

# **OPTIMIZATION OF PID CONTROLLERS OF CASCADED SYSTEMS USING HYBRID METAHEURISTIC ALGORITHM MODEL**

**Thesis Submitted  
in Partial Fulfillment of the Requirements  
for the Degree of**

**MASTER OF TECHNOLOGY**

**in**

**Control & Instrumentation**

**by**

**Kumar Ujjwal**

**(2k23/C&I/04)**

**Under the Supervision of**

**Prof. Mini Sreejeth**

**Assistant Prof. Anupama**



**To the**

**Department of Electrical Engineering  
DELHI TECHNOLOGICAL UNIVERSITY  
(Formerly Delhi College of Engineering)  
Shahbad Daulatpur, Main Bawana Road, Delhi-11042, India**

**May, 2025**

## **ACKNOWLEDGEMENTS**

I am highly grateful to the Department of Electrical Engineering, Delhi Technological University (DTU) for providing this opportunity to carry out this project work. The constant guidance and encouragement received from my supervisors Prof. Mini Sreejeth and Assistant Prof. Anupama have been of great help in carrying out my project work. I also extend my sincere thankfulness to all the faculty members and the entire staff of Research Laboratory, Electrical Engineering Department, D.T.U for their continuous support and motivation. Finally, I would like to express gratefulness to my family and friends for having confidence in me which encouraged me to pursue Master of Technology at an advanced stage of my academic career.

Place: **New Delhi**

Date: May 2025

**Kumar Ujjwal**

(2K23/C&I/04)

M.Tech. (Control & Instrumentation)

Delhi Technological University

# **DELHI TECHNOLOGICAL UNIVERSITY**

(Formerly Delhi College of Engineering)  
Shahbad Daultpur, Main Bawana Road, Delhi-11042

## **CANDIDATE'S DECLARATION**

I, **Kumar Ujjwal**, Roll No. **2K23/C&I/04**, M.Tech (Control & Instrumentation), hereby declare that the work which is being presented in the thesis entitled “**optimization of pid controllers of cascaded systems using hybrid metaheuristic algorithm model**” which is submitted by me to the **Department of Electrical Engineering**, Delhi Technological University, Delhi in partial fulfillment of the requirement for the award of the degree of Master of Technology, is an authentic record of my own work carried out under the supervision of **Prof. Mini Sreejeth** and **Assistant Prof. Anupama**.

**Candidate's Signature**

This is to certify that the student has incorporated all the corrections suggested by the examiners in the thesis and the statement made by the candidate is correct to the best of our knowledge.

**Signature of Supervisor (s)**

**Signature of External Examiner**

# **DELHI TECHNOLOGICAL UNIVERSITY**

(Formerly Delhi College of Engineering)  
Shahbad Daultapur, Main Bawana Road, Delhi-11042

## **CERTIFICATE**

We, hereby certify that the Project Dissertation titled “optimization of pid controllers of cascaded systems using hybrid metaheuristic algorithm model” which is submitted by Kumar Ujjwal, Roll No. 2K23/C&I/04, Department of Electrical Engineering, Delhi Technological University, Delhi in partial fulfillment of the requirement for the award of the degree of Master of Technology is a testimony of the project work carried out by the student under our supervision. To the best of our awareness this work has not been submitted in part or full for any Degree or Diploma to this University or to a different place.

Place: **New Delhi**

Date: May 2025

**Dr. Mini Sreejeth**

(Professor, DTU)

**Ms. Anupama**

(Assistant Professor, DTU)

## **ABSTRACT**

This thesis presents a novel methodology for optimizing a Cascaded Proportional-Integral-Derivative (PID) controller for a flow-level control system using a hybrid metaheuristic algorithm approach. The proposed model integrates the Genetic Algorithm (GA) and Ant Lion Optimizer (ALO) combining the strengths of both the standalone algorithms. System used is flow-level control system which is a Cascaded system, known for its effectiveness in improving the disturbance rejection capabilities and dynamic response of control systems. In this work, the inner loop regulates the faster flow dynamics, while the outer loop addresses the slower level dynamics. The hybrid GA-ALO algorithm is designed to capitalize on the global search capabilities of GA and the exploitation efficiency of ALO, thus overcoming limitations like premature convergence and slow optimization often associated with single algorithms. The PID parameters are optimized through a weighted objective function considering rise time, settling time, overshoot, and integral absolute error (IAE). This research not only provides an effective control strategy for flow-level systems but also highlights the potential of hybrid metaheuristic algorithms in complex control optimization problems.

## **TABLE OF CONTENTS**

<b>ACKNOWLEDGEMENT</b>	<b>i</b>
<b>CANDIDATE DECLARATION</b>	<b>ii</b>
<b>CERTIFICATE</b>	<b>iii</b>
<b>ABSTRACT</b>	<b>iv</b>
<b>TABLE OF CONTENTS</b>	<b>v</b>
<b>LIST OF FIGURES</b>	<b>vii</b>
<b>LIST OF TABLES</b>	<b>viii</b>
<b>LIST OF ABBREVIATIONS</b>	<b>ix</b>
<b>LIST OF SYMBOLS</b>	<b>x</b>
<b>CHAPTER 1 INTRODUCTION</b>	<b>1</b>
1.1. Introduction	1
1.2. Literature Review	1
1.3. Related Work	2
1.3.1. Traditional Methods	2
1.3.2. Metaheuristic Algorithms	3
1.4. Summary of Work	4
1.5. Outline of Dissertation	5
1.6. Conclusion	6
<b>CHAPTER 2 METHODOLOGY</b>	<b>7</b>
2.1. Introduction	7
2.2. Optimization Techniques	7
2.2.1. Genetic Algorithm	8
2.2.2. Ant Lion Optimizer	10
2.2.3. Hybrid Genetic Algorithm – Ant Lion Optimizer	13
2.3. Performance Metrics	16
2.4. Conclusion	17
<b>CHAPTER 3 SYSTEM DESCRIPTION</b>	<b>18</b>
3.1. Introduction	18
3.2. System Design	18

3.2.1. Flow Process Loop	20
3.2.2. Level Process Loop	21
3.2.3. Hardware Setup	21
3.2.4. Mathematical Modeling	22
3.3. System Dynamics	23
3.4. Optimization Algorithm Parameters	24
3.5. Conclusion	26
<b>CHAPTER 4 RESULTS AND DISCUSSION</b>	<b>27</b>
4.1. Introduction	27
4.2. Simulation Setup	27
4.3. Performance Analysis	27
4.3.1. Rise Time	30
4.3.2. Settling Time	30
4.3.3. Maximum Overshoot	31
4.3.4. Integral of Absolute Error	31
4.4. Disturbances Handling	32
4.5. Optimized PID Parameters	32
4.6. Hardware Output Analysis	33
4.7. Conclusion	35
<b>CHAPTER 5 CONCLUSION AND FUTURE SCOPE</b>	<b>36</b>
5.1. Conclusion	36
5.2. Future Scope	36
<b>REFERENCES</b>	<b>38</b>
<b>APPENDIX I</b>	<b>42</b>
<b>LIST OF PUBLICATIONS</b>	<b>56</b>

