned using PSO based on an improved algorithm where by the optimum system can be designed through parameters can be adjusted as per the specification [22].

2.4.9 Evaluation of the Objective Function

The performance of the control system is usually evaluated based on its transient response behavior. To design an efficient controller the objective is to improve the system response by minimizing the error between the input signal u (t) and the output y (t). In general, the error signal is expressed as:

$$e(t) = u(t) - y(t). \quad (2.8)$$

The error signal defined by Eq 2.8 is widely used in the integral performance criteria in the frequency domain such as the integral square error (ISE), integral absolute error (IAE), integral time square error (ITSE), and the integral time absolute error (ITAE). The ISE and IAE weigh all errors equally and independent of time. Consequently, optimizing the control system response can result in a response with a relatively small overshoot but a long settling time or vice versa. To overcome this problem the ITSE and ITAE criteria weigh the error such that late error values are considerably taken into account. The performance criteria formulas are as indicated in Eq 2.9 -2.14 [22].

a) Integral Absolute Error-

$$IAE=\int_0^t |e(t)|dt \quad (2.9)$$

b) Integral square Error-ISE=$\int_0^t |e(t)|^2 dt$ (2.10)

c) Integral Time Absolute Error=ITAE=$\int_0^t t|e(t)|dt$ (2.11)

d) Integral Time Square Error-ITSE=$\int_0^t t|e(t)|^2 dt$ (2.12)

e) Mean square Error-MSE =$\frac{\int_0^t |e(t)|dt}{t}$ (2.13)

f) Integral Error-IE=$\int_0^t e(t)dt$ (2.14)

However ITSE and ITAE can result in a multiple minimum optimization problem. In addition, all the integral performance criteria attempt to minimize only the error which does not necessarily mean minimizing all the basic evaluation parameters such as the maximum overshoot (Mp), rise time (tr), settling time (ts), and the steady-state error (Ess). A performance criterion in the time domain where the aforementioned parameters are evaluated and minimized was proposed) as shown in Eq 2.15 [24].

$$Fun = (1 - e - \beta).(Mp + Ess) + e - \beta.(ts - tr) \quad (2.15)$$

Where β is the weighting factor and can be set greater than 0.7 to reduce the overshoot and steady-state error or less than 0.7 to reduce the rise time and settling time. For this system, it has been found that setting β to 1.5 yields the best result.

Therefore, for tuning a PID controller based on the PSO technique, the performance evaluation indices ITSE, ITAE, and Fun will be used as the objective function and their responses will be compared with each other to find the optimal set of the PID controller gains. The structure of the PSO PID controller with its performance is evaluated using the objective function System error can be used as an evaluation index of its control effect. This study selected ITAE as the objective function. The fitness function controls the error which is given by the ITAE equation [26]

$$\text{ITAE= F (e, t) =J=}\int_0^{\infty} |e|tdt \quad (2.16)$$

ISE results in poor settling time while ITSE is the best method for getting minimum index. The objective function provides a measure of how individuals have performed in the main problem domain [19].

$$\text{ITSE=}\int_0^{t} te^2(t)dt \quad (2.17)$$

Where e (t) is the error signal

$$V_c\text{=Kpet (t)} + K_i\int_0^t et(t)dt + \text{Kd } \frac{det}{dt} \text{ (t)} + K_e\int_0^t t(et)^2(t)dt \quad (2.18)$$

The modified PID controller gain and the constant K are used to optimize the system's total error. The following listed parameters represent values for each as stated:

i = 1, 2, 3…n

m= 1, 2, 3…d

n = number of particular population in a group-

d = dimension of space or number of tuning parameters

t = current iteration value

v_{im}= velocity of a particle I at iteration t

w= inertia weight factor

c_1/c_2= learning factors

Random number r between 0 and 1 while x_{id} represent current position of particle i at t iteration and pbest is the best particle position of ith particle and finally gbest is the best particle position among all the particles in the whole population.

[22]The objective of performance index is to seek a set of PID parameters such that the feedback control system has minimum performance index .PSO tune parameters Kp and others using a standard model equation .PSO produces initial swarm of particles in search space represented by a matrix. Each particle represents a conditional solution for the PID parameters where the values are set in the range of 1 to 100.For

the three dimensional problem position /velocity are represented by matrices with dimension of 3 times swarm size. Step response of the performance is evaluated based on the rise time and settling time.

Reasons why PSO is preferred in tuning simple architecture is due to acceptable performance and ease of use. PID controllers do not require the use of extensive back ground in mathematics, control theory engineering to understand however the goodness of the traditional methods is that it provides us with initial PID values for optimal tuning. PSO tuning is basically used to handle the following types of problems: nonlinear, on differentiable, multiple optimum and high dimensional. [25]

2.4.10 Improved PSO

Introduction of inertia weight ensures reduced velocities over time, control exploration, exploitable abilities of the swarm, converge the swarm more accurately and guarantees convergence as per Eq 2.19 [17]

$$J_{ITAE}=\sum_{r=1}^{P} X \int_{0}^{T} t|e1(t)|dt \qquad (2.19)$$

To improve performance of PSO the use of linear decreasing inertia weight constriction factor approach is undertaken. Introduction of inertia weight ensures reduced velocities over time, P is number of controllers. Factors that determine the effectiveness of PSO is the searching capability, computational efficiency and convergence property. Each member is defined as a particle while the population is referred to as the swarm. Methods used to achieve convergence is velocity clamping, inertia weight, and constriction coefficient. The impact of inertia weight is on search capability however there are situations where the inertia rule does not apply.

As an intelligent optimization algorithm, PSO can be applied to determine the optimal parameters of different industrial or domestic functions. The major task is to look for the solution via collaboration and sharing information within the individual group's particle velocity update as provided by Eq 2.20 [27]:

$$V_{id}^{K}=wV_{id}^{k-1}+c_1r_1(P_{id}-X_{id}^{k-1})+ c_2r_2(G_{id}-X_{id}^{k-1}) \qquad (2.20)$$

Where by

W = inertia weight

c_1/c_2 = acceleration factor –ranging from 0 to 1

r_1/r_2 = random numbers

Particle position update is given by Eq 2.21

$$x_{id}^{k}=x_{id}^{k-1}+v_{id}^{k} \quad (2.21)$$

The purpose of PID parameter optimization is to find a set of parameters to make the PID achieve optimal control effect as per Eq 2.22 [19]

$$Ut= Kpe(t) +K_i\int_0^t e(\tau)\,dt + K_D\frac{de(t)}{dt} \quad (2.22)$$

Where e (t) is the system error brought by the fluctuations of plant dynamics. Alternatively, the new position can be obtained from Eq 2.23

$$V_{ij}^{t+1}=wV_1^tj+C_1V_1^t[P_{best}^t-X_{ij}] + C_2V_2^{t+1} \quad (2.23)$$

Where weights was added which resulted into Eq 2.24 [20]

$$V_{ij}^{t+1}=X\,[wV_1^tj+ \phi_1j(P_{best}^t-X^t{}_{ij}) + \phi_2\,(G_{best}^t-X^t{}_{ij}) \quad (2.24)$$

Where

X –constriction factor

W=inertia weight

$\phi_1/\ \phi_2=$ random values

The combined Eq 2.23 and Eq 2.24 results into Eq 2.25 with the addition of the constriction factor.

$$V_{ij}^{t+1}=wV_1^tj+ c_1X_{1j}^t[c_2G_{best}-X_{ij}] \quad (2.25)$$

2.4.11 Empirical analysis of PSO characteristics

The following factors show the analysis of Particle Swarm Optimization: Improved convergence speed, Control exploration trade off, Overcoming stagnation problem, Avoiding premature convergence, Velocity clamping technique, Boundary value problem, Initial and stopping conditions.

Normally Vmax = E (Xmax - Xmin) which gives the maximum velocity. From experience it was noted that a poorly chosen Vmax can lead to extremely poor performance. If a particle's personal best position and the neighborhood best position are far apart from each other the particle may follow a wider cycle leading to non-convergence.

Steps taken to achieve effective optimization using the PSO algorithm include particle initialization, particle selection, iteration, function evaluation, and stopping condition. It is worth noting that PSO is good at solving robust nonlinear optimization problems [19].

2.4.12 PSO Design analysis

To use PSO to tune the parameters of a PID system, the following steps were taken: in this study:-Defined the optimization problem: The first step was to clearly define the optimization problem, which in this case is to find the optimal values of the PID controller parameters, Defined the fitness function: The fitness function is a metric used to evaluate the performance of the PID controller. It can be defined as the integral of the squared error (ISE), the integral of the absolute error (IAE), or another suitable metric, initialized the particles: Initialize a population of particles, each representing a set of PID controller parameters. The particles can be initialized randomly or according to a specific distribution, evaluated the fitness of each particle: Calculate the fitness of each particle based on its set of PID controller parameters. This involves simulating the PID control system using the particle's parameters and evaluating its performance using the fitness function, updated the particles: Update the position of each particle based on its current position, its velocity, and the position of the best-performing particle in the swarm. This involves updating the parameters of the PID controller based on the position of the particle and its velocity, repeated steps 4 and 5 until convergence: Continue to evaluate the fitness of each particle and update their positions until the swarm converges on an optimal set of PID controller parameters. This involves repeating steps 4 and 5 for a specified number of iterations or until the fitness function reaches a minimum, implemented the optimized PID controller: Once the PSO algorithm has converged on an optimal set of PID controller parameters, the optimized controller can be implemented in the system. The performance of the system should be monitored to ensure that it meets the desired specifications.

Overall, using PSO to tune the parameters of a PID controller can be an effective way to optimize the performance of a control system. However, the success of the approach depends on the selection of appropriate parameters and the choice of fitness function. Additionally, PSO can be computationally expensive and may require significant computational resources to achieve optimal results.

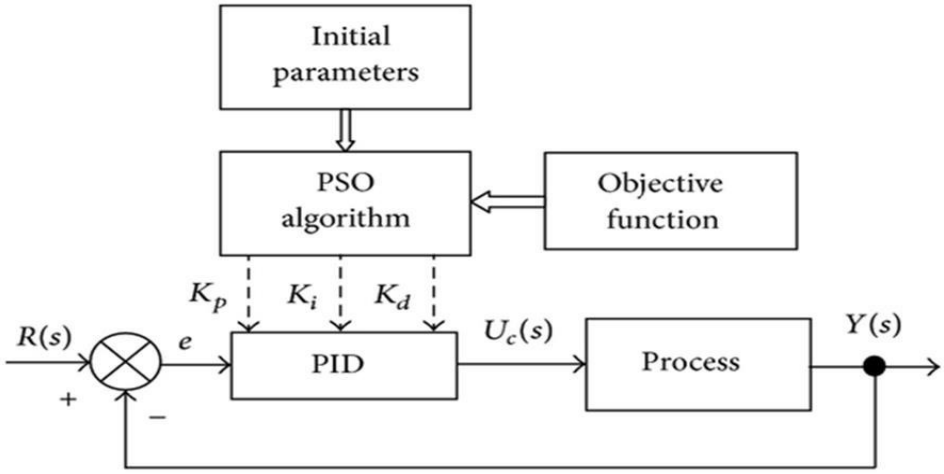


Figure 2.8 PSO/PID block diagram

Finally, the system is observed for the effectiveness in terms of process performance andresponse on the MATLAB/Simulink platform. The Figure 8 above shows the domestic plant unitfor the PID controller gains (Kp, Ki & KD) obtained using Ziegler-Nichols (ZN) method and the gains obtained using PSO with ITSE, ITAE and Fun performance assessment indices Comparative results for the plant response performance in the time domain are evaluatedbased on the rise time (tr), the settling time (ts), the maximum overshoot (Mp), and the peaktime (tp) [20].

It is clear from the responses that the PSO based controller has greatly reduced the overshoot and the settling time compared to the Ziegler-Nichols tuned controller. Furthermore, the PIDcontroller tuned using the PSO evaluated using the Fun assessment criterion (PSO - Fun) gives the best response as it reduces the overshoot to almost zero and has the fastest rise, settling and peak times. [25]The PSO global best (Best Cost) solution is selected for the particlewhich has the smallest value based on the Fun performance assessment criterion for the 100 iterations as shown above It can be seen that global best value is minimized throughout the iterations, and thus, the solution is optimized.

The PID controller gains obtained using PSO with its objective function evaluated using the Fun equation (PSO - Fun) are used to simulate the plant model. First, the original position and velocity of PSO are initialized and the fitness values of each particle are calculated. Then, for each particle, the fitness value is compared with the fitness value of Pbest. If the current value is better, the position is renewed with the current position and the fitness value is updated simultaneously. [18]After that, the best particle of the group with the best fitness value is determined. If the fitness value is better than the fitness value of gbest, then the gbest and its fitness value are updated with the position. The finalizing criterion is checked, and if satisfied, the iteration process is ended. Finally, the position and velocity of PSO are updated, and the procedure returns to step 2 above. Suitable selection of the inertia weight provides a balance between global and local searching capability.

To achieve process optimization, the use of a computation strategy is employed for self-tuning or automatic tuning of PID controller parameters. There are several ways to tune the PID parameters, such as analytical methods brought up by Lyaponov/root locus, heuristic methods worked on by Ziegler-Nichols & Cohen-Coon and frequency response method. An alternative is the adaptive tuning method, which is an online real-time tuning method involving the use of automated online mechanisms to tune the PID controller parameters. The complexity of the plant dynamics determines the difficulty of tuning the controller parameters [28]**.**

Sometimes the plant faces tuning problems which are normally a result of difficulties dueto environmental or operational uncertainties and perturbations, these features contribute tothe difficulty of obtaining appropriate or suitable controller parameters geared towards the achievement of set performance objectives. Additionally, existing applications require meeting numerous performance conditions or requirements at once and are usually in conflictwith the parameters selected. [17] Meta-heuristic methods have the following advantages: they are concurrent, have a simple implementation, can handle large and complex problems at reduced computational cost, and have wide

usage in many areas. The Teaching of Learning-based Optimization (TLBO) [20] is a simple concept but lacks an exact algorithm variant. However, the method is easy to implement and has fast convergence. The BAT met heuristic algorithm method is used for PID tuning and can solve realistic real-world problems with a unique ability to handle optimization problems. Bacteria Foraging Optimization (BFO) is a nature-inspired optimization method that imitates a swarm of Escherichia bacteria. For continuous monitoring and maintenance of plant dynamics in real-time the chosen controller should be highly sensitive to parameter variations and require lower processing time for a quick response.