## LIST OF FIGURES

<b>Figure No.</b>	<b>Description</b>	<b>Page No.</b>
<b>2.1</b>	Flowchart of G.A Algorithm	<b>10</b>
<b>2.2</b>	Flowchart of A.L.O Algorithm	<b>12</b>
<b>2.3</b>	Flowchart of Hybrid G.A - A.L.O Algorithm	<b>15</b>
<b>3.1</b>	Block Diagram of Cascaded Flow–Level System	<b>20</b>
<b>3.2</b>	Experimental Architecture of Hardware System	<b>22</b>
<b>4.1</b>	System Response with Optimized PID Gains using G.A	<b>28</b>
<b>4.2</b>	System Response with Optimized PID Gains using A.L.O	<b>29</b>
<b>4.3</b>	System Response with Optimized PID Gains using Hybrid G.A – A.L.O	<b>29</b>
<b>4.4</b>	Comparison of System Response using G.A, A.L.O and Hybrid G.A – A.L.O	<b>30</b>
<b>4.5</b>	System Response with Delayed Disturbance at $t = 40$ seconds	<b>32</b>
<b>4.6</b>	Hardware System Response of G.A – A.L.O based PID controller	<b>34</b>



## **LIST OF TABLES**

<b>Table No.</b>	<b>Description</b>	<b>Page No.</b>
<b>2.1</b>	Difference Between I.S.E, I.T.A.E, and I.A.E	<b>17</b>
<b>3.1</b>	Parameter Values of Flow-Level Control System	<b>24</b>
<b>3.2</b>	Parameter Values of G.A Algorithm	<b>25</b>
<b>3.3</b>	Parameter Values of A.L.O Algorithm	<b>25</b>
<b>4.1</b>	Performance Analysis of G.A, A.L.O, and G.A-A.L.O Algorithms	<b>28</b>

## **LIST OF ABBREVIATIONS**

<b>GA</b>	Genetic Algorithm
<b>ALO</b>	Ant Lion Optimizer
<b>GA-ALO</b>	Hybrid Genetic Algorithm-Ant Lion Optimizer
<b>PID</b>	Proportional-Integral-Derivative (Controller)
<b>IAE</b>	Integral of Absolute Error
<b>ISE</b>	Integral of Squared Error
<b>ITAE</b>	Integral of Time Absolute Error
<b>Tr</b>	Rise Time
<b>Ts</b>	Settling Time
<b>Mp</b>	Maximum Overshoot
<b>Ess</b>	Steady-State Error

## LIST OF SYMBOLS

$G_f$	Transfer function of the flow process
$A_f$	Flow process gain
$\tau_f$	Flow process time constant
$T_f$	Flow process delay
$G_l$	Transfer function of the level process
$A_l$	Level process gain
$\tau_l$	Level process time constant
$G_F$	Flow sensor transfer function
$A_F$	Flow sensor gain
$\tau_F$	Flow sensor time constant
$G_L$	Level sensor transfer function
$A_L$	Level sensor gain
$\tau_L$	Level sensor time constant
$G_r$	ON/OFF controller transfer function
$K_r$	ON/OFF controller gain
$G_{PID}$	PID controller transfer function
$K_p$	Proportional gain
$K_i$	Integral gain
$K_d$	Derivative gain
$T_{inner}$	Inner-loop (flow control) transfer function
$T_{outer}$	Outer-loop (level control) transfer function

# CHAPTER 1

## INTRODUCTION

### 1.1 Introduction

This chapter introduces the importance of flow-level control in modern control industries like HVAC, chemical plants, etc. It will give a brief overview of the PID controllers for effective control in these industries. A detailed literature review is presented covering the traditional tuning methods like Cohen-Coon and Ziegler-Nichols outlining its limitations towards dealing with the complex and non-linear systems. Afterwards, the chapters explores the emergence and innovation of metaheuristic algorithms such Genetic Algorithm, Particle Swarm, Grey Wolf Optimizer, etc. which offer robust and adaptive tuning of the PID controller. A hybrid model of GA-ALO is also discussed for the PID controller. Finally, the structure of the dissertation is presented, providing a roadmap for the subsequent chapters on methodology, system modeling, simulation results, and practical implementation.

### 1.2 Literature Review

Industrial processes depend fundamentally on precise flow-level rate management. Manufacturing facilities alongside petroleum operations and water purification sectors together with Heating Ventilation and Air Conditioning systems require exact flow-level rate controls for maintaining operational smoothness and industry efficiency. Chemical facilities attain correct reaction results through the utilization of precise flow control to combine various reactants which assists in reducing operational danger from adverse events [1]. Thermal comfort performance and levels of energy efficiency are decided through the installation of flow-level rate controls in HVAC systems [2]. Flow-level rate control systems assist water treatment facilities to allocate resources in a fair manner and enhance energy efficiency utilization in operations [3]. Having the required tank levels and flow rates is critical in order to attain operational efficiency, safety, product quality, and energy conservation [4].

Finding proper flow-level rate control is a challenging problem primarily due to the complexities of the system. There are nonlinear characteristics present in these systems preventing the response prediction using simple linear mathematical models. Disturbances outside the system from input supply variations and environmental factors along with varying levels of demand have significant

effects on system performance. System delays from sensors to actuators with measurement of response time and actuator's response time reduce traditional control methods' effectiveness when it comes to maintaining desired performance. Handling of these complications requires feedback control systems that integrate the Proportional-Integral-Derivative (PID) controller as their most well-established and efficient solution [5]. The PID controller operates by adjusting the system's input based on its three variable parameters. The proportional term detects current errors, the derivative term forecasts future error trends to minimize oscillations and improve system's stability, and the integral term eliminates steady-state discrepancies. Control components synchronize to achieve precise and adaptive system control under challenging conditions.

However, the effectiveness of PID controllers heavily depends on the tuning of their parameters ( $K_p$ ,  $K_i$ ,  $K_d$ ). Poorly tuned controllers can result in sluggish response, high overshoot, instability, or poor disturbance rejection. Traditional PID tuning methods such as Ziegler-Nichols [6] and Cohen-Coon [7] provided simple heuristic techniques based on process reaction curves but often fail to optimally tune complex nonlinear or time-delay systems.

Recent research highlights the adoption of metaheuristic algorithms — nature-inspired optimization techniques — to solve complex control problems. Techniques like Genetic Algorithm (GA) [8], Ant Lion Optimizer (ALO) [9], Particle Swarm Optimization (PSO) [10], and Grey Wolf Optimizer (GWO) [11] have proven effective in tuning PID controllers by efficiently searching large, complex solution spaces where conventional methods are inadequate. Hybridization of algorithms, combining the strengths of two or more metaheuristic methods, is increasingly recognized as a powerful approach. This has motivated the current research toward developing a hybrid GA-ALO model for cascade PID control optimization.

## **1.3 Related Work**

### **1.3.1 Traditional Methods**

Flow – level control systems within modern control industries like chemical processing, HVAC and water treatment industries highly depend on PID controllers because of their simplicity and reliable capabilities to manage flow and level rates to maintain different pressures and temperatures [12]. With the use of traditional PID tuning methods, such as Cohen-Coon and Ziegler-Nichols, have been widely adopted to address these challenges. The Cohen-Coon method is particularly effective for systems with significant time delays, a common characteristic in flow-level control processes [13]. By analyzing the process reaction curve generated from a step input, this method estimates key parameters

such as time constant, dead time, and process gain. These estimates serve as the basis for calculating initial PID gains, providing a reliable starting point for systems with delays, such as those found in chemical flow regulation.

The Ziegler-Nichols approach, however, is usually used on systems with oscillatory responses [14]. It consists of two forms: the open-loop technique, which calculates the ultimate gain and time constant from the process reaction curve, and the closed-loop technique, which consists of tuning the system until there are sustained oscillations and then determining the ultimate gain and oscillation period. This technique has found successful application in water treatment and chemical processing, where stability and low oscillations are essential for efficient flow-level control.