We also have the fuzzy method [22] but its applicability to real-time systems is a major problem, and it also requires high human expertise because it is based on artificial intelligence. The major difficulty in parameter tuning arises when the controlled variables need to be tuned to achieve the following objectives: closed-loop control system stability, durability, performance tracking, measurement and noise reduction, disturbance rejection, and robustness against environmental changes. Generally, given a set of objective functions, there are several types of tuning strategies for the PID controller, such as model-free, non-parametric model, in-depth data method, and parametric model.

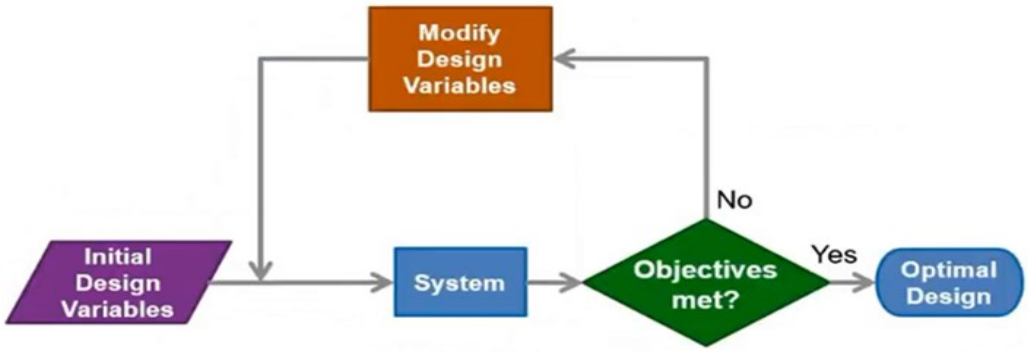


Figure 2.9 Particle swarm optimization design

The Figure 2.9 Above describe the process which was taken in ensuring correct optimization sequence:-The setting of the design variables and from there the system is set with the desired variables to see if the objectives are met if yes get the optimized design if not kindly modify the variables. Using the time domain requirement the

plant input model was designed for its use in the Simulink model [11].To improve on the process performance some parameters were set as follows:-the final value was set as 1 while the settling time was designed to 1, the initial condition was set to 0, the time domain set to continuous mode and finally the parameters K_i, K_p and K_d were initialized.

The procedure for initialization was done as follows:-On the MATLAB platform select Modelling, then go to model work space, assign values to the variables K_i, K_p then finally save the data as step response. The design variable set was added in order to affect the real optimized process of the plant. The variables K_i, and K_p were selected and transferred to the Simulink workspace model. The minimum value was set to be 0 for each variable and then OK button pressed to run the optimization.

2.4.13 Applications of PSO

The particle swarm optimization is used in wide areas such as antennae design, signal processing, networking, biomedical, digital circuits, electrical and robotics

2.5 PID Tuning works

Most plant control mechanisms are highly nonlinear and strongly coupled Multi-Input and Multi-Output (MIMO) systems that are largely influenced by external environmental conditions. To achieve the control objective, the use of conventional PID control techniques is applied. However, this strategy may not guarantee the desired performance, as determining the combination of desired settings for the parameters to be tuned is normally cumbersome and complex. Due to the disadvantages inherent in linear control methods researchers have resorted to utilizing advanced control techniques to deal with amorphous conditions in any domestic or industrial setup.

The PID tuning by using human beings as the main controllers of the process has a lot of human errors resulting in poor performance. Different plants usually require controller timers, counters, relays, and assorted controllers which in reality do not provide optimum performance. The change from the static state of the process to the dynamic condition causes significant variation in the process state which in turn produces inaccurate final poor output. The highlights and overview of the literature review are discussed below from different global authors:

After evaluating various techniques it was noted that the relay-based PID controller provided the optimal response using the Ziegler Nichols technique [29].The control of the temperature in a heat ex changer was analyzed in the studies [30] - [31] where the system took 45 seconds to reach the set point and 20 seconds to achieve stability during oscillation. The implementation of new controllers demonstrated a reduction in switching and the model predictive controller resulted in a smoother temperature control. The application of a simplified thermal model and optimal temperature control using feedback and feed forward techniques accurately predicted the thermal characteristics with average mean square errors of 11.14% and 12.82%, respectively [32].

Ant colony optimization (ACO) and PSO techniques were used in designing a plant control where ACO was proved to be slightly better than PSO. Distancing function technique was implemented and the method was found to be easy, very fast and easily implementable however [26] worked on the latest PID adjustment technique by using PSO-SCALF(Sine Cosine Algorithm and Levy flight) form optimization which resulted in Overshoot decreasing from 25% to 10% respectively. Optimal tuning of PID controller gain based on PSO method using the integral time square error (ITSE) performance index and the efficacy of the controller is tested with trio-loop error at different transient conditions [28] - [33] where it was noted that ITSE is fairly reduced.

The use of Fuzzy Logic Control /Genetic Algorithm and GA-PID schemes for stability control of nonlinear systems was examined [34]- [35] and the researchers could not determine which controller was superior to the other. Simulated results showed an improvement in system performance compared to previously employed systems with faster steady-state error control output achieved through Genetic Algorithm-GA determined PID tuning parameters. A study on the determination of optimal performance of a temperature control system using PID Fuzzy Controllers is undertaken [29] [36]and it is realized that the temperature control using the fuzzy PID controllers had better performance based on transient response and error integral criteria. The problem of tuning required solution to achieve concise control.

A controller was tuned manually which lead to trial and error and could not meet the desired target compared to utilizing the Self-tuning PID controllers [37] [38]. In the context of Hybrid fuzzy control [29] [36]was fully studied and the effectiveness of each controller in performing specific tasks was noted, taking into account factors such as the fractional order rate of the error signal. Focus was then turned to conventional factors for the PID controller, whereas attempt to optimize the controller using a Genetic Algorithm resulted into slow response when it came to settling time, However the use of PLC for control functions showed poor performance in terms in terms of settling time.

The implementation of fuzzy proportional integral and derivative (FPID) was found to significantly enhance the step response characteristics of the system, improving rise time, overshoot, and settling time, as shown in studies conducted by [39]Additionally it was demonstrated that automatic control was superior to human control. Despite various studies investigating different plant models and control methods the unique structure, conditions, equipment requirements, environmental factors and operating conditions of individual plants make it difficult to obtain a universal control model for the process. [40] .

Fuzzy logic self-tuning PID controller for a ball mill grinding circuit using an improved distance observer was studied however the system suffered from instability problems. Another researcher investigated bumping switching methods for PID controllers in a nuclear power DCS control system and concluded that the PID tuning resulted in excessive overshoot. Finally a study was conducted [34] - [41] on simulation study on controller design for temperature control of a heat ex changer system and it was revealed that the system had a long heating time due to poor control mode.

Fuzzy self-tuning based their study on the design on an intelligent temperature controller of furnace system using the, however the system suffers from parametric reform complications, their effort to reduce over shoot, settling time, and raising time worked well hence showing that fuzzy method was capable of providing more accurate results with robust performance and very stable response compared with the conventional PID controller [3].

Improved back propagation (BP)neural network PID controller for greenhouse environment control was studied and focus was also directed to network traffic prediction using PSO with Elman mode which resulted into significantly improved performance compared to conventional tuning methods [42]- [43] . Neural network control system of unmanned auto vehicle attitude dynamic and its comparison with the PID control system was investigated [31] and it was realized that the neural network control was more superior in terms of performance. Another researcher [27] developed a tuned PID controller for speed control of DC motor using soft computing techniques and after analysis the soft computing method produced better results.

Root locus was used in designing a decentralized PID controller based on inverted decoupling for a Two Input Two Output (TITO) system and it resulted in achieving proper parameter control and regulation of process variable [44].

Big Bang Big Crunch (BB-BC) algorithm [13] was used to design a fast tuning of PID controller in a Heating Ventilation Air Conditioning (HVAC) while applying it to a second order system and noted that there was high speed achievement resulting into improvement in optimization, accuracy and high convergence speed.

Mathematical modelling and application of PSO was studied and it was concluded that optimization is a mathematical technique that ensures the finding of maxima or minima of some functions in some feasible region. Classical PSO Algorithm MATLAB codes was Developed [45]- [46] and it could solve three variable optimization problem easily and finally research on proportional integral derivative PID controller tuning for inverted pendulum using PSO Algorithm was undertaken and it was concluded that rate of convergence was extensively improved by 3.17seconds. [47].

Investigation on the use of a PSO tuned PID controller to monitor the camera position in a UAV with a two-axis gimbals was undertaken and it demonstrated stable convergence and good computational efficiency, which outperformed the conventional PID controller [17]. A position control system was developed for a servo system using a PID controller tuned with soft computing optimization techniques and

the later provided better results compared to the PID controller alone. A novel PSO algorithm with adaptive inertia weight was proposed under different inertia weight strategies such as constant time, time-varying and adaptive inertia weight. A flexible inertia weight particle swarm optimization algorithm was developed, where the PSO performance depended on appropriate parameter selection strategies for fine-tuning the parameters with the introduction of a new nonlinear strategy for selecting the inertia weight. However identifying the appropriate parameter values can be challenging, and improper settings may lead to sub optimal results. [25]

Researchers [19]- [48].investigated a systematic clustering method based on PSO and compared it to other clustering techniques discussed in the literature plus the development of a novel optimization algorithm called Direct Bee Colony Algorithm, which was inspired by group decision making process of honey bees. This approach was found to produce unique optimal solutions and could be a suitable option for real-time online optimization problems when compared to other nature-inspired algorithms. However the method has limitations in that performance of PSO based clustering can be sensitive to the initial placement of clusters. If the initialization is not well-balanced, it may lead to sub optimal solutions.

High resolution PID temperature control was designed for use in low cost therm-luminescence system and it noted for its simplicity in structure and applicability to variety of process work was also done on a simple PID based strategy for PSO Algorithm [49] - [50] and the method suffered from premature convergence.

Meta heuristic algorithm for PID parameter tuning review of different approaches for various scenarios was undertaken and they noted quite improvement in the settling time in terms of performance, however authors [51]- [52]worked on modelling and simulation of temperature control system of farm products baking equipment, the author concentrated more on micro controller based control and not tuning for optimum performance

Fuzzy self-tuning PID based research was used [53] - [54]on the design of an intelligent temperature control system while [14]concentrated on temperature control of water bath system using PID controller, further work was carried out on the design

of fuzzy based PID algorithm for temperature control of an incubator and they proved the effectiveness of the fuzzy logic controller as superior to PID. Many fuzzy self-tuning PID controllers require initial training data to establish the fuzzy rule base and membership functions. Gathering and reprocessing this data can be time-consuming and may not always be feasible.

In the Hybrid fuzzy control [36] [55] such as the fractional order rate of error signal the ability of each controller to effectively perform specific task was noted. [56] Worked on PID controller based conventional factors while [57]tried to use Genetic Algorithm and the PID, however [58]used the PLC for its control functions and the results were not good when it came to settling time. [59] Researched on Genetic Algorithm based PID controller tuning approval for continuous storage tank reservoir and observed that the Genetic Algorithm to be of better response compared to the conventional PID controller. GAs explore a large search space, which means they may require a significant number of evaluations (i.e., fitness function calculations) to converge to a satisfactory solution. This can lead to a time-consuming tuning process.

A study on PSO based tuning of PID controller for load frequency control in a two-area power system showed that using the PSO technique for tuning the PID controller resulted in less overshoot and a less sluggish system with reduced Integral Absolute Error (IAE) [24].

PID controller was applied for MIMO process using an PSO to a multi-variable process with two inputs [60]- [61] and two outputs was investigated and a of PSO Algorithm for PID controller tuning was studied and the research concluded that it yielded the best results with a value of ITSE found as 0.5673. [62]

After examining the literature on optimizing PID parameters using IPSO algorithm for non-linear systems, it was observed that several researchers have worked on using the PSO algorithm for PID tuning in various applications such as greenhouse environment control, network traffic prediction, ball mill grinding circuit and temperature control of a heat ex changer system. However little in road has been achieved on comparison of performance index at various iterations, number of swarm particles and the variation of inertia weight by applying the right algorithm. The basic

performance of the plant needs further improvement in comparison to other research studies found in the literature review. From the above literature review many researchers focused on the application of PSO for PID tuning, but lack of comprehensive comparative analysis to existing systems such as PID and SPSO. A research gap is in the systematic comparison of improved PSO variants in the created algorithm for standard plant transfer function as done by [10].

2.6 Summary

This chapter outlines the comprehensive review of the existing literature related to the optimization of PID parameters using PSO algorithm for non-linear systems. It highlight the gaps in the literature and identify the research questions that will be addressed in the thesis.

CHAPTER THREE
METHODOLOGY

3.1 Introduction

This chapter discusses the approach taken to achieve the set specific objectives of the study. Also discussed in detail are the Controllability status of the non-linear plant, Selection criteria for simulation software used in the study discussed, how to Design a controller system based on PID control in the MATLAB/Simulink platform, and Process Optimization of the PID parameters using the standard PSO and improved PSO algorithms. Limitations and assumptions are made with stated references.

3.2 Research Design

In this section there is the plan on how data shall be collected and evaluated to achieve research objectives, literature is review is done as described in detail in chapter two and the problem statement guiding the study is also explained. For the study there is need to have the MATLAB software 2022 version and knowledge of PSO. The study period is one year which shall cover events described in the flow chart in Figure 3.1

3.2.1 Data Collection

MATLAB/Simulink Simulation data were used to achieve the set objectives of the study. Journal articles, conference proceedings, textbooks and web pages were the main sources of secondary data used to review the study on Optimization of Proportional Integral Derivative Parameters based on Improved Particle Swarm Optimization algorithm for non-Linear Systems.

3.2.2 Study Hypothesis

This section predicts what the study will yield at the end of everything. The following two hypothesis will be tested: a) the plant under study is controllable is it true or false? b). IPSO Algorithm performs better than SPSO and PID in terms of ITAE, is it true or false?

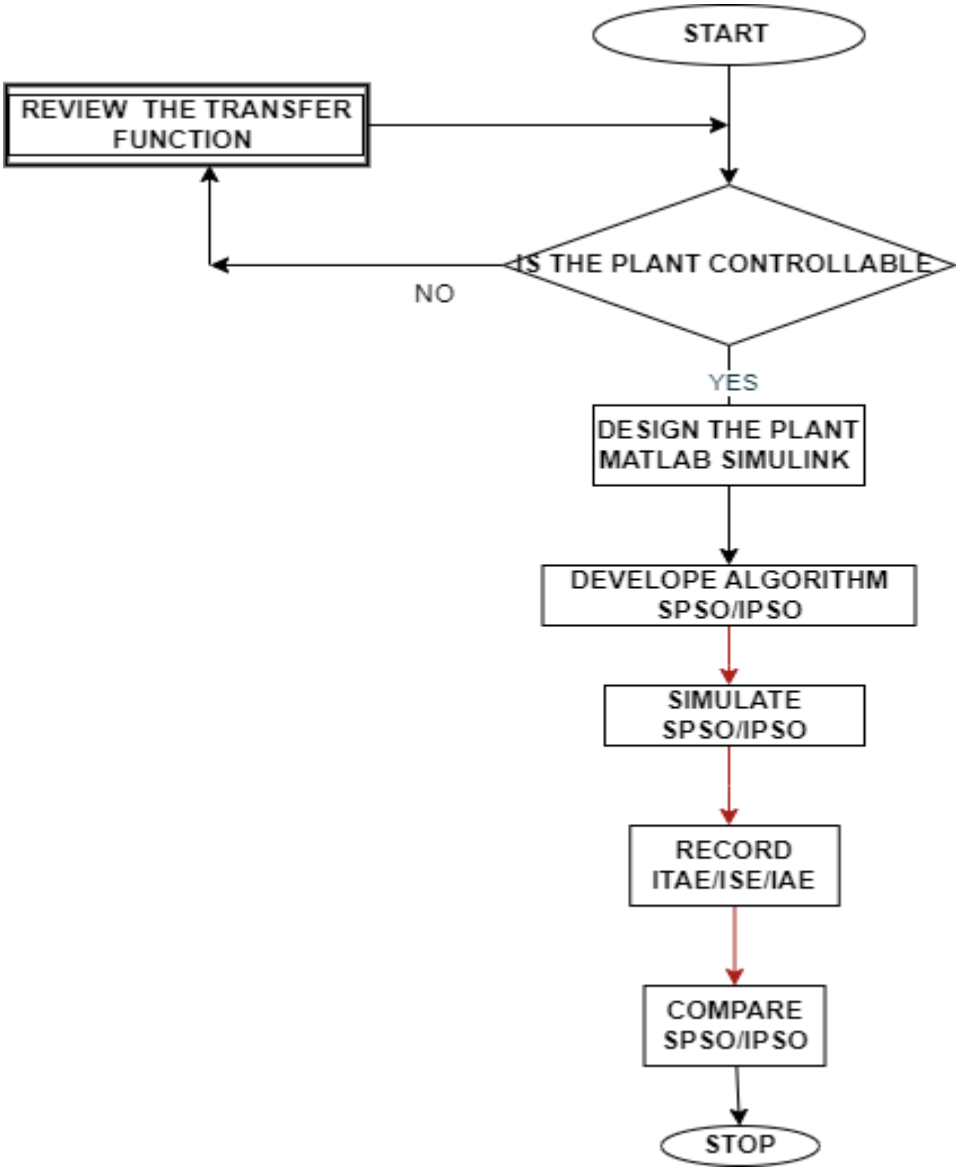


Figure 3.1 Study Flow chart

3.3 Controllability status

In this section the controllability status of the plant is determined from the plant transfer function as used in [10] as stated in Eq 3.12 .It always is important to determine if the process is controllable [7] before carrying out any process simulation. The system parameters are checked and thorough calculations done. Let the state matrix be stated as follows for the controlled process**:** A, B and C, where M_C = (B AB A^2B……$A^{n-1}B$) but since it is a 2x2 matrix n=2, the value of M_C = (B AB) and AB is obtained by making M_C= (B AB A2B).Note that for the process to be controllable the determinant should not be 0 for any matrix elements. The determinant is calculated from the solution of final the matrix [7] .

The second stage is to obtain the plant transfer function which is given by G=C (SI-A)$^{-1}$B after getting the inverse and the determinant.

$$\text{T.F} = \frac{G}{1+GH} \tag{3.11}$$

$$= \frac{Y)s)}{U(s)} = \frac{12.8}{16.7s^2 + s + 1} \tag{3.12}$$

After cross multiplying

$(16.7s^2 + s + 1)Y(s) = 12.8\ U(s)$ Getting Laplace Transform for both sides

$$16.7\ \ddot{y} + \dot{y}(t) + 1y = 12.8u(t) \tag{3.13}$$

This can be approximated to $17\ \ddot{y} + \dot{y}(t) + y = 13u(t)$ (3.14)

Note that

$x_1 = y$

$x_2 = \dot{y}$

$x = \ddot{y}$

$x = \dddot{y}$

Also

$x_2 = \dot{x}_1$

$x_3 = \ddot{x}_2$

$x_4 = \dddot{x}_3$

Hence $x_3 = 13/17\ u(t)\ -x1/17x_2(t)-1/17x_1(t)$

Now getting the space equation

$$\begin{bmatrix}\dot{x}1\\ \ddot{x}2\end{bmatrix} = \begin{bmatrix}-1/17 & 0\\ 0 & -1/17\end{bmatrix}\begin{bmatrix}x_1\\ x_2\end{bmatrix} + \begin{bmatrix}0\\ 13/17\end{bmatrix} u$$

$$y = \begin{bmatrix}1 & 0\end{bmatrix}\begin{bmatrix}x_1\\ x_2\end{bmatrix}$$

From the above

$$A = \begin{bmatrix}-1/17 & 0\\ 0 & -1/17\end{bmatrix} \qquad B = \begin{bmatrix}0\\ 13/17\end{bmatrix} \qquad C = [10]$$

For the system to be Controllable

Qc = [B AB A2B …….]

We need to find the matrix AB

$$= \begin{bmatrix} -1/17 & 0 \\ 0 & -1/17 \end{bmatrix} \begin{bmatrix} 0 \\ 13/7 \end{bmatrix} = \begin{bmatrix} 0 \\ -13/289 \end{bmatrix}$$

Therefore

$$Qc = \begin{bmatrix} 13/17 & 0 \\ 0 & -13/289 \end{bmatrix}$$

Getting the Determinant of Qc which is |Qc|

= 13/17x -13/289 = -169/4913

The system has rank n = 2 which shows the matrix is non-singular and also the determinant is not equal to zero hence the plant used in the improved particle swarm optimization is fully controllable and hence can be further used to study the performance analysis.

3.4 Selection of Simulation software.

Engineers worldwide over use MATLAB to analyze and design the systems and products transforming our world to a more technological environment. The MATLAB toolbox is the most efficient platform to express computational mathematics in the modern world. Since it is built-in graphics it easy to visualize and gain more insights from data. To model and simulate a multi domain physical system, the platform enables rapid creation of a model's physical system within the Simulink work space .With the software, physical component models can be built based on physical connections that directly integrate with block diagrams and other modeling paradigms.

Simulink, integrated with MATLAB, supports system-level design, automatic code generation, simulation and continuous test and verification of embedded systems. It is a block diagram environment for multi domain simulation and Model-Based Design and it exports simulation results to MATLAB for further analysis. Simulink is used to model algorithms, linear and nonlinear physical systems using block diagrams. With Simulink, the modeled system can be simulated and the results viewed on the scopes and other graphical displays. Configuring the models to run in real time enables the use of hardware-in-the-loop simulation to test the controller. Design changes can be made earlier in the development process and hence reducing costs and shortening in the design cycle. Simulators are very useful in our day-to-day life since they have significant advantages as they can result into quick solution to industrial problems without getting the real components in place.

The need for a high level of accuracy implies that the equations are usually technically too complex to solve and comprehend. To achieve the above the use of high-capacity computers to perform very complex simulations come as a quick technical solution. The bigger the capacity of the computer the better it can serve a specific purpose. Normally, simulations take a long time and end up consuming a lot of mental energy and produce varying amounts of results which can make analysis cumbersome and tedious. Additionally it is very difficult to use simulations to conduct 'what-if' experiments as they sometimes take too long to run and can be expensive. Simulations only tend to work in problems where the basic engineering science and technology are well understood such as IPSO.

3.5 The plant system

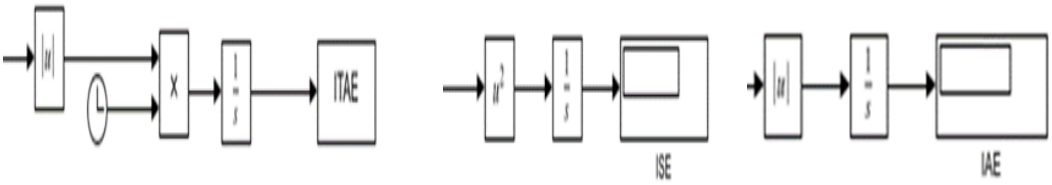
This section consist of the MATLAB Simulation, operation procedure, the plant design work and finally the Algorithm development and testing so as to validate results.

3.5.1. MATLAB Simulink operation procedure

MATLAB is usually started from the MATLAB tool strip then Simulink button is clicked, after that the blank model template is clicked, simulation editor open. Create the desired circuit components from the Simulation library. To save the project, select the save as with a name of the project.

3.5.2. Plant design work

This section covers the design work of the system and how the components will be connected to achieve desired plant operation .From the study and analysis done from different circuits which working towards PID parameter optimization, the following components are used: PID controller block, MUX block, Clock, Integrator, Step input, sum block, product, scope, clock, Abs block, square block, and ITAE, ISE and IAE display blocks. To achieve the desired task of parameter tuning using PID, SPSO and IPSO the following blocks shall be connected in ITAE, ISE and IAE block as shown in Figure 3.2 (a),(b) and (c) .



(a).ITAE circuit (b) ISE circuit (c) IAE circuit

Figure 3.2 ITAE, ISE and IAE circuit blocks

The fourth block of the design is the PID/Plant which consist of the summer PID, plant (second order function), MUX and Scope as described in Figure 3.3.In the diagram the MUX block transfer a combined signal to the scope while the sum compares the input with the feedback resulting into the error or deviation.

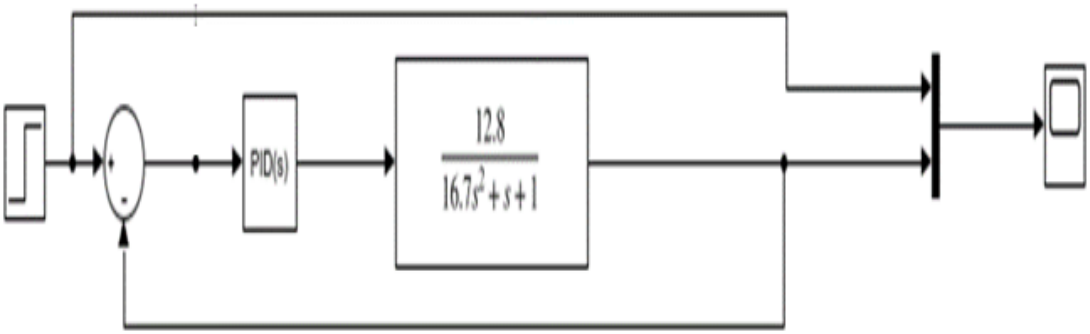


Figure 3.3 PID/ plant block diagram

The four blocks were then connected to come up with the circuit diagram described in Figure 3.3 which displays the arrangement of several components that will give results for analysis when run on MATLAB/Simulink with the standard and improved particle

swarm algorithms. The product block performs matrix multiplication while the clock outputs the current simulation time at each simulation step [64].