Though both Cohen-Coon and Ziegler-Nichols methods are used extensively, they are empirically approximated and might not provide the best PID parameters for processes with intricate dynamics, e.g., nonlinear or highly changing processes. In a bid to overcome such limitations, sophisticated tuning techniques have been developed that combine the conventional methods with evolutionary optimization methods such as Particle Swarm Optimization (P.S.O), Genetic Algorithms (G.A), and Grey Wolf Optimizer (G.W.O).

### **1.3.2 Metaheuristic Algorithms**

With increasing complexity in industrial processes, the limitation of conventional PID tuning approaches has become more conspicuous [15]. Old methods such as Ziegler-Nichols and Cohen-Coon, though significant in the past, rely on empirical guidelines and plain linear models. Under conditions with load variability and external disturbances, there is a need for more advanced and adaptive techniques in order to perform optimal PID tuning. Thus, metaheuristic algorithms have emerged to tune PID controllers [16].

One of the first metaheuristic methods ever used to tune a PID is the Genetic Algorithm (GA) [17], based on the natural selection and evolution principles. GA works by evolving a population of candidate PID parameter sets using selection, crossover, and mutation. Research has demonstrated that PID controllers tuned with GA perform better than those tuned with classical techniques in terms of lower overshoot, faster settling times, and improved disturbance rejection.

After GA, other metaheuristic algorithms like Particle Swarm Optimization (PSO) [18], Ant Colony Optimization (ACO) [19], and Simulated Annealing (SA) [20] were also implemented successfully for PID tuning. Recent advances had resulted in newer metaheuristics like the Ant Lion Optimizer (ALO) [21], Grey Wolf Optimizer (GWO) [22], and Whale Optimization Algorithm (WOA) [23], each of which introduced distinct search abilities. These algorithms provided

better-balanced trade-offs between exploration (sampling new areas) and exploitation (improving known good solutions), which are essential for efficient and robust PID tuning.

The wide use of metaheuristics in PID tuning can be attributed to several factors:

- **Flexibility:** They do not require prior knowledge about the system model.
- **Global Search Capability:** They can escape local minima and find globally optimal parameters.
- **Robustness:** They perform consistently across a variety of systems, including nonlinear, time-delayed, and uncertain systems.
- **Multi-Objective Optimization:** Some metaheuristics can optimize multiple performance criteria simultaneously (e.g., minimizing rise time, settling time, and overshoot together).

Moreover, the trend of **hybrid metaheuristic algorithms** combines the strengths of two or more algorithms. For instance, combining GA's global search capabilities with ALO's fast convergence has led to superior PID tuning outcomes which is better than the individual algorithms.

## 1.4 Summary of Work

This research work introduces a novel hybrid optimization model for effective tuning of cascade PID controllers using Genetic Algorithm (GA) and Ant Lion Optimizer (ALO). Following is the summary of the work done for optimization of PID controller:

- A MATLAB simulation model was developed to closely replicate a real-world flow-level system. It incorporates actuator dynamics (pump and valve behavior), process delays, and external disturbances. The model ensures realistic evaluation of the control automation system.
- Independent applications of Genetic Algorithm (GA) and Ant Lion Optimizer (ALO) were used to optimize PID parameters. Performance of individual GA and ALO methods was compared on the basis of important parameters such as Integral of Absolute Error (IAE), settling time, and overshoot. Output of the both the algorithms assists in determining the baseline of the strengths and limitations of the individual algorithms.
- The hybrid model combines the global search strength of GA with the fast local convergence of ALO.

- The tuned **controllers** were evaluated based on dynamic response metrics such as Integral of Absolute Error, Settling Time, Maximum Overshoot, and Disturbance rejection.

In summary, the hybrid GA-ALO model constitutes an effective and handy solution to PID tuning in sophisticated industrial processes. Through the synergy of the strengths of two well-known metaheuristic algorithms, this contribution provides improved control performance, robustness, and adaptability, which significantly contributes to the development of intelligent control systems in modern automation industries.

## 1.5 Outline of Dissertation

This dissertation is systematically structured into five main chapters, each chapter describing the objectives, methodology, results, and future directions of the research.

- **Chapter 1 (Introduction):**

This chapter presents the relevance of flow-level control systems in industrial processes, addressing their difficulties and the necessity for improved PID controller tuning techniques. A comprehensive literature review is given, including both conventional PID tuning techniques like Ziegler-Nichols and Cohen-Coon, and the advancement of metaheuristic algorithms.

- **Chapter 2 (Methodology):**

The chapter addresses the optimization methods used in tuning the cascaded systems' PID controllers. It describes the operation principles of the Genetic Algorithm (GA) and the Ant Lion Optimizer (ALO), with their respective strengths. Subsequently, writing on the design and implementation of the hybrid GA-ALO model. Additionally, the chapter explains the choice of performance indicators, with the focus placed on using Integral of Absolute Error (IAE) [24] instead of its alternatives ISE [25] and ITAE [26] for the purpose of robust performance evaluation.

- **Chapter 3 (System Description):**

This chapter explains the flow-level control system, modeled in detail such as the flow process loop and the level process loop. It derives the necessary transfer functions for the processes, sensors, and PID controllers, and discusses system dynamics, including considerations of process delays and external disturbances. The complete cascade control system model is formulated, forming the basis for simulation and optimization studies.



- **Chapter 4 (Results and Discussion):**

This chapter provides the simulation setup and a detailed analysis of the results obtained through GA, ALO, and Hybrid GA-ALO optimization methods. Comparative studies are conducted based on rise time, settling time, maximum overshoot, and IAE performance metrics. Additionally, hardware output results are discussed, validating the simulation results and demonstrating the effectiveness of the proposed hybrid model under real-world conditions.

- **Chapter 5 (Conclusions and Future Work):**

The final chapter concludes the study by summarizing key findings. It highlights the practical contributions of the research to modern control industrial applications and suggests future directions.

## **1.6 Conclusion**

In conclusion this chapter has briefly reviewed both the traditional and new metaheuristic algorithm methods for PID tuning. It highlighted the limitations of traditional tuning techniques like Ziegler-Nichols and Cohen Coon methods when applied to complex and non-linear systems. Whereas, the evolution of metaheuristic algorithms have many advantages in terms of fast and robust tuning of the PID controller. The chapter introduced the hybrid GA-ALO model as a powerful optimization approach, combining GA's global search strength with ALO's local refinement capabilities.

## **CHAPTER 2**

### **METHODOLOGY**

#### **2.1 Introduction**

This chapter focuses on advanced optimization techniques for tuning of cascade PID controllers. It begins by discussing the limitations of traditional tuning methods in handling complex control systems especially those with non linear behavior. To address these challenges, the chapter introduces three metaheuristic algorithms - Genetic Algorithm (GA), Ant Lion Optimizer (ALO), and a hybrid GA-ALO algorithm. Each algorithm is described in detail, including its operation and flowchart representation. The chapter also discusses the principal time-domain performance measures used to analyze controller performance, including rise time, settling time, percent overshoot, and steady-state error. Particular consideration is provided to the Integral of Absolute Error (IAE) as the performance index to be optimized. A clear understanding of the hybrid metaheuristic algorithm will be established by the end of this chapter.