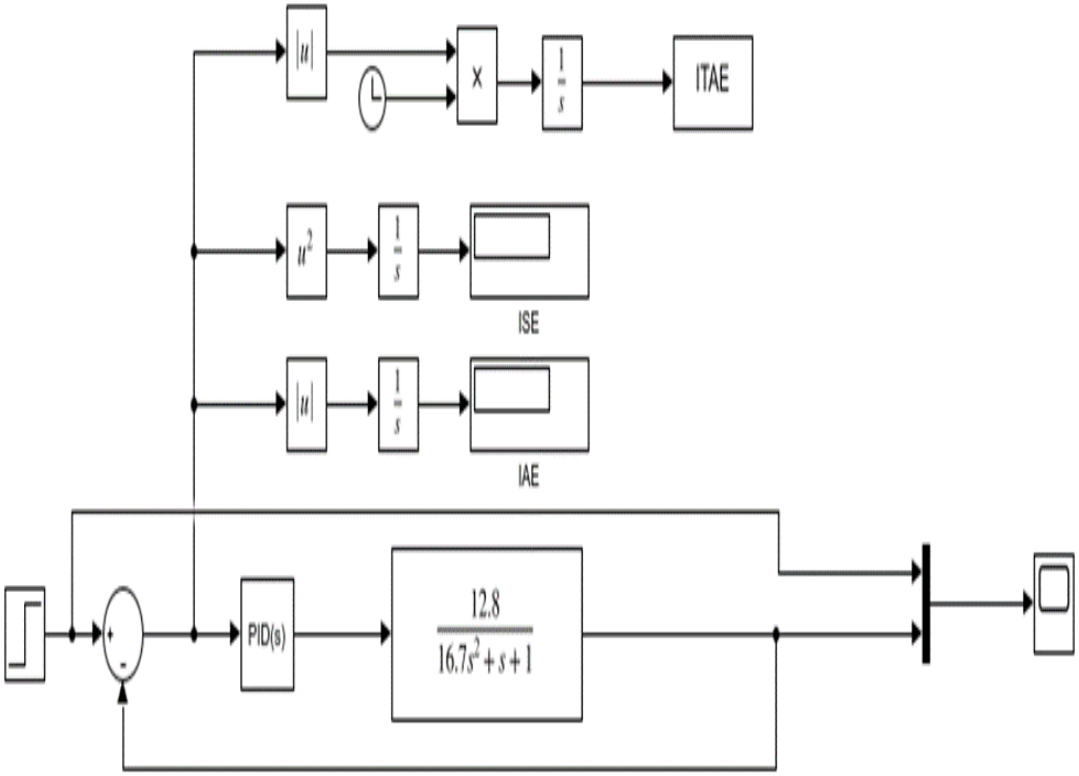


Figure 3.4 Combined block diagram

The PID block diagram consist of the process, Proportional integral and derivative blocks which acts on the error. The input is provided to the proportional, integral, and derivative sections simultaneously, and their corresponding outputs are added using an adder. The summed input is then provided to the process, which determines the corresponding parameters that are applied to the system. The process variable input and the actual input are compared to generate an error, and the system's output is adjusted accordingly. This process continues until the error signal reaches zero or until the process variable value equals the set point [65]. The purpose of PID parameter optimization is to find a set of parameters to make the PID achieve optimal control

3.5.3. Study Algorithm

3.5.3.1 PSO Algorithm background.

This section covers the development of the Algorithm that was used to come up with the codes which enabled solution to the specific objective. An algorithm is a finite sequence of rigorous instructions, typically used to solve a class of specific problems to perform a computation. The Algorithm adopted a set of particles flying over a

search space to locate the optimum point where the error is minimal. The background of PSO was discussed as explained in Figure 3.5 before coming up with the code, more details are fully covered in section 2.3 in chapter 2.

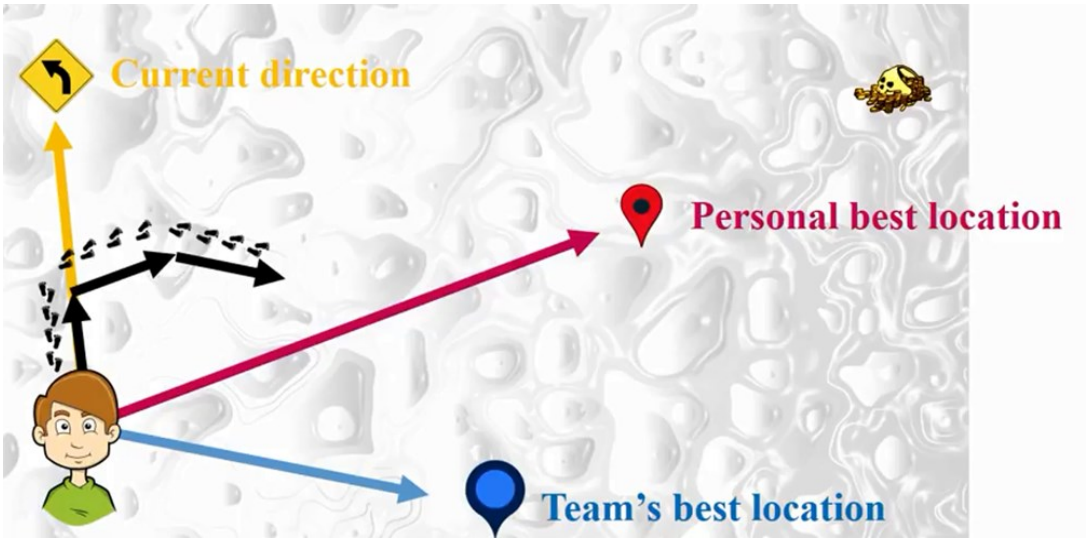


Figure 3.5 PSO swarm movement

The formulae adopted as the code is created is as shown in Figure 3.6 which consist of current velocity, personal best solution and the global best solution. Initial parameter of choice are the swarm size, position of the particle, velocity of the particle, acceleration coefficient and the maximum number of iterations.

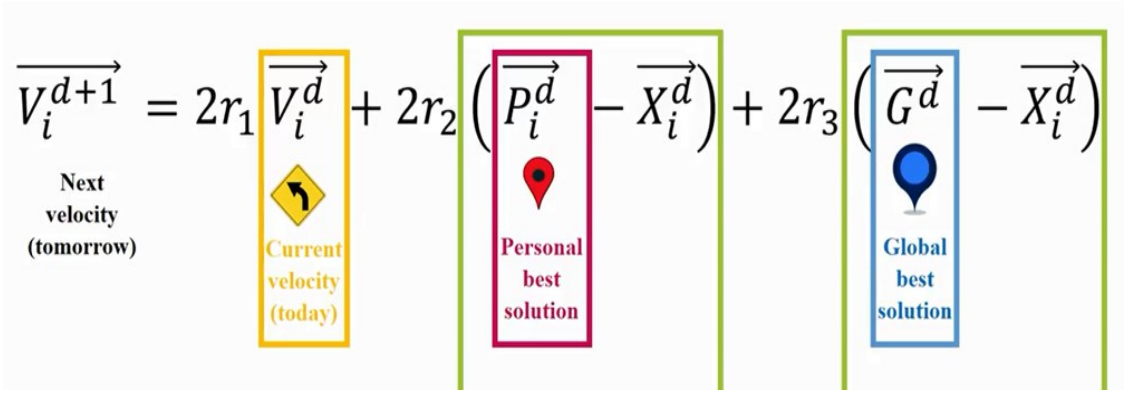


Figure 3.6 Velocity parameters

3.5.3.2 Phases of Algorithm coding

At the beginning objective function is defined, PSO parameters are noted, the Position and Velocity are initialized ,objective function is evaluated, the value of Pbest and Gbest are computed, the velocity and position are updated then finally the best value of the optimal value of ITAE,IAE and ISE is stored. The detail coding are in the Appendices.

3.5.3.3 Acceleration coefficient, Inertia weight analysis and random number

There are two types of acceleration coefficients that is Cognitive and social acceleration .The Cognitive Acceleration Coefficient (C_1)-This coefficient controls the extent to which a particle is influenced by its personal best (individual learning). A higher value of C_1 makes the particle give more weight to its own best-known position when determining its next move while Social Acceleration Coefficient (C_2)-This coefficient governs the degree to which a particle is influenced by the best-known position among all particles in the swarm (global or collective learning). A higher value of C_2 makes the particle place greater emphasis on the global best-known position. Summery is illustrated in Figure 3.7.For this study the value of C_1 and C_2 are set at 2 for PID, SPSO and IPSO.

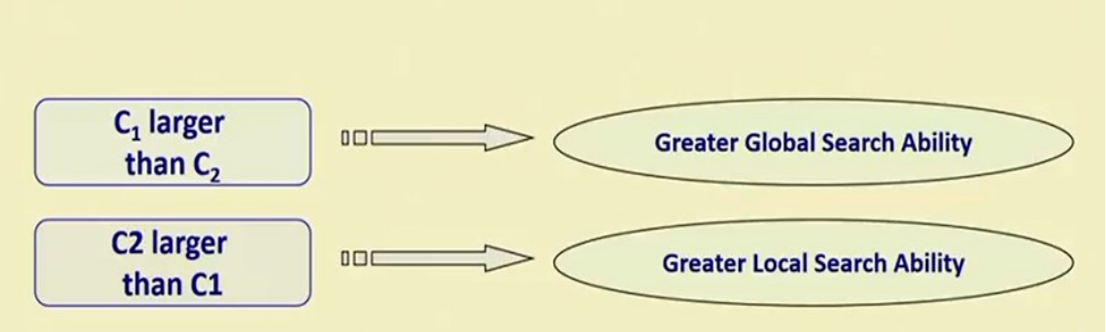


Figure 3.7 Effect of acceleration coefficient on search ability

Inertia weight is a crucial parameter in this study.It influences the balance between exploration and exploitation during the optimization process. In PSO, particles (representing potential solutions) move through a search space to find the optimal solution, and their movement is influenced by their own experience (personal best) and the experience of the entire swarm (global best). Inertia weight controls how much particles retain their previous velocity when updating their positions as explained in

Figure 3.8.in this study the inertia weight is set to 0.8 for SPSO while for IPSO it varies from 0.1 to 10 as iterations are run.

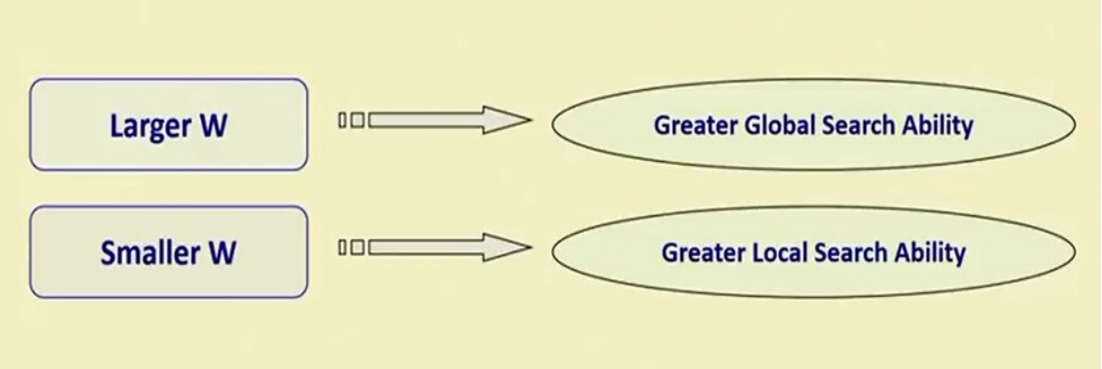


Figure 3.8 Effect of inertia weight on search ability

In the developed algorithm random numbers play a crucial role in introducing stochastic into the algorithm. Random numbers are used in various parts of PSO to ensure diversity in particle movement and to explore the search space effectively. One important use of random numbers in PSO is in the update equations for particle velocities and positions. For this study the random numbers was set between 0-1.

3.5.3.4 PID parameter setting at different iterations

Setting PID (Proportional-Integral-Derivative) controller parameters using Particle Swarm Optimization (PSO) involves optimizing the three gains, namely the proportional gain (Kp), integral gain (Ki), and derivative gain (Kd) to achieve desired control performance for a specific system. For Iterations set at 10: PID (Kp = 2 ,Ki = 0.1708, K_D =2),SPSO(Kp = 100, Ki = 37.11 K_D = 100) and finally for IPSO (Kp=100,Ki =15.27,K_D =100).For the second run the iterations was set at 50:PID (Kp = 2,Ki = 0.1708 K_D = 2), SPSO:(Kp = 100,Ki =15.67,K_D = 100) and finally IPSO:(Kp =100 Ki = 15.24,K_D = 100) and finally for Iterations 100:PID (Kp = 2 Ki = 0.1708 K_D = 2),SPSO(Kp = 2.61,Ki = 3.09 ,K_D = 100) and for IPSO:(Kp = 5.93,Ki =5.99,K_D = 100).The circuit was then tested in MATLAB SIMULINK together with the code which is in the appendices section. The results for each run was recorded and is discussed chapter four. Furthermore, the results obtained using different performance evaluation indices for the PSO tuned PID controller are shown and compared [26] [66].

3.6 Ethical issues

Research involving PID (Proportional-Integral-Derivative) tuning using improved particle swarm optimization (PSO) was conducted with the goal of optimizing control systems for various applications. The following ethical issues were considered while collecting and handling data: - Consent for Data Collection: Data Privacy and Security-Safeguarded collected data to protect the privacy and confidentiality of individuals, Bias and Fairness: Mindful of potential biases in the algorithms created, Transparency and Reproducibility: Ensured that the research methods, data, and algorithms were well-documented and stored, Publication Ethics: Adhered to ethical guidelines while publishing research Conflict of Interest: Disclosed all conflicts of interest, Environmental Impact: Considered related the environmental impact on the research . Dual-Use Concerns: Considered whether the knowledge gained could potentially be used for harmful purposes. However PID tuning research itself is not typically associated with dual-use concerns. Regulatory Compliance: Ensured compliance with NACOSTI regulations and laws governing research conduct in Kenya, Permission to conduct research was obtained from Murang'a University. The stated ethical considerations were fully addressed while conducting research on PID tuning using improved particle swarm optimization thereby contributing to the advancement of knowledge while minimizing potential harm.

3.7 Summary

This chapter outlines the methodology used in this research to achieve the intended objectives. The chapter continues to outline the framework that was used to show the research process, which explains how the research started which describe the mathematical models used to describe the non-linear system, the PID controller, and the PSO algorithm. It also provide details on the experimental setup, simulation tools, and performance metrics used to evaluate the effectiveness of the optimized PID controller.

CHAPTER FOUR
RESULTS AND DISCUSION

4.1 Introduction

This chapter presented the results obtained following the methodology adopted to achieve the set specific objectives of the study. This chapter deals with the testing, simulation, and discussion of different tuning techniques using MATLAB/Simulink. Simulink [52] is a software toolbox extension of the MATLAB program that is capable of simulating dynamic systems. Its major advantage is the ability to compute very complex dynamic systems simultaneously in a graphical environment with real-time visual programming.

The MATLAB graphical interface platform allowed for the selection of function blocks, interactive placement of functional parameters on the worksheet, and the description of signal flow just by a click of the computer mouse. The system is capable of simulating both analogue and discrete digital systems. The test bench used to realize the results consisted of MATLAB software, designed circuit, SPSO and IPSO Algorithm.