#### **2.2 Optimization Techniques**

Optimization methods are crucial to PID controller parameter fine-tuning to provide optimal dynamic performance in control systems. Tuning the PID parameters in complicated systems like cascaded control structures with process delays and external disturbances are particularly challenging. Classical techniques, such as manual tuning or basic analytical methods, tend to be incapable of fulfilling the performance levels for such systems because they cannot handle effectively the nonlinearities, time delays, and external disturbances characteristic of real-world problems.

Under such circumstances, advanced optimization methods have come to the forefront due to their capacity for exploring broad solution spaces and adaptively determining the optimum set of PID parameters. These include metaheuristic algorithms that imitate biological processes. This research utilizes three such algorithms, including Genetic Algorithm (GA), Ant Lion Optimizer (ALO), and a hybrid GA-ALO, to optimize the PID parameters of cascade control systems.

The Genetic Algorithm (GA) is one of the most well-known evolutionary optimization methods that mimics natural selection. Through operations like selection, crossover, and mutation, GA searches through a large population of potential solutions and evolves toward the optimal set of PID parameters. With its

capability to solve complex, nonlinear optimization problems, GA can be applied to systems with numerous interacting components and disturbances [27].

However, the Ant Lion Optimizer (ALO) is a swarm metaheuristic that draws inspiration from ant lion hunting behavior. ALO performs well in searching the problem's solution space by balancing exploration with exploitation, thus being good at escaping local minima and providing a global search for optimal solutions. Such a property is particularly useful in tuning PID controllers for systems with time delays and changing external conditions [28].

The hybrid GA-ALO method uses the strengths of both algorithms and benefits from the global search capabilities of GA and the efficient local search mechanisms of ALO. With the combination of the two methods, the hybrid scheme is intended to leverage the strengths of both algorithms to produce quicker convergence and improved solutions for optimizing the cascade PID controllers. By implementing such metaheuristic algorithms in a cascade PID control system, this research aims to illustrate how higher-level optimization methods offer better performance than traditional approaches. The outcomes emphasize their promise in enhancing the stability, transient behavior, and robustness of control systems, especially in difficult real-world applications that include delays and disturbances. Each optimization techniques are outlined shortly below:

### **2.2.1 Genetic Algorithm (GA)**

Genetic Algorithm (GA) is a search and optimization method which is commonly used to solve complex problems where the traditional approach may find it difficult. GAs is based in Darwin's survival of the fittest theory such that more superior solutions are provided with greater opportunities to generate offspring in the subsequent generation.

The normal operation of a GA, as illustrated by the flowchart, consists of several significant steps, each playing a role in seeking the optimum solution:

#### **1. Begin**

The algorithm starts by initializing all necessary parameters such as population size, crossover rate, mutation rate, and termination conditions.

#### **2. Initial Population**

The first step after initialization is generating the initial population. This population consists of a set of randomly created individuals, where diversity of the initial population is crucial for effective exploration of the search space.

### 3. Calculate the Fitness Value

Each individual is evaluated using a fitness function designed specifically for the problem. The fitness function measures how "good" an individual is compared to others. Higher fitness values indicate better solutions.

### 4. Selection

In the selection phase, individuals are chosen to reproduce based on their fitness values. There are various selection strategies, such as roulette wheel selection, tournament selection, or rank selection. Generally, individuals with higher fitness have a greater chance of being selected, ensuring that good traits are carried forward.

### 5. Crossover

After selection, the crossover operation is performed. This process combines the genetic information of two parent individuals to produce one or more offspring. The goal is to create new individuals that inherit the strengths of their parents, potentially leading to better solutions. Crossover can occur in several ways, such as single-point, two-point, or uniform crossover.

### 6. Mutation

To maintain genetic diversity within the population and avoid premature convergence, mutation is applied. Mutation introduces random small changes to individuals, which helps the algorithm explore new parts of the search space. Typically, mutation occurs with a low probability to balance exploration and exploitation.

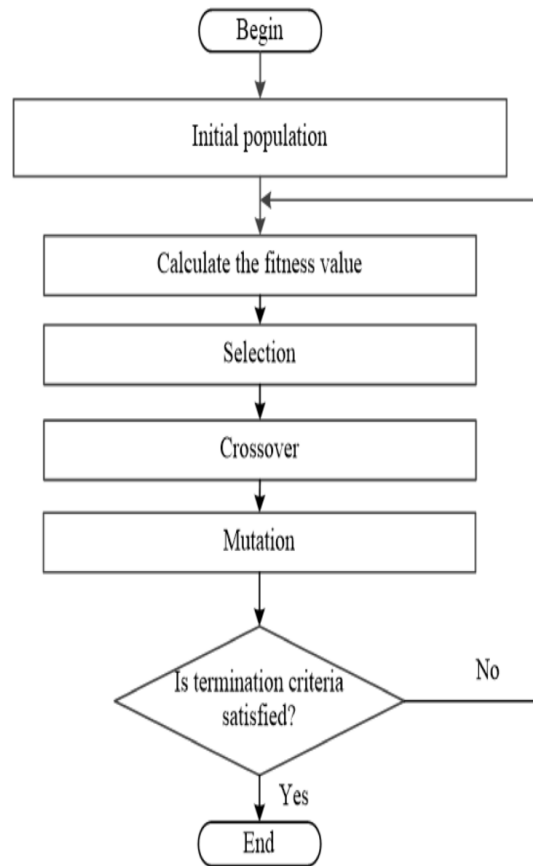
### 7. Is Termination Criteria Satisfied?

After mutation, the algorithm checks if the termination condition is met. Termination criteria can include reaching a maximum number of generations, achieving a predefined fitness value, or observing no significant improvement over a number of generations.

- If the termination condition is **satisfied**, the algorithm proceeds to the end.
- If **not satisfied**, the cycle repeats, starting again from calculating fitness values for the new generation.

### 8. End

Once the termination condition is met, the algorithm stops. The best individual solution found during the process is the optimal value for the system.



**Figure 2.1:** Flowchart of G.A Algorithm

Figure 2.1 illustrates the flowchart of GA Algorithm. In brief, Genetic Algorithms are a strong and versatile approach to optimizing problems by simulating nature's evolutionary processes. Because they can solve complex, multimodal, and high-dimensional search spaces, they are highly sought after in numerous real-world applications.

### 2.2.2 Ant Lion Optimizer (ALO)

The Ant Lion Optimizer (ALO) is a population-based metaheuristic algorithm motivated by natural ant lion's hunting mechanism. Ants are used to symbolize candidate solutions searching the solution space within ALO, whereas ant lions serve as navigators that lead ants towards improved solutions. The algorithm optimizes exploration and exploitation, thus becoming much efficient for solving intricate optimization problems in fields ranging from engineering design, machine learning, operational research etc.

The working process of ALO, as depicted in the flowchart, follows a structured series of steps:

## 1. **Start**

The algorithm initializes all necessary parameters, such as the size of the ant and ant lion populations, the maximum number of iterations, and the boundaries of the search space.

## 2. **Initialize Population of Ants and Ant lions**

At the beginning, two separate populations are randomly created: one for ants (candidate solutions) and another for ant lions (potential traps or guides). Each individual's position within the search space is randomly assigned within specified limits.

## 3. **Evaluate Fitness of Ants and Ant lions**

Each ant and ant lion's fitness is evaluated using a predefined objective function. The fitness value measures the quality of a solution, determining how close it is to the optimal answer. High-performing solutions are crucial for guiding the search process.

## 4. **For Each Ant**

Several operations are performed for each ant during every iteration:

- **Select an Ant lion:** An ant lion is selected based on a roulette wheel selection strategy, which gives higher probability to better ant lions, ensuring strong solutions influence the ants.
- **Update Ant Position:** Each ant simulates a random walk influenced by the selected ant lion, representing exploration within the search space.
- **Apply Boundary Checks:** After movement, boundary checks are applied to ensure ants remain within the valid limits of the search space.
- **Evaluate New Ant Fitness:** The fitness of each updated ant is re-evaluated to measure improvement.

## 5. **Update Ant lions (Replace if Ants are Better)**

If an ant discovers a solution better than its corresponding ant lion, it replaces the ant lion's position. This strategy ensures that the ant lions continuously improve over time, promoting the evolution of better solutions.

## 6. **Decrease Search Radius (Adaptive Random Walk)**

As the algorithm proceeds, the randomness in ant movement is gradually decreased. This converging of the search radius helps shift the algorithm from broad exploration in early stages to fine exploitation near promising solutions later.

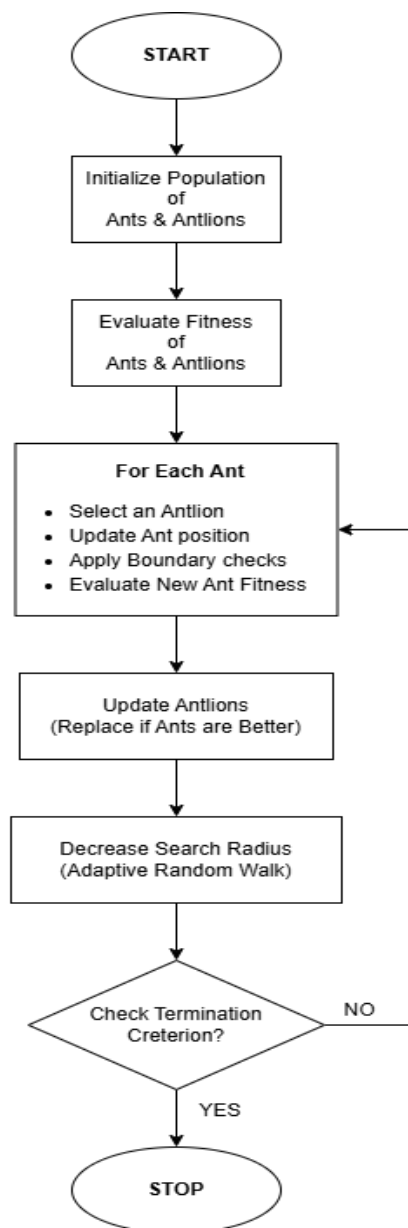
## 7. **Check Termination Criterion**

Common criteria include reaching a maximum number of iterations or achieving an acceptable fitness value.

- If the termination condition is **not satisfied**, the process loops back, continuing the optimization.
- If **satisfied**, the algorithm proceeds to the stop phase.

#### 8. Stop

Once the termination condition is met, the algorithm stops and it produces the best solutions from the search space.



**Figure 2.2:** Flowchart of A.L.O Algorithm

Figure 2.2 illustrates the ALO algorithm flowchart. In brief, Ant Lion Optimizer (ALO) is a compact metaheuristic algorithm that mimics the hunting behavior of ant lions. ALO achieves a balance between exploration and exploitation, enabling it to explore far in the solution space as well as to fine-tune solutions precisely. The simplicity of ALO, as well as its robustness and excellent performance, makes it a suitable method to solve engineering, machine learning, and other complex optimization problems, thus an efficient optimization tool.

### 2.2.3 Hybrid Genetic Algorithm – Ant Lion Optimizer (GA-ALO)

Genetic Algorithms (GA) and Ant Lion Optimizers (ALO) both have powerful optimization problem-solving capability, but with a set of limitations as well. GA works very well with global exploration and diversity preservation in the search space but is sluggish when converging close to the optimal solution. Conversely, ALO is optimal in exploring local search spaces for fine-tuning solutions but is very likely to be trapped in local minima if the initial population is weak and not diverse.

In order to overcome the above individual limitations, the hybrid GA-ALO model is presented. The hybrid process, as shown in the flowchart, consists of the following principal steps:

1. **Start:**

The algorithm starts by setting the initial parameters, such as the PID controller bounds, GA population size, and fitness evaluation settings.

2. **Initialize PID Bounds and GA Population**

An initial population for the Genetic Algorithm is randomly generated within defined PID parameter bounds. This population represents a diverse set of candidate solutions for the problem at hand.

3. **Evaluate Fitness of GA Individuals**

Each individual in the GA population is evaluated using a predefined fitness function. The fitness value reflects how well the individual solves the optimization problem.

4. **Apply Genetic Algorithm**

Standard GA operations are performed:

- **Selection:** High-performing individuals are selected based on their fitness.
- **Crossover:** Selected individuals are paired, and their genetic material is recombined to produce new offspring.



- **Mutation:** Small random changes are introduced in the offspring to maintain genetic diversity.

#### 5. **Sort Population by Fitness**

After GA operations, the population is sorted based on fitness values, ensuring the best solutions are easily identified.

#### 6. **Use GA Offspring as ALO Ants**

The offspring generated by GA are then used as the initial population of ants for the ALO phase. This step ensures that ALO starts with a high-quality and diverse set of solutions.

#### 7. **Initialize ALO Ant lions (Best GA Individuals)**

The best individuals from the GA phase are selected to act as ant lions in the ALO phase. These act as strong guides for the ants during the local search.

#### 8. **Apply ALO Algorithm**

ALO operations are carried out, focusing on exploitation:

- **ALO Random Walk:** Ants perform a random walk influenced by the ant lions.
- **Fitness Evaluation:** Updated positions are evaluated for fitness.
- **Update Elite Ant lion:** The best solution (elite) is tracked and updated.
- **Replace Worst if Improved:** If a newly found solution is better, it replaces the worst-performing ant lion.

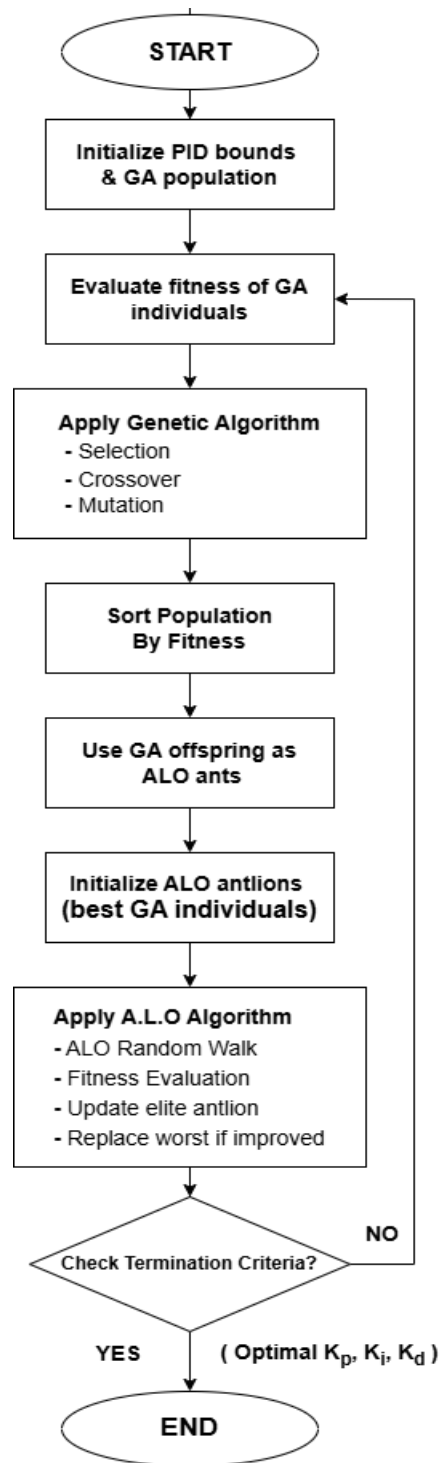
#### 9. **Check Termination Criteria**

The algorithm checks if the stopping conditions are met, such as reaching the maximum number of iterations or achieving a desired fitness level.