4.2 Controllability

As explained in section 3.2 and illustrated by Eq 3.11-3.14 the determinant was found to be -169/4913 which is the results after calculating the determinant of the matrix. For the system to be controllable the determinant should not be equal to zero. On observing the results the plant under study is controllable hence further analysis can be done.

4.3 PID, SPSO and IPSO Simulation results

The number of particles taken was assumed to be thirty and this was indicated in the algorithm so as to be automatically operated while the number of iterations was varied from 10, 50 then 100.The PID was set as described in section 3.4.3.3.The results are discussed in the following sections

4.3.1 Simulation at 10 iterations

The optimization process of PID involves tuning the proportional, integral, and derivative gains to achieve the desired system response. IPSO and SPSO are

optimization techniques that were used to automate this tuning process. IPSO used a hybrid approach that combined the standard PSO update equation with the gradient-based update method, while SPSO included a self-organizing mechanism and a mutation operator. Both IPSO and SPSO resulted in faster convergence and improved performance compared to the standard PSO algorithm and PID, as revealed by the results in Table 4.1.

Table 4.1: Comparison of performance indices at 10 Iterations at 30 Particles

ITEM	K_P	K_I	K_D	ITAE	ISE	IAE
PID	2	0.170800`	2	2.795100	0.31840	0.92270
SPSO	100	37.10776	100	0.061543	0.01141	0.03459
IPSO	100	15.26855	100	0.054806	0.01155	0.03159

Table 4.1 shows that the Integral Time Absolute Error (ITAE) is very high for PID with a value of 2.7951 and very low at 0.054806 for Improved Particle Swarm Optimization (IPSO). However, the Standard Particle Swarm Optimization (SPSO) is also better in terms of error reduction compared to the conventional PID.

For the case of integral square error (ISE), PID performed dismally with a value of 0.31840, while Standard PSO performed better compared to IPSO. For integral absolute error (IAE), the error is further reduced as the system runs from PID to SPSO and finally to IPSO.

The system response curve, as shown in the Fig 4.1, clearly shows that IPSO behaves much better compared to SPSO and the conventional PID. The magnified caption shows that the overshoot is much higher for SPSO as compared to IPSO. This directly

shows that the behavior of IPSO has improved the system behavior as a result of the improved algorithm.

For 10 iterations, convergence is achieved faster with IPSO compared to SPSO. IPSO settles much faster, while SPSO takes longer, as can be seen in the graphic representation in Fig 4.2.The inner zoomed graphic has system response in the vertical axis and the time in seconds in the horizontal axis,

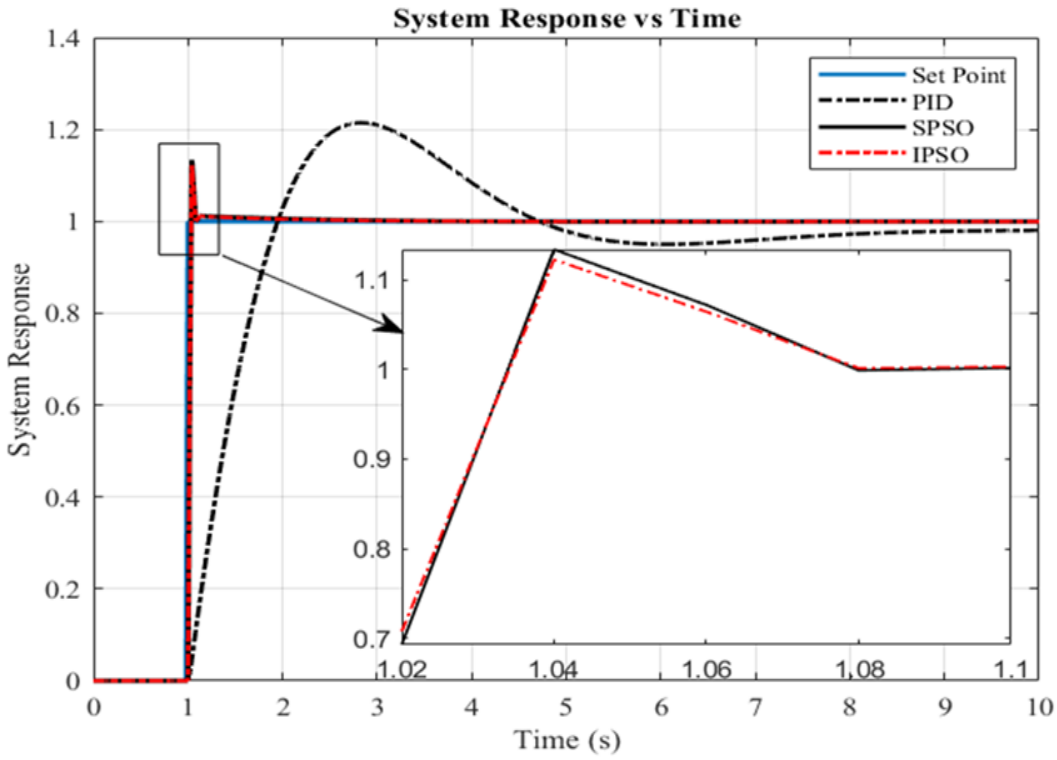


Figure 4.1 System response verses time for 10 iterations

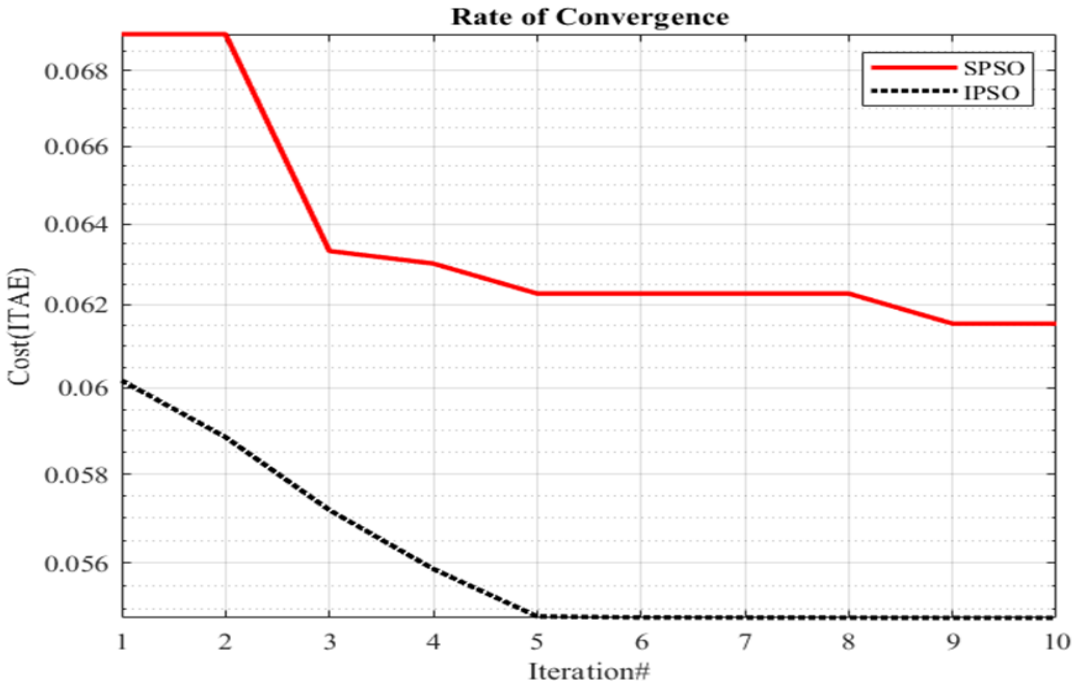


Figure 4.2 Convergence behavior for 10 iterations

4.3.2 Simulation at 50 iterations

Table 4.2: Comparison of performance indices at 50 Iterations and 30 Particles

ITEM	K_P	K_I	K_D	ITAE	ISE	IAE
PID	2	0.170800	2	7.326000	0.32080	1.14900
SPSO	100	15.66876	100	0.054696	0.01140	0.03200
IPSO	100	15.23772	100	0.054806	0.00155	0.03158

The Table 4.2 above shows that the Integral Time Absolute Error-ITAE is very high for PID with a value of 7.326000 and very low at 0.054696 for standard particle swarm optimization. However, the improved particle swarm optimization is better in terms of error reduction compared to the conventional PID, with a value of 0.054806.

For the case of ISE, PID performed dismally with a value of 0.32080. However, the improved PSO performed better compared to SPSO with values of 0.00155 and

0.01140 respectively. For IAE, the error is further reduced as the system is run from PID to SPSO and finally to the improved particle swarm optimization with values of 1.14900, 0.03200, and 0.03158 respectively. Overall, standard SPSO has better results compared to PID and IPSO while considering ITAE, however, for IAE, IPSO worked well compared to the rest.

The system response curve, as shown in the Fig 4.3, clearly shows that IPSO behaves much better compared to SPSO and the conventional PID. The magnified caption shows that the overshoot is much higher for SPSO as compared to IPSO. This directly shows that the behavior of IPSO has improved the system behavior as a result of the improved algorithm.

For 50 iterations, convergence is achieved faster with IPSO compared to SPSO. IPSO settles much faster while SPSO takes longer, as can be seen in the graphic representation in the Fig 4.4

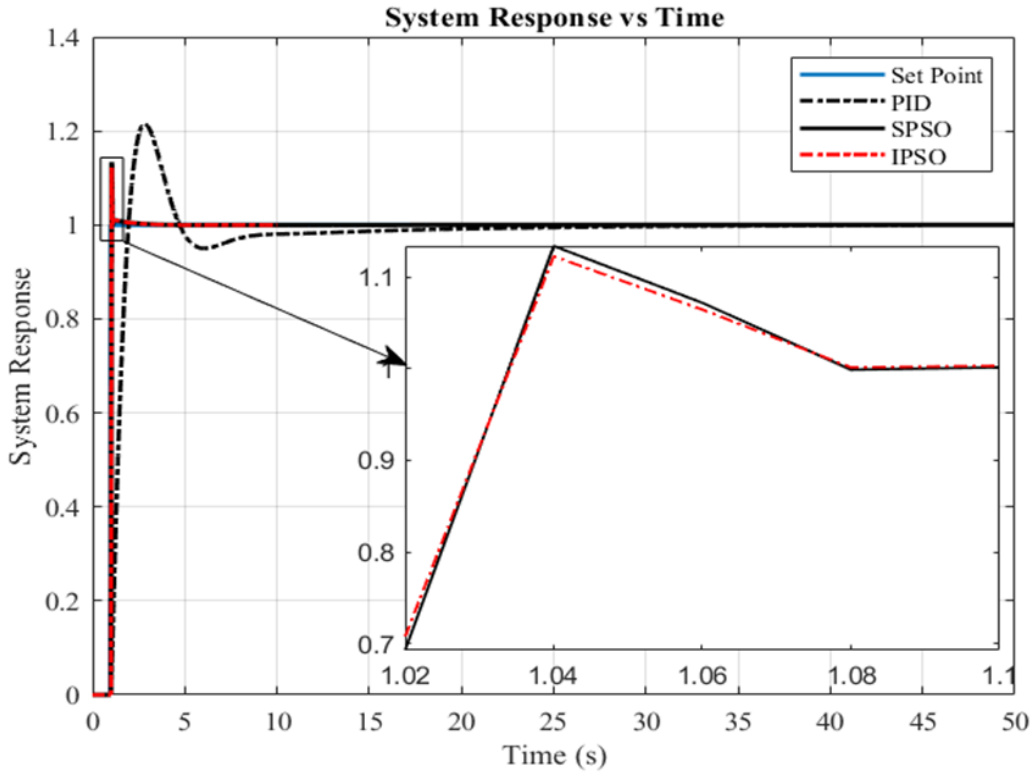


Figure 4.3 System response verses time for 50 iterations

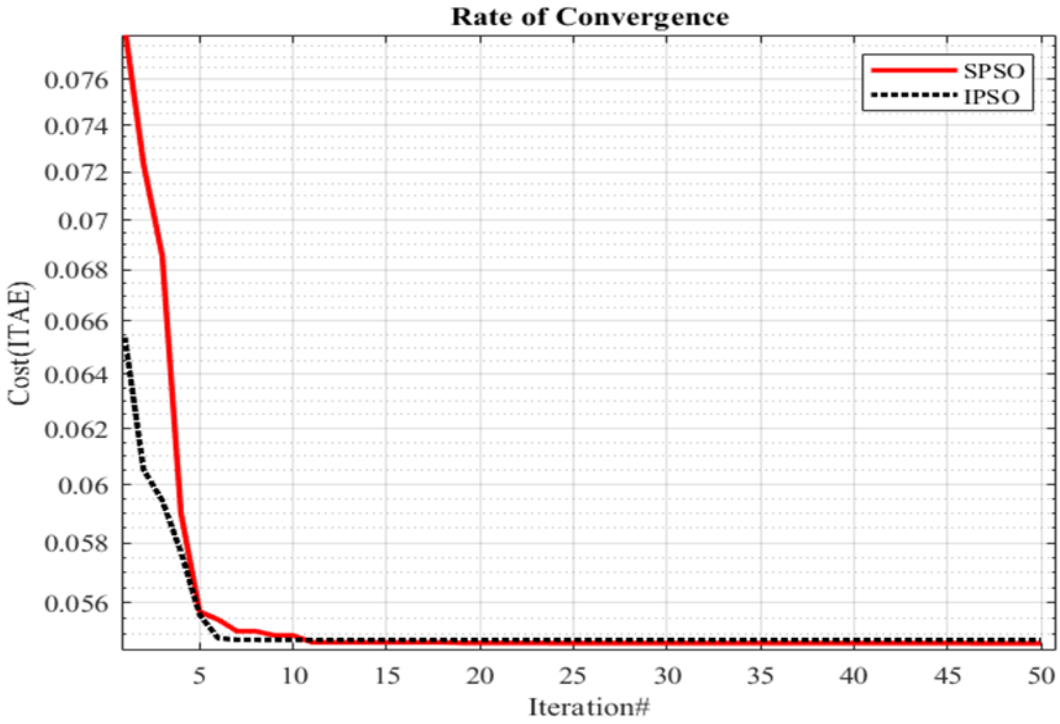


Figure 4.4 Convergence behavior for 50 iterations

4.3.3 Comparison of performance indices at 100

Table 4.3: Comparison of performance indices at 100 Iterations and 30 Particles

ITEM	K_P	K_I	K_D	ITAE	ISE	IAE
PID	2	0.1708	2	7.710200	0.32050	1.15500
SPSO	2.610987	3.09485	100	0.020476	0.01145	0.09648
IPSO	5.93141	5.991576	100	0.019001	0.01151	0.01869

The Table 4.3 above shows that the Integral Time Absolute Error-ITAE is very high for PID with a value of 7.710200 and very low at 0.020476 for standard particle swarm optimization. However, the improved particle swarm optimization is far better in

terms of error reduction compared to the conventional PID and SPSO, with a value of 0.019001.