- If the termination criteria are not met, the process continues.
- If met, the algorithm proceeds to the final step.

#### 10. **End**

The algorithm concludes by returning the best solution found through the combined exploration and exploitation processes.



**Figure 2.3:** Flowchart of Hybrid G.A – A.L.O Algorithm

Figure 2.3 shows the flowchart of hybrid GA-ALO algorithm. The hybrid GA-ALO model offers a comprehensive and effective optimization strategy by combining the strengths of both Genetic Algorithm (GA) and Ant Lion Optimizer (ALO). GA is responsible for providing broad exploration of the search space,

ensuring that a wide range of potential solutions. While, ALO takes over to intensively exploit the best solutions through the search space. This combination leads to faster convergence toward optimal solutions and achieves higher-quality results compared to using GA or ALO individually. The hybrid model thus provides a balanced process for optimization of PID controller.

## 2.3 Performance Metrics

In order to analyze the performance of a controller and to direct the process of optimization, some of the most important time-domain performance measures are generally taken into consideration. These measures assist in defining the performance of the control system in terms of efficiency, effectiveness, and stability in meeting the output specifications. The principal measures are Rise Time ( $T_r$ ), Settling Time ( $T_s$ ), Maximum Overshoot ( $M_p$ ), Steady-State Error ( $E_{ss}$ ), and the Integral of Absolute Error (IAE). Each of these, in combination, gives a complete view of system behavior and is crucial for validation that the control design satisfies performance requirements.

**Rise Time ( $T_r$ )** is the time taken for the system response to change from 10% to 90% of the final steady-state value.

**Settling Time ( $T_s$ )** is the time period in which the output of the system gets stabilized to a specified limit (normally within 2% or 5%) of the final value without oscillation beyond that limit.

**Maximum Overshoot ( $M_p$ )** is the amount by which the system response overshoots the desired final value, as a percentage of that value. Steady-State Error ( $E_{ss}$ ) is the difference between the output required and the output obtained when the system settles.

**Steady-State Error ( $E_{ss}$ )** measures the difference between the desired output and the actual output after the system has settled.

Considering all these metrics, it allows for a balanced controller design that emphasizes a fast response, minimal overshoot, stability, and robustness to disturbances. Optimization based on these performance metrics is essential for achieving high-performance control systems in practical applications.

While choosing cost function for minimization of the error, it is important to know which performance index will work best with the given system. Integral performance indices are crucial for objective evaluation of control system performance. Three popular indices are shown in Table 2.1.

**Table 2.1:** Difference between I.S.E, I.T.A.E, and I.A.E

Metric	Definition	Characteristics
ISE (Integral of Squared Error)	$ISE = \int_0^T e^2(t)dt$	Penalizes larger errors more heavily due to the squaring.
ITAE (Integral of Time-weighted Absolute Error)	$ITAE = \int_0^T t  e(t) dt$	Penalizes errors that persist longer, encouraging faster settling.
IAE (Integral of Absolute Error)	$IAE = \int_0^T  e(t) dt$	Measures the total absolute error over time.

In this study, IAE is selected because:

- IAE penalizes all errors equally, avoiding excessive sensitivity to large initial transients, as seen with ISE, or prolonged minor errors, as emphasized by ITAE. This characteristic ensures consistent control performance across a wide range of operating conditions, making it ideal for real-world systems.
- IAE is computationally simple to implement, which is beneficial for real-time applications.
- IAE also promotes smoother control actions, avoiding the aggressive responses that might result from minimizing ISE, and preventing noise amplification.

In the event of a sudden disturbance, minimizing IAE leads to a steady correction of the error without overreacting. Unlike ISE, might cause excessive control effort, or ITAE, which could overly prioritize late-stage errors.

## 2.5 Conclusion

This chapter outlined the methodology for optimizing cascade PID controllers using Genetic Algorithm (GA), Ant Lion Optimizer (ALO), and their hybridization. The combined approach leverages the global search strength of GA and the local exploitation ability of ALO, ensuring robust and efficient controller tuning. Controller performance is evaluated based on key dynamic metrics, including rise time, settling time, overshoot, and steady-state error, with a particular focus on minimizing the Integral of Absolute Error (IAE). The choice of IAE provides balanced penalization across errors, promoting stable, fast, and reliable system responses. The hybrid optimization strategy developed ensures adaptability and robustness under varying operating conditions and disturbances.

## CHAPTER 3

### SYSTEM DESCRIPTION

#### 3.1 Introduction

This chapter describes the design and application of a cascade control system for liquid level control in a process tank. The system employs a master-slave strategy with an inner loop controlling the flow rate and an outer loop controlling the tank level. This configuration provides faster response time and disturbance rejection than single-loop control. The chapter encompasses system design, block diagrams, and hardware configuration with sensors, pumps, and valves. First-order transfer functions are utilized to model flow and level processes mathematically to represent system dynamics. Three optimization techniques - Genetic Algorithm (GA), Ant Lion Optimizer (ALO), and hybrid GA-ALO for the tuning of PID controllers are proposed. Their parameters and MATLAB programming are explained for effective controller design. In general, this chapter gives an overall framework for modeling, regulation, and optimization of a cascaded flow-level control system, applicable to industrial process control problems.

#### 3.2 System Design

Cascade systems are a type of control systems in which two or more loops are used for improving performance of control systems. In this main loop or outer loop controls the main process of the system whereas the secondary loop or inner loop is used for controlling intermediate variable of the system. Figure 3.1 shows the block diagram of the hardware system used for the research and below are the descriptions of each and every block of the hardware system.

1. **ON-OFF Controller ( $G_r$ ):**

The level error is first processed by an ON-OFF controller, represented by block  $G_r$ . This component provides a basic level of control, often used for switching actions such as enabling or disabling the main PID loop or providing an initial control action.

2. **PID Controller ( $G_s$ ):**

The signal from the ON-OFF controller is then fed into the **PID controller ( $G_s$ )**. This controller calculates a continuous control signal based on **proportional (P)**, **integral (I)**, and **derivative (D)** terms, ensuring smoother and more accurate control compared to the ON-OFF method. The PID controller processes deviations and generates a flow rate setpoint to correct the error.

### 3. Flow Process ( $G_f$ ):

The output of the PID controller acts as a command signal for the **flow process**, represented by  $G_f$ . This block simulates the dynamics of the fluid flow system—such as the behavior of pumps, valves, and pipes—which directly affect the inflow to the tank.

### 4. Disturbance Input:

An external **disturbance** enters the system after the flow process block. This could represent environmental changes, valve position shifts, or pressure fluctuations. The disturbance impacts the final output and must be counteracted by the control loops.

### 5. Level Process ( $G_l$ ):

The combined result of the flow input and disturbances is passed to the **level process block ( $G_l$ )**. This block models the tank level dynamics, where the flow influences the rise or fall of the liquid level. The output of this block is the **actual tank level**, which is the final controlled variable.

### Feedback Mechanisms

To maintain control, the system incorporates two critical feedback loops:

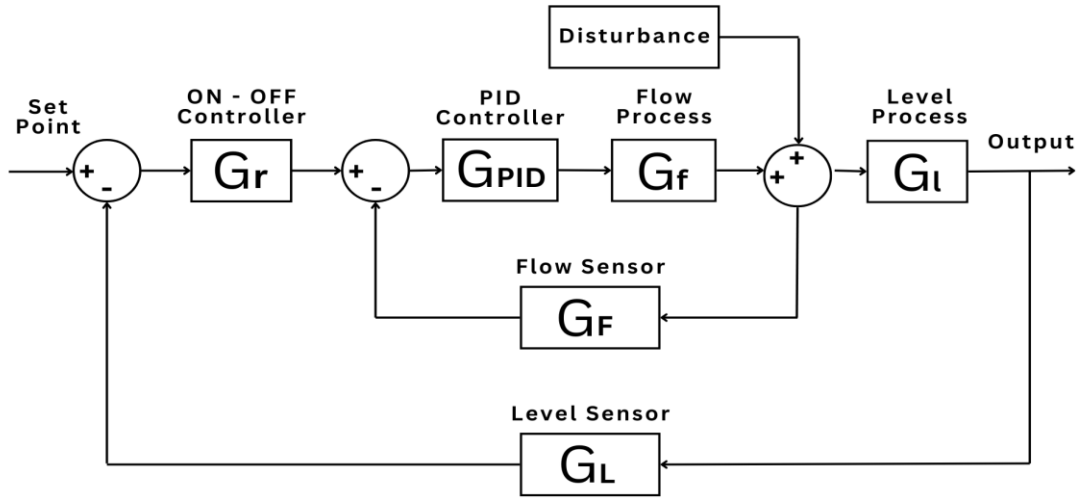
- **Flow Sensor Feedback ( $G_F$ ):**

A **flow sensor** measures the actual flow rate and provides feedback to the PID controller. This helps the controller compare the desired and actual flow rates, allowing for rapid correction of discrepancies.

- **Level Sensor Feedback ( $G_L$ ):**

A **level sensor** measures the tank level and sends it back to the comparator at the input. This enables the continuous calculation of the level error and ensures that the system adjusts in real time to maintain the setpoint.

In summary, this cascade control block diagram shows how an **inner flow control loop** supports the **outer level control loop**, with the combination of sensors, controllers, and process models working together to achieve precise and stable regulation of a two-variable system. Figure 4 shows the block diagram of a cascaded flow-level control system.



**Figure 3.1:** Block Diagram of Cascaded Flow- Level System

### 3.2.1 Flow Process Loop

In industrial automation, flow control involves regulating the flow rate of a fluid (liquid or gas) through a pipeline or system. In the current study, the flow process loop is designed as the inner loop of the cascade control structure.

The objective of the inner flow control loop is to ensure that the desired flow set-point, provided by the outer level controller, is accurately achieved by controlling the speed of a pump. The flow process dynamics are typically first-order with a time delay due to actuator dynamics, piping resistance, and flow sensor response time. The flow process is represented by the following first-order transfer function with delay:

$$G_f(s) = \frac{A_f}{\tau_f s + 1} \cdot e^{-t_f s} \quad (3.1)$$

where:

- $A_f$  = Flow process gain
- $\tau_f$  = Flow process time constant
- $t_f$  = Flow process delay

These parameters define how the flow rate responds to changes in the pump speed.

### 3.2.2 Level Process Loop

The level control system forms the outer loop of the cascade structure. Its purpose is to maintain the liquid level within a process tank at a desired set-point, despite variations in inflow or outflow. The level process is inherently slower compared to the flow process, as tank levels change gradually in response to flow variations. The level dynamics are also modeled as a first-order system:

$$G_1(s) = \frac{A_l}{\tau_l s + 1} \quad (3.2)$$

where:

- $A_l$  = Level process gain
- $\tau_l$  = Level process time constant

Thus, the level process is much slower and more inertial compared to the fast-responding flow process.

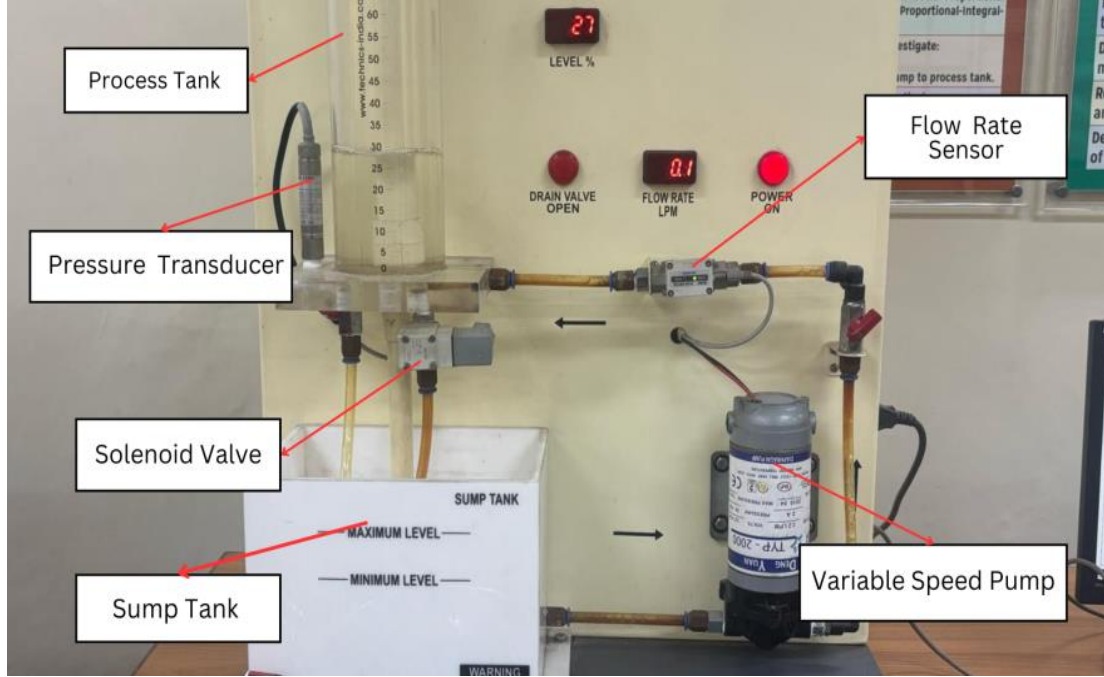
### 3.2.3 Hardware Setup

The hardware setup system used for flow-level control is a cascaded control system which is used to regulate the liquid level in a process tank by controlling the inflow rate. This system uses an inner cascade PID control strategy, where the inner loop governs the flow rate and the outer loop maintains the level set-point. The system works as the master-slave loop working where the master loop is the outer loop that is the level control loop and the slave loop is the inner loop that is the flow control of the system. This arrangement ensures faster response and improved disturbance rejection compared to single-loop control systems. The outer loop is responsible for keeping the liquid level at the desired value. A pressure transducer installed around the process tank continuously measures the actual level of the tank. This measurement is compared with the user's set-point, generating a level error between the set-point and the measured output. Figure 3.2 shows the hardware setup of flow-level cascaded control system.