For the case of ISE, PID performed dismally at 0.32050. Still, standard PSO performed better compared to IPSO at values of 0.011450 and 0.011510, respectively. For IAE, the error is further reduced as the system is run from PID to SPSO and finally to the improved particle swarm optimization, with values of 1.15500, 0.09648, and 0.01869, respectively. Overall, standard PSO has better results compared to PID and IPSO while considering ISE, but for ITAE and IAE, IPSO worked well.

The system response curve, as shown in the Fig 4.5, clearly shows that IPSO behaves much better compared to SPSO and the conventional PID. The magnified caption shows that the overshoot is much higher for SPSO as compared to IPSO. This directly shows that the behavior of IPSO has improved the system behavior as a result of the improved algorithm.

For 100 iterations, convergence is achieved faster with IPSO compared with SPSO. IPSO settles much faster, while SPSO takes longer, as can be seen in the graphic representation in Fig 4.6.The figures shows the response to step input and the settling time for each of the tuning method simulated.

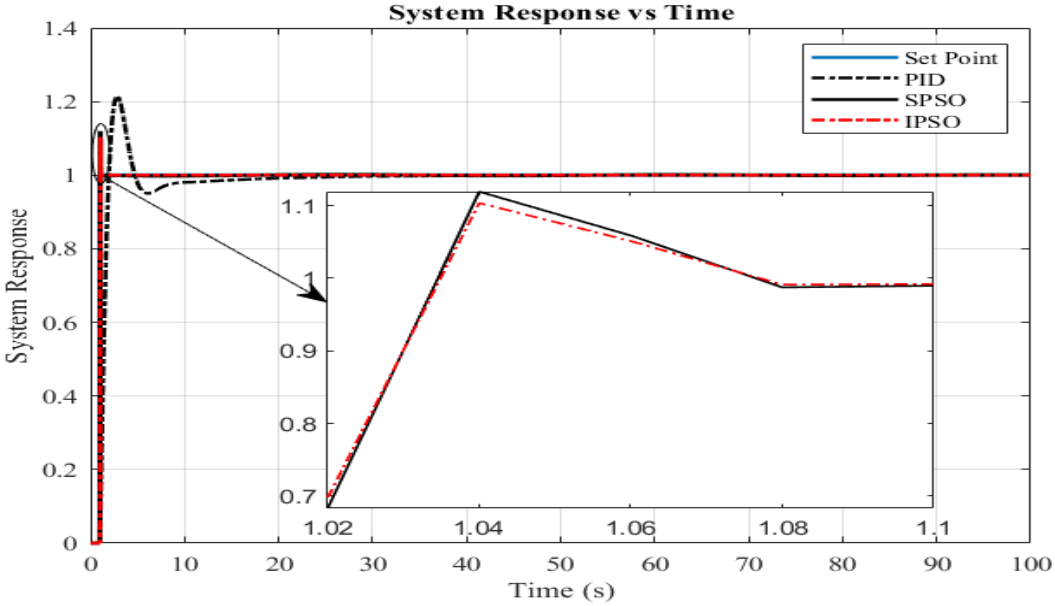


Figure 4.5 System response verses time for 100 iterations.

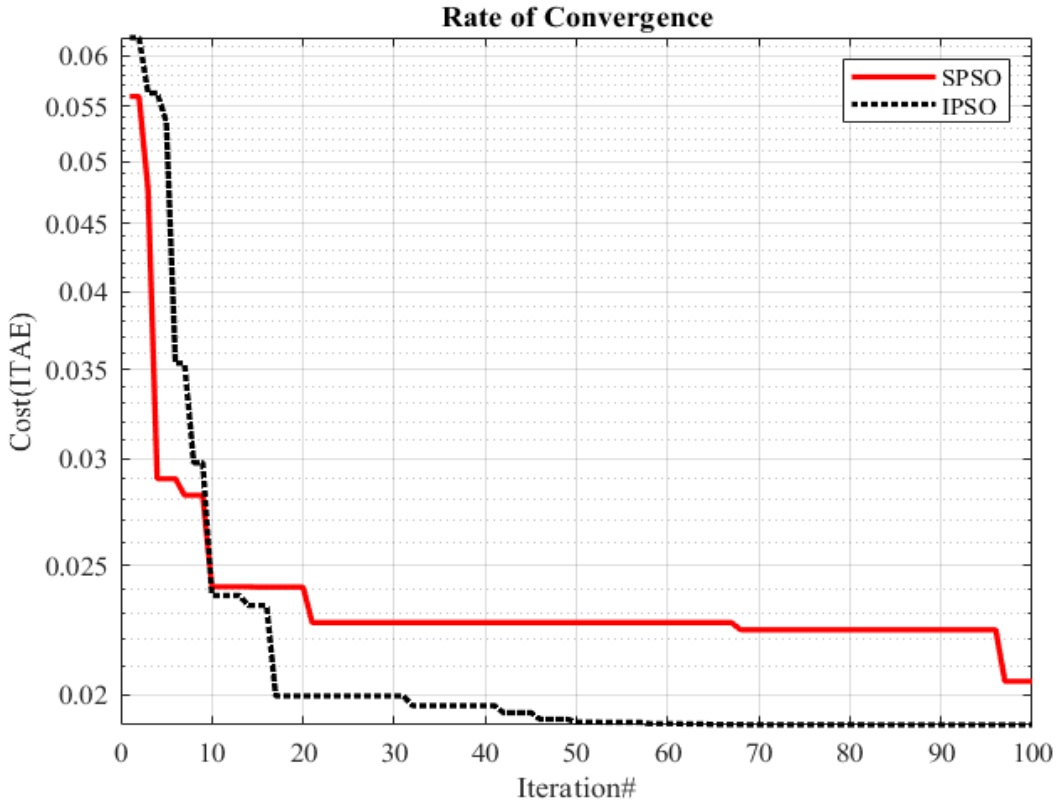


Figure 4.6 Convergence behavior for 100 iterations

4.3.3.1 Comparison of performance indices at 10, 50 & 100 Iterations

Table 4.4: Comparison of performance indices at 10, 50 & 100 Iterations

ITERATION	10			50			100		
ITEM	ITAE	ISE	IAE	ITAE	ISE	IAE	ITAE	ISE	IAE
PID	7.326000	0.31840	0.92270	7.326000	0.32080	1.14900	7.7710	0.32050	1.1550
SPSO	0.054696	0.01141	0.03459	0.054696	0.01140	0.03200	0.0205	0.01150	0.0960
IPSO	0.054806	0.01155	0.03159	0.054806	0.00155	0.03158	0.0190	0.01150	0.0187

When the number of iterations is changed from 10 to 50, as shown in Table 4.4, it can be seen that ITAE for PID, SPSO, and IPSO remain constant. However, there is a significant improvement in ISE and IAE under SPSO. It is noted that the error is reduced from 0.03159 to 0.03158 on IAE under IPSO. The finding that an increased number of iterations improves ISE and IAE is thus realized.

For 100 iterations, ITAE for PID increases to 7.7710, while for ITAE-SPSO, the error is reduced to 0.0205, which shows that ITAE is significantly improved. The same effect is realized with IPSO. The same is seen in SPSO-ISE and IAE, where there is significant improvement in the parameters. Therefore, this justifies the theory that an increase in iterations improves the error factor.

4.3.3.2 Comparison of performance indices at 15 and 30 particles

The Comparison of performance indices at 15 and 30 particles was tabulated for PID, SPSO, and IPSO respectively and different values for ITAE, ISE, and IAE were noted as illustrated on Table 4.5

Table 4.5: Comparison of performance indices at 15 and 30 particles

PARTICLES	**15**			**30**		
ITEM	**ITAE**	**ISE**	**IAE**	**ITAE**	**ISE**	**IAE**
PID	2.795100	0.31840	0.92270	2.79510	0.31840	0.92270
SPSO	0.056296	0.01139	0.03210	0.055119	0.01140	0.03294
IPSO	0.054806	0.01155	0.03150	0.054806	0.01155	0.03151

From Table 4.5 it can be noted that increasing the number of particles from 15 to 30 while maintaining other parameters does not affect the PID errors in terms of ITAE, ISE, and IAE. Which are maintained at 279%, 32% and 92% respectively.

However, it has a significant impact on SPSO-ITAE (5.6% to 5.5%) which is a reduction of 0.1%, where it reduces the error on ITAE while increasing the error on ISE and IAE (.3.2% to 3.3%).For IPSO, ITAE, ISE and IAE .remain the same at 5.48%, 1.16% and 3.15 % respectively. The study has clearly shown that IPSO, using the improved algorithm, can outperform SPSO and PID, and can be used in plant optimization in either an industrial or domestic setup.

4.4 Discussions

4.4.1 Effect of modified inertia weight

Particle Swarm Optimization is a nature-inspired optimization algorithm that is based on the social behavior of birds and fish. In a standard PSO, a population of particles moves through the search space with the aim of finding the optimal solution to a given problem. Particles adjust their positions based on their own best-known positions and the global best-known position within the swarm.

Improved or modified versions of PSO was developed to address some of the limitations and challenges associated with the standard PSO algorithm. These improvements as used in this study involved modifications to the inertia weight at different iterations and number of particles, this was done to enhance convergence and robustness.

The effectiveness of these modifications can be seen from the results indicated above from Fig 4.1 to 4.6 where the performance of the above improved PSO variants is compared against standard PSO on nonlinear system for a standard plant [11].Performance metrics included the rate of convergence and response time .

4.4.2 Effect of Iterations change on performance

As was realized from the study the performance of both standard Particle Swarm Optimization (SPSO) and its improved variants was influenced by the number of particles in the swarm and the change in iterations. The number of particles was a crucial parameter in SPSO as it affected the balance between exploration and exploitation. A smaller number of particles lead to more exploration, as each particle had a larger influence on the swarm's overall movement. This was beneficial in searching a larger portion of the solution space but suffered in terms of convergence to a precise solution as seen in the convergence curves Fig 4.2, Fig 4.4 and Fig 4.6.

4.4.3 Effect of large number of particles

A larger number of particles often enhances exploitation, making it more likely to converge to a good solution. However, it might also lead to premature convergence and the possibility of getting stuck in local optima. The optimal values for PSO

parameters (such as inertia weight and acceleration coefficients) was found to differ when using a smaller swarm hence for larger swarm needed parameter adjustment so as to maintain a good balance between exploration and exploitation. Generally a smaller swarm is associated with faster convergence but might suffer in terms of solution quality while a larger swarm may take longer to converge but has a higher chance of finding a better solution.

4.5 Summary

This chapter present the results of the study, which include the performance of the optimized PID controller under different operating conditions and disturbance levels. The results are analyzed and interpreted to provide insights into the effectiveness of the research in improving the control performance of the non-linear system using improved particle swarm optimization

CHAPTER FIVE
CONCLUSION, RECOMMENDATIONS AND FUTURE WORK

5.1 Conclusion

In this thesis, the plant transfer function was checked for controllability and from the mathematical calculation the determinant was not zero, therefore it was concluded that the plant can be further controlled using various methods. A standard domestic-industrial plant was taken as the objective function, IPSO and SPSO optimization methods were used to set up the PID parameters as per the created algorithm. In the PID controller parameter setting process, in addition to applying empirical formulae, it was necessary to tune the PID controller in combination with the actual site conditions and site experiences to achieve the most suitable parameters for the site conditions. Simulated results showed that SPSO and IPSO have a good effect on PID controller parameters tuning. The PID controller parameter tuning method based on the IPSO Algorithm improves the situation while the traditional tuning method requires multiple attempts to select the optimum parameters compared to IPSO and SPSO, which select the parameters automatically as per the algorithm within a short time. This study successfully developed an AI automatic PID tuning scheme using IPSO algorithm that can automatically acquire or readapt the PID parameter during plant operation in a routine way. The results showed the effectiveness of IPSO as a modern optimization technique and also to study the plant performance at different iterations and number of particles.

The simulation was performed on the plant transfer function using the MATLAB/Simulink platform at various parameter choices and situations, and realizations made from the data obtained. As the iteration was increased from 10 to 50 and then to finally 100, there was a significant reduction in ITAE error from 0.054806 to a minimum of 0.01900(5% to 1.9%in terms of percentages), which is also far better than the SPSO algorithm. SPSO reduced the error from 0.065143 to 0.020476(6.5% to 2%). When considering ISE, it was noted that PID increased the error factor significantly; however, SPSO and IPSO reduced the error at iteration 50 from 10 and then reduced the error further at 100. For IAE, the PID error gap increased from 0.92270 to 1.1550; however, for SPSO and IPSO, the error factor was drastically reduced as the iteration was raised.

It can thus be concluded that the number of iterations improves the error both for SPSO and IPSO hence ensuring that the plant performance is at the optimum point. From the data obtained from the simulation and after analyzing the graphs, it was noted that the performance of IPSO was found to be much better than SPSO and the conventional PID which justifies our main objective of this study.

While studying the degree of convergence, IPSO performed better than SPSO, however an increase in population from 15 to 30 while maintaining other parameters did not in any way affect the PID errors in terms of ITAE, ISE, and IAE performance indices. However, it had great impact on SPSO-ITAE, where it reduced the error on ITAE while it increased the error on ISE and IAE. The study has clearly found out that IPSO using the improved algorithm can perform wonders compared to SPSO and PID; hence, it can be used in plant optimization as a means of improving performance.

Future researchers will use this work as a springboard for quality research in areas that have not been deeply undertaken by my study. The work contributes to the study on PID controller tuning methods and limitations while using Meta Heuristic algorithms that have not been deeply explored in PID parameter tuning in industrial applications.

.

5.2 Recommendations

Based on my study on optimization of PID parameters using improved PSO algorithm for non-linear systems, the following recommendations can be made: Collect and analyze relevant data to accurately determine the behavior of the system, including identifying any non-linearity and disturbances, Consider incorporating additional control strategies, such as model-based control or adaptive control, to further improve the performance of the system and finally Continuously update and refine the optimization algorithm and control strategy as needed to adapt to changes in the system operating conditions.

5.3 Future Works

Some potential areas for future work on optimizing PID parameters using improved PSO algorithms for nonlinear systems include: Investigating the performance of different types of PSO algorithms, such as hybrid PSO algorithms, on PID parameter

optimization for nonlinear systems and the use of higher order function as the objective function in the improved particle swarm optimization.

Difficulties in mathematically proving why these algorithms are able to achieve the optimization process exist. Convergence of any other method has not been developed using mathematical models. Therefore, research can be done on developing more effective variants of the algorithm. Additionally, work can be done on the effect of the inertia weight on the performance and seeing the system behavior with more iterations, possibly more than 200.

Inventive approaches can be demonstrated by constructing new algorithms that combine distinct components of heuristic and meta-heuristic interactions in an intelligent and stable manner to provide effective algorithms that aggregate under specified conditions. Although the essential components of most algorithms are well understood, how they interact to achieve efficiency remains like a black box.

REFERENCES

[1] J. Iqbal and e. elal, "Non Linear control system," vol. 6, no. 4, pp. 301-312, 2017.

[2] P. A. Hokam and e. elal, Lecture notes on non linear systems and controln, Switzerland : ABB Ltd, 2018.

[3] S. I. Al-Amin, "Design of an Intelligent Temperature Controller of furnace System using the Fuzzy self tuning PID Controller," in *Intenational Conference on Electronics Communications and Information technology*, Khulna,Bangladesh, 2021.

[4] P. Kumar and V.NagaBabu, "Position Control of Servo systems using PID Controller Tuning with Soft Computing optmization Tecgnique," *International Journal of vEngineeriong Research and Technology,* vol. 3, no. 11, 2014.

[5] T. Astrom, PID Controllers, NEW YORK: ISA, 1995.

[6] Y. K. Singh and others, "Temperature control system and its control using PID controller," *IJERT,* 2016.

[7] M.Gopal, Controllability, New age international , 2005.

[8] K. Pedersen and e. al, "A review of process Controllability," *Research Gate,* October 1999.

[9] DEWWOOL, "Nonlinear equations," *dewwool.com/examplesofnonlinearequations/.*

[10] M.W.Berry and R.K.Wood, "Chemical Engineering science," *Pergam Press,* vol. 28, pp. 1707-1712, 1973.

[11] P. Ogutu and a. et, "Humidity control scheme for Chicken Banda using PSO," *International Journal of Engineering and Advanced Technology,* vol. 1, no. 3, pp. 101-104, 2022.

[12] M. Maldini, "PID control Explained," https://maldus512.medium.com/pid-control-explained-45b671f10bc, 2018.

[13] A. Abdoalnasr Almabrok, Fast Tuning of PID Controller in the HVAC system using BB-BC, 2018.

[14] P. Kativa.T, "Temperature Control of water Bath System using PID Controller," *International journal of Applied Engineering Research,* vol. 10, no. 4, 2015.

[15] W. Nic, "Tuning Method for PID Control Parameters," *IEEE,* 2020.

[16] A. Rakesh and P.Borase, "A review of PID Control tuning methods and Approaches," *International Journal of Dynamics and Control,* pp. 818-827, 2021.

[17] R.J.Rajesh and C. Ananda, "PSO Tuned PID Controller for Controlling Cemera position," *IEEE,* 2015.

[18] J. Kennedy, "The behaviour of Particle Swarm," in *7th International conference on evolution program*, 1998.

[19] S. Alam and others, "Rsearch on PSO based Clustering," 2017, pp. 60-73.

[20] M. N. Alam, PSO Algorithim and its code in MATLAB, india: Roukee, 2016.

[21] S. N. Ikhalas, "Optimization of PID control parameters with GA plus fuzzy logic stirred tank heater temperature control," *ICEECS,* 2017.

[22] J. A. El-Hadj, "Tuning PID using PSO for Controlling the temperature of the infant incubator," *IJCSN,* vol. 20, no. 3, 2020.

[23] J. S. R. Jang and C. T. Sun, "Neurofuzzy Modelling and control," in *IEEE 83*, 1995.

[24] R. N.R and R. P.R.K, "PSO based tuning of PID Controller for Load Frequency Control....," *International Journal of Engineeers research and Applications,* vol. 1, no. 3, pp. 1499-1505, 2012.

[25] M. J. Amshashy and others, "A Novel Flexible Interia Weight PSO Algorithm," *Open Access,* 2016.

[26] E. Y. Bejarbaneh and A. Bagheri, "A new adjustment technique for PID type fuzzy logic controller using PSOSCALF optimization algorithm," 2019.

[27] P.G.Medewar and R.K.Munje, "PSO based PID controller Tuning for PMDC motor," *International Journal on energy systems and Applications-ICESA,* 2015.

[28] V. Fevrier and a. et, "Fuzzy dynamic parameter adaptation in ACO and PSO for designing fuzzy controller –the case of water level and temperature control," 2018.

[29] E. D. Bolat, "Implimentation of MATLAB SIMULINK Based Real Time Temperature control for set point changes," *Internation Journal of Circuits Systems and Signal Processing,* vol. 1, no. 1, pp. 54-61, 2007.

[30] J. Agus and a. et, "Heat exchanger control based on artificial intelligence approach," *international journal of engineering and research,,* vol. Vol 11, no. No 16, pp. (pg. 9063-9082), 2016.

[31] Z. Zhao and Y. Lishe, "Simplified thermal model and optimal control for chilled water pipe in air conditioning system," 2019.

[32] A. Sisay and Achenef, "Fuzzy PID based temperature control of electric furnace for glass temperature control," Addis Ababa Institute of Technology, 2017.

[33] J. Dombi and A. Hissein, "A new approach to fuzzy control using the distancing function," *journal of process control,* 2020.

[34] H. G. Hamed Khodadadi, "Fuzzy Logic Self Tuning PID for Ball Mill Grinding," *Mining Engineering ,* vol. 72, no. 2, pp. 63-70, 2020.

[35] S.J.Basi, "Automatic Tuning of PID controller using PSO Algorithm," *International Journal of Artificial Inteliigemnce and Applications,* vol. 2, no. 4, 2011.

[36] M.Ahmad and Z.Mohamed , "Hybrid Fuzzy logic Control with Input Shaping for Input Tracking and Sway Suppression of Gantry Crane System," *American Journal of Engineering and Applied Sciences,* vol. 2, no. 4, pp. 241-251, 2009.

[37] H. Bushra and a. et, "Superheated steam temperature control using fuzzy logic," *ASRJETS,* 2007.

[38] S. Padhee and Y. Sing, "Performance evaluation of different conventional and intelligent controllers for temperature control of shell and tube heat exchanger system," 2011.

[39] S.Gottwald, "Mathematical fuzzy logic as a tool for the treatment of vague information," in *info-science*, 2005, pp. 41-71.

[40] J. S. Taur and a. et, "Temperature Control of a Plastic Extrussion Barrel Using PID Fuzzy Controllers," in *Industrial Automation and Control Emerging Technologies*, Taipei Taiwan, 1995.

[41] O. Zhang Fangjie, "Bumplta Switching method of PID controller of nuclear power plant," *Automation Panaroma,* vol. 01, pp. 92-95, 2017.

[42] Y. jiaping, "Improved neaural network PID control in green house environment," *Electronic Measurement Technology ,* vol. 42, no. 04, pp. 19-24, 2019.

[43] G. Zhaojun, "Network traffic prediction based on PSO," *Elman mode journal of modern Electronic Technique,* vol. 42, no. 01, pp. 82-86, 2019.

[44] S. W. Chnanchi Watthithanyawat, Design of decentralized PID controller with Root Locus method inverted decoupling, 2018.

[45] S. Taukdr, Mathematical Modeling and Application of PSO, 2010.

[46] J. S.B and D. E.G, "Proportional Integral Derrivative(PID) Controller Tuning for Inverted Pendulum using PSO Algorithm," *FUDMA Joutnal of Sciences ,* vol. 2, no. 2, pp. 73-79, 2018.

[47] D. Joseph .B, "PID Controller Tuning for Inverted Pendulum using PSO Algorithm," *Fuduma Journal of Science ,* vol. 2, no. 2, pp. 73-79, 2018.

[48] A. Sharma and others, Swarm and Evolution Computation, 2014.

[49] R. O. Mduduzi Mbongo, "Design of High Resolution PID Tempearature Control for use in Low cost Thermo Luminance system," in *SAIP*, South Africa, 2012.

[50] Z. Xiang and others, "A Simple PID based Strategy for PSO Algorithm," *Information Sciences ,* pp. 558-574, 2019.

[51] S. B. joseph and others, "Metaheuristcs Algirithm for PID parameter Tuning," Heliyon, 2022.

[52] L. Jun and others, "Modeling and Simulation of Tempearature Control system of farm products Baking equipment," in *International Conference on Computer Science and Electronics Engineering* , 2013.

[53] J. Jiang.W, "Design of an Intelligent Temperature control system based on the fuzzy self Tuning PID," *Procedic Engineering 43,* pp. 307-311, 2012.

[54] I. M. Rahman M.M, "Design of fuzzy based PID Algorithm for Tempearature Control of an Incubator," *International Journal of physical sciences,* vol. 1969, no. 1, pp. 20-55, 2021.

[55] T. K. Das and Y. Das, "Design of a room temperature and humidity controller using fuzzy logic controller," *AJER,* vol. Vol 02, no. Issue 11, pp. pg. 86-107, 2013.

[56] C. T. Chao and a. et, "An optimal fuzzy PID controller design base on conventional PID control and nonlinear factors," *mdpi journal of applied sciences,* 2019.

[57] L. A. Yusuf and N. Magaji, "Comparison of fuzzy logic and GA-PID controller for position control of inverted pendulum," 2015.

[58] Saidi and Sumurdi, "Room temperature control system industry based PLC," *ISDE,* Vols. Vol 6,, no. No 4, 2015.

[59] R. Joychitra, "Genetic Algorithm based PID Controller Tuning Approach," *Advanced in Artificial Intelligence ,* 2014.

[60] C.Y. C. Wei-der chang, "PID Controller Design for MIMO process using improved PSO," *Springer Science and businwss media,* 2013.

[61] O. Stimac Goranka, "Comparative Annalysis of PSO Algorithm for PID Controller tuning," *Chines Kournal of Mechanical Engineering ,* vol. 27, no. 5, pp. 200-204, 2014.

[62] N.A.Selamat, "Particle Swarm optimization for Mutlivariable PID Controller Tuning," in *IEEE 9th International Colloquium on Signal Processing and its applications*, Kuala Lumpar,Malaysia, 2013.

[63] D. J. (, "Modelling and simulation of temperature control system of coating plant air conditioning," *ICICT,* 2017.

[64] K.Manoj and P. Ashig "Tuning PID Controller for Speed Control of DC motor using Soft computing Technique," *A review advance in Electrical and Electronic Engineerin,* vol. 4, no. 2, pp. 141-148, 2014.

[65] W. Electronics, PID Working,types and Applications, 2022.

[66] https://www.tutorialspoint.com/modelling_and_simulation/index.htm#, "Modeling and Simulation," in *Tutorial*, 2023.

[68] M. Coban, "Fuzzy logic based temperature control," 2019.

[69] H. A. Mohammed, "Comparative study of optimal self-tuning fuzzy PID controller for speed controller of BLDC motor," vol. 4, no. 7, 2021.

[70] G. Nicolas and a. et, "An innovative adaptive control systems to regulate micro-controller conditions in green house," 2017.

[71] D. M. Atia and H. T. El-Madany, "Analysis and design of greenhouse temperature control using adaptive neuro -fuzzy inference system," *journal of electrical systems and information technology,* 2017.

[72] S. Padhee, "Controller Design for Temeprature Control of heat exchanger system simulation study," *WSEAD Transaction on systems and control,* vol. 9, 2014.

[73] Muliadi.J and Kusumputro.B, "Neural Network Control system of UAN," *Hindawi Journal of Advance Transportation ,* 2018.

[74] A. Nickabadi and others, "A novel PSO Algorithm with Adaptive Inertia Weight," *Elsiever ,* 2011.

[75] K.V.Pavarnsiker, "Investigation on PID Controller Tuning Mthods for Aircraft Fuse Lag Temperature Control," India, 2020.

APPENDICES

Appendix 1. Algorithm

The Algorithm for the program is as shown in the following codes

Iteration 10

```
close all
clc

% Define the details of the table design problem
nVar = 3;               % number of variables
ub = [1000 1000 1000]; %upper Bound
lb = [0 0 0];           % lower bound
fobj = @tuning;         % Objective function Name

% Define the PSO's paramters
noP = 15;               % number of particles for initialization
maxIter = 100;          % maximum iterations
wMax = 1;
wMin = 0.1;
c1 = 2;
c2 = 2;
vMax = (ub - lb) .* 0.2;
vMin  = -vMax;

% The PSO algorithm

% Initialize the particles
for k = 1 : noP
   Swarm.Particles(k).X = (ub-lb) .* rand(1,nVar) + lb;
   Swarm.Particles(k).V = zeros(1, nVar);
   Swarm.Particles(k).PBEST.X = zeros(1,nVar);
   Swarm.Particles(k).PBEST.O = inf;

   Swarm.GBEST.X = zeros(1,nVar);
   Swarm.GBEST.O = inf;
```

```
end

% Main loop
for t = 1 : maxIter

   % Calcualte the objective value
   for k = 1: noP
      currentX = Swarm.Particles(k).X;
      Swarm.Particles(k).O = fobj(currentX);

      % Update the PBEST
      if Swarm.Particles(k).O < Swarm.Particles(k).PBEST.O
         Swarm.Particles(k).PBEST.X = currentX;
         Swarm.Particles(k).PBEST.O = Swarm.Particles(k).O;
      end

      % Update the GBEST
      if Swarm.Particles(k).O < Swarm.GBEST.O
         Swarm.GBEST.X = currentX;
         Swarm.GBEST.O = Swarm.Particles(k).O;
      end
   end

   % Update the X and V vectors
   w = wMax - t .* ((wMax - wMin) / maxIter);

   for k = 1 : noP
      Swarm.Particles(k).V = w .* Swarm.Particles(k).V + c1 .* rand(1,nVar) .*
(Swarm.Particles(k).PBEST.X - Swarm.Particles(k).X) ...
                                                + c2 .* rand(1,nVar) .*
(Swarm.GBEST.X - Swarm.Particles(k).X);


      % Check velocities
```

```
        index1 = find(Swarm.Particles(k).V > vMax);
        index2 = find(Swarm.Particles(k).V < vMin);

        Swarm.Particles(k).V(index1) = vMax(index1);
        Swarm.Particles(k).V(index2) = vMin(index2);

        Swarm.Particles(k).X = Swarm.Particles(k).X + Swarm.Particles(k).V;

        % Check positions
        index1 = find(Swarm.Particles(k).X > ub);
        index2 = find(Swarm.Particles(k).X < lb);

        Swarm.Particles(k).X(index1) = ub(index1);
        Swarm.Particles(k).X(index2) = lb(index2);

    end

    outmsg = ['Iteration# ', num2str(t) , ' Swarm.GBEST.O = ' ,
num2str(Swarm.GBEST.O)];
    disp(outmsg);

    cgCurve(t) = Swarm.GBEST.O;
end

semilogy(cgCurve);
xlabel('Iteration#')
ylabel('Weight')
```

Iteration 50

```
clear all
close all
clc
```

```
% Define the details of the table design problem
nVar = 3;               % number of variables
ub = [1000 1000 1000]; %upper Bound
lb = [0 0 0];           % lower bound
fobj = @tunning;        % Objective function Name

% Define the PSO's paramters
noP = 15;               % number of particles for initialization
maxIter = 20;           % maximum iterations
wMax = 0.5;
wMin = 0.1;
c1 = 2;
c2 = 2;
vMax = (ub - lb) .* 0.2;
vMin  = -vMax;



% The PSO algorithm

% Initialize the particles
for k = 1 : noP
   Swarm.Particles(k).X = (ub-lb) .* rand(1,nVar) + lb;
   Swarm.Particles(k).V = zeros(1, nVar);
   Swarm.Particles(k).PBEST.X = zeros(1,nVar);
   Swarm.Particles(k).PBEST.O = inf;

   Swarm.GBEST.X = zeros(1,nVar);
   Swarm.GBEST.O = inf;
end



% Main loop
for t = 1 : maxIter
```

```
    % Calcualte the objective value
    for k = 1 : noP
        currentX = Swarm.Particles(k).X;
        Swarm.Particles(k).O = fobj(currentX);

        % Update the PBEST
        if Swarm.Particles(k).O < Swarm.Particles(k).PBEST.O
            Swarm.Particles(k).PBEST.X = currentX;
            Swarm.Particles(k).PBEST.O = Swarm.Particles(k).O;
        end

        % Update the GBEST
        if Swarm.Particles(k).O < Swarm.GBEST.O
            Swarm.GBEST.X = currentX;
            Swarm.GBEST.O = Swarm.Particles(k).O;
        end
    end

    % Update the X and V vectors
    w = wMax - t .* ((wMax - wMin) / maxIter);

    for k = 1 : noP
        Swarm.Particles(k).V = w .* Swarm.Particles(k).V + c1 .* rand(1,nVar) .*
(Swarm.Particles(k).PBEST.X - Swarm.Particles(k).X) ...
                                                    + c2 .* rand(1,nVar) .*
(Swarm.GBEST.X - Swarm.Particles(k).X);


        % Check velocities
        index1 = find(Swarm.Particles(k).V > vMax);
        index2 = find(Swarm.Particles(k).V < vMin);

        Swarm.Particles(k).V(index1) = vMax(index1);
        Swarm.Particles(k).V(index2) = vMin(index2);
```

```
        Swarm.Particles(k).X = Swarm.Particles(k).X + Swarm.Particles(k).V;

        % Check positions
        index1 = find(Swarm.Particles(k).X > ub);
        index2 = find(Swarm.Particles(k).X < lb);

        Swarm.Particles(k).X(index1) = ub(index1);
        Swarm.Particles(k).X(index2) = lb(index2);

    end

    outmsg = ['Iteration# ', num2str(t) , ' Swarm.GBEST.O = ' ,
num2str(Swarm.GBEST.O)];
    disp(outmsg);

    cgCurve(t) = Swarm.GBEST.O;
end

semilogy(cgCurve);
xlabel('Iteration#')
ylabel('Weight')
```

Iteration 100

```
clear all
close all
clc

% Define the details of the table design problem
nVar = 3;               % number of variables
ub = [1000 1000 1000]; %upper Bound
lb = [0 0 0];           % lower bound
fobj = @tunning;        % Objective function Name
```

```
% Define the PSO's paramters
noP = 15;                % number of particles for initialization
maxIter = 20;            % maximum iterations
wMax = 0.5;
wMin = 0.1;
c1 = 2;
c2 = 2;
vMax = (ub - lb) .* 0.2;
vMin  = -vMax;

% The PSO algorithm
% Initialize the particles
for k = 1 : noP
   Swarm.Particles(k).X = (ub-lb) .* rand(1,nVar) + lb;
   Swarm.Particles(k).V = zeros(1, nVar);
   Swarm.Particles(k).PBEST.X = zeros(1,nVar);
   Swarm.Particles(k).PBEST.O = inf;
      Swarm.GBEST.X = zeros(1,nVar);
   Swarm.GBEST.O = inf;
end

% Main loop
for t = 1 : maxIter
   % Calcualte the objective value
   for k = 1 : noP
      currentX = Swarm.Particles(k).X;
      Swarm.Particles(k).O = fobj(currentX);
      % Update the PBEST
      if Swarm.Particles(k).O < Swarm.Particles(k).PBEST.O
         Swarm.Particles(k).PBEST.X = currentX;
         Swarm.Particles(k).PBEST.O = Swarm.Particles(k).O;
      end

      % Update the GBEST
```

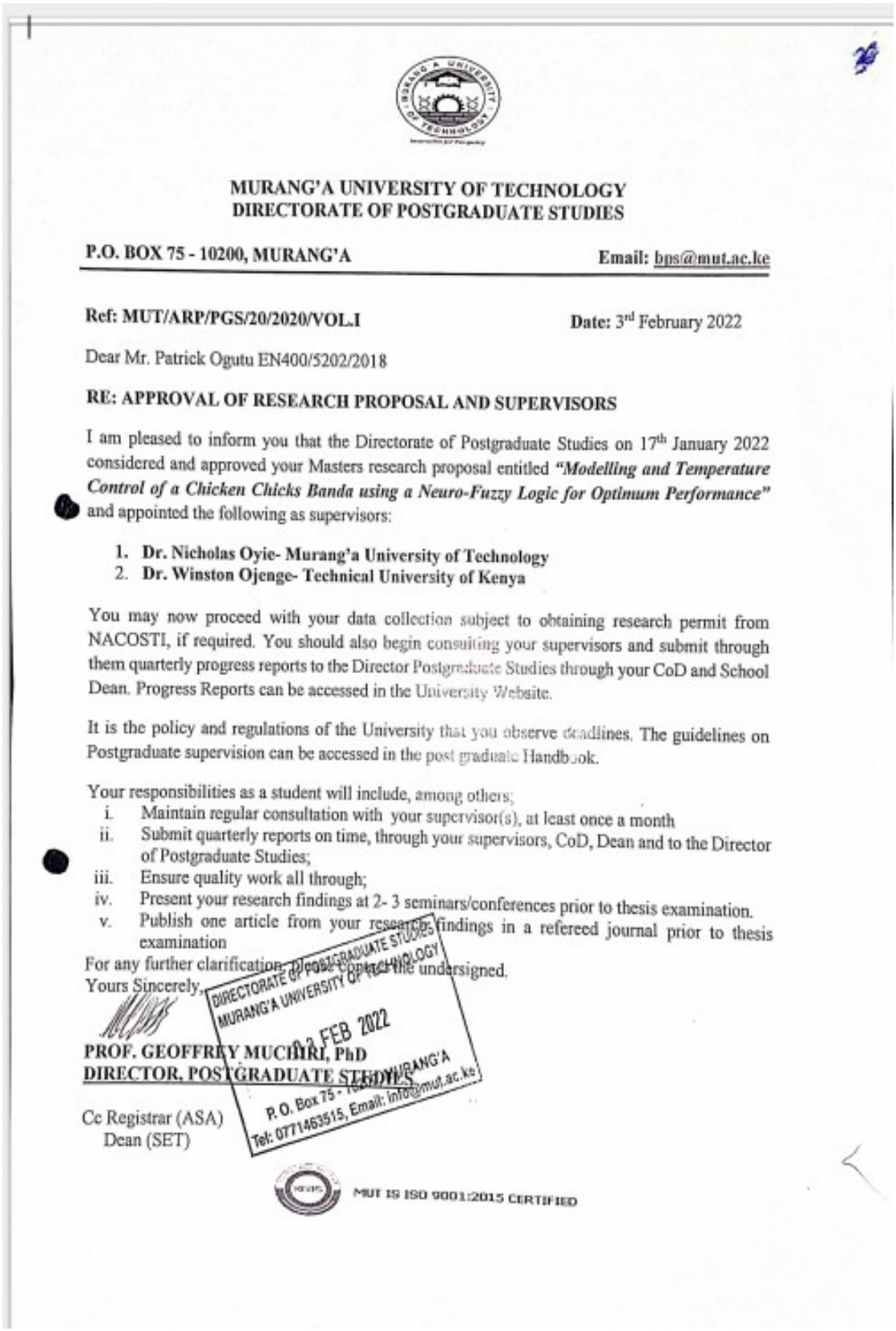
```
        if Swarm.Particles(k).O < Swarm.GBEST.O
            Swarm.GBEST.X = currentX;
            Swarm.GBEST.O = Swarm.Particles(k).O;
        end
    end

    % Update the X and V vectors
    w = wMax - t .* ((wMax - wMin) / maxIter);

    for k = 1 : noP
        Swarm.Particles(k).V = w .* Swarm.Particles(k).V + c1 .* rand(1,nVar) .*
(Swarm.Particles(k).PBEST.X - Swarm.Particles(k).X) ...
                                                    + c2 .* rand(1,nVar) .*
(Swarm.GBEST.X - Swarm.Particles(k).X);
```

Appendix 2. Authorization

The thesis was authorized as per the following letter:



MURANG'A UNIVERSITY OF TECHNOLOGY
DIRECTORATE OF POSTGRADUATE STUDIES

P.O. BOX 75 - 10200, MURANG'A Email: bps@mut.ac.ke

Ref: MUT/ARP/PGS/20/2020/VOL.I Date: 3rd February 2022

Dear Mr. Patrick Ogutu EN400/5202/2018

RE: APPROVAL OF RESEARCH PROPOSAL AND SUPERVISORS

I am pleased to inform you that the Directorate of Postgraduate Studies on 17th January 2022 considered and approved your Masters research proposal entitled ***"Modelling and Temperature Control of a Chicken Chicks Banda using a Neuro-Fuzzy Logic for Optimum Performance"*** and appointed the following as supervisors:

1. **Dr. Nicholas Oyie- Murang'a University of Technology**
2. **Dr. Winston Ojenge- Technical University of Kenya**

You may now proceed with your data collection subject to obtaining research permit from NACOSTI, if required. You should also begin consulting your supervisors and submit through them quarterly progress reports to the Director Postgraduate Studies through your CoD and School Dean. Progress Reports can be accessed in the University Website.

It is the policy and regulations of the University that you observe deadlines. The guidelines on Postgraduate supervision can be accessed in the post graduate Handbook.

Your responsibilities as a student will include, among others;

i. Maintain regular consultation with your supervisor(s), at least once a month
ii. Submit quarterly reports on time, through your supervisors, CoD, Dean and to the Director of Postgraduate Studies;
iii. Ensure quality work all through;
iv. Present your research findings at 2- 3 seminars/conferences prior to thesis examination.
v. Publish one article from your research findings in a refereed journal prior to thesis examination

For any further clarification, please contact the undersigned.

Yours Sincerely,

DIRECTORATE OF POSTGRADUATE STUDIES
MURANG'A UNIVERSITY OF TECHNOLOGY
03 FEB 2022
P.O. Box 75 - 10200, MURANG'A
Tel: 0771463515, Email: info@mut.ac.ke

PROF. GEOFFREY MUCHIRI, PhD
DIRECTOR, POSTGRADUATE STUDIES

Cc Registrar (ASA)
Dean (SET)

MUT IS ISO 9001:2015 CERTIFIED

Appendix 2. Publications

The thesis has produced the following publications:

1. P. Ogutu et al "Humidity control scheme for Chicken Banda using PSO," *International Journal of Engineering and Advanced Technology,* vol. 1, no. 3, pp. 101-104, 2022.

2. P. Ogutu et al , "Performance Analysis of an improved particle swarm optimization and the standard particle swarm optimization," *International Journal of Engineering and Advanced Technology,* vol. 13, no. 1, pp. 37-42, 2023