The controller produces an error and determines the optimal flow rate needed to correct the level divergence which becomes the set-point for the inner flow control loop. In the inner loop, a flow rate sensor measures the actual inflow. The inner-loop PID controller compares this measurement with the desired flow rate set by the outer loop. Based on the flow error, it adjusts the speed of the pump using a control signal. The variable speed pump then regulates the water supply to the tank accordingly. The system includes a proportional solenoid drain valve to manage controlled outflow and manual drain valves for maintenance purposes.



The sump tank collects drained fluid, completing the circulation. This cascade control setup effectively handles dynamic changes and disturbances, making it suitable for studying real-time control applications in industries like water treatment, process control, and chemical plants. Figure 3.2 shows the real world flow-level control hardware system used for the research.



**Figure 3.2:** Experimental Architecture of Hardware System

### 3.2.4 Mathematical Modeling

Accurate mathematical modeling is critical for designing and optimizing control systems. The flow-level cascaded control system can be described through a series of transfer functions for each component. Flow and Level process transfer functions are already mentioned above. Rests of the system component's transfer function are mentioned below:

#### 1. Flow Sensor Transfer Function ( $G_F$ ):

The Flow Sensor's transfer function is modeled as a first-order system with a gain  $A_F$  and a time constant  $\tau_F$ . It is expressed as:

$$G_F(s) = \frac{A_F}{\tau_F s + 1} \quad (3.3)$$

#### 2. Level Sensor Transfer Function ( $G_L$ ):

The Level Sensor's transfer function is modeled as a first-order system with a gain  $A_L$  and a time constant  $\tau_L$ . It is expressed as:

$$G_L(s) = \frac{A_L}{\tau_L s + 1} \quad (3.4)$$

### 3. ON/OFF Controller Transfer Function ( $G_r$ ):

ON/OFF controller representing the controller's gain with  $u(t)$  representing step function showing switching behavior. It is expressed as:

$$G_r(s) = K_r * u(t) \quad (3.5)$$

### 4. PID Controller Transfer Function:

The PID controller's equation is expressed as:

$$G_{PID} = K_p + \frac{K_i}{s} + K_d s \quad (3.6)$$

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

## 3.3 System Dynamics

### 1. Inner-Loop (Flow Control) Transfer Function ( $T_{inner}$ ):

The inner flow control loop is the product of the PID controller and the flow process with flow sensor in negative feedback to both of them. This is expressed as:

$$T_{inner} = \frac{G_{pid} * G_f}{1 + G_F * G_{pid} * G_f} \quad (3.7)$$

### 2. Outer-Loop (Level Control) Transfer Function:

The disturbance transfer function  $G_d$  is summed up with the open-loop system  $G_{open}$ . The closed-loop transfer function is computed by taking the feedback of the open-loop system and disturbance and it is expressed as:

$$T_{outer} = \frac{G_r * T_{inner} * G_l}{1 + G_r * T_{inner} * G_l * G_L} \quad (3.8)$$

As the process's time constant is generally very high as compared to sensor's time constant ( $\tau_{sensor} \ll \tau_{process}$ ). Therefore,  $G_F = G_L = 1$  is reasonable for most systems because sensors are faster than processes. Now, Equation 3.7 and 3.8 can be expressed respectively as:

$$T_{inner} = \frac{G_{pid} * G_f}{1 + G_{pid} * G_f} \quad (3.9)$$

$$T_{outer} = \frac{G_r * T_{inner} * G_l}{1 + G_r * T_{inner} * G_l} \quad (3.10)$$

Table 3.1 summarizes the value of parameters used in the real system. The overall system becomes a cascade of two feedback loops, enabling faster correction of disturbances.

**Table 3.1:** Parameter Values of Flow-Level Control System

S. No.	Description	Parameter	Value(units)
1	Flow Process Gain	$A_f$	50
2	Flow Process Time Delay	$t_f$	1 second
3	Flow Process Time Constant	$\tau_f$	30 seconds
4	Level Process Gain	$A_l$	0.13
5	Level Process Time Constant	$\tau_l$	3 seconds
6	On/Off Controller Gain	$K_r$	5

### 3.4 Optimization Algorithm Parameters

Three optimization strategies were implemented to tune the PID controllers:

- Genetic Algorithm (GA)
- Ant Lion Optimizer (ALO)
- Hybrid GA-ALO

The parameters for each algorithm were carefully selected based on trial-and-error and literature review. Table 3.2 and Table 3.3 summarize the value of parameters used in the Genetic Algorithm Code & Ant Lion Optimizer respectively.

**Table 3.2:** Parameter Values of G.A Algorithm

S.No.	Parameter	Value
1	Size of Population	25
2	No. of Generations	100
3	Elitism Factor	1
4	Mutation	0.4
5	Crossover	0.8
6	Search Range	0 - 100

**Table 3.3:** Parameter Values of A.L.O Algorithm

S.No.	Parameter	Value
1	Maximum Iterations	100
2	Population Size	25
3	Random Walk Ratio	Adaptive
4	Elitism Factor	1
5	Search Range	0 - 100

MATLAB Coding which provides MATLAB implementations of advanced optimization algorithms. **Genetic Algorithm Code** includes MATLAB scripts for selection, crossover, and mutation processes used to evolve solutions. **Ant Lion Optimizer Code** features MATLAB functions that simulate the hunting mechanism of ant lions through random walks and adaptive search. **GA-ALO Hybrid Algorithm Code** integrates both GA and ALO techniques in MATLAB, using GA for global exploration and ALO for local exploitation. These MATLAB codes (refer to Appendix I) are structured for modularity, allowing easy customization and application to various optimization problems in engineering problems.

### **3.5 Conclusion**

This chapter presented a detailed summary of the cascade control system designed to regulate the liquid level and flow in a process tank using a master-slave PID control structure. The inner loop focused on controlling the flow rate, while the outer loop managed the tank level. Mathematical modeling of each component was discussed through first-order transfer functions, capturing the system's dynamic behavior accurately. Optimization algorithms, including Genetic Algorithm (GA), Ant Lion Optimizer (ALO), and a hybrid GA-ALO, were introduced to fine-tune the PID parameters for the flow control of the cascaded system. These algorithms were implemented in MATLAB, enabling precise controller tuning for improved system stability and accuracy.

## **CHAPTER 4**

### **RESULTS AND DISCUSSION**

#### **4.1 Introduction**

This chapter presents the performance analysis of the hardware setup using the three optimization techniques: Genetic Algorithm (GA), Ant Lion Optimizer (ALO), and a Hybrid GA-ALO method. A MATLAB Simulink model was developed to evaluate system performance. Key performance metrics such as rise time, settling time, maximum overshoot, and Integral of Absolute Error (IAE) are analyzed for each optimization algorithms. Further, a hardware implementation using an HMI-enabled setup demonstrates the real-time efficiency and robustness of the optimized controllers. This chapter highlights the comparative performance and effectiveness of the hybrid optimization technique in improving system response and reliability.

#### **4.2 Simulation Setup**

To validate the performance of the cascade PID controllers tuned by different optimization techniques, a MATLAB simulation model was developed.

**The system configuration includes:**

- Inner Loop: Flow control loop with flow process transfer function along with PID controller.
- Outer Loop: Level control loop with level process transfer function along with ON-OFF controller.
- Disturbances: External disturbances are applied to test robustness.
- Time Delay Handling: Flow process delay handled using Pade approximation.

**Software Tools:** MATLAB 2023a (Simulink)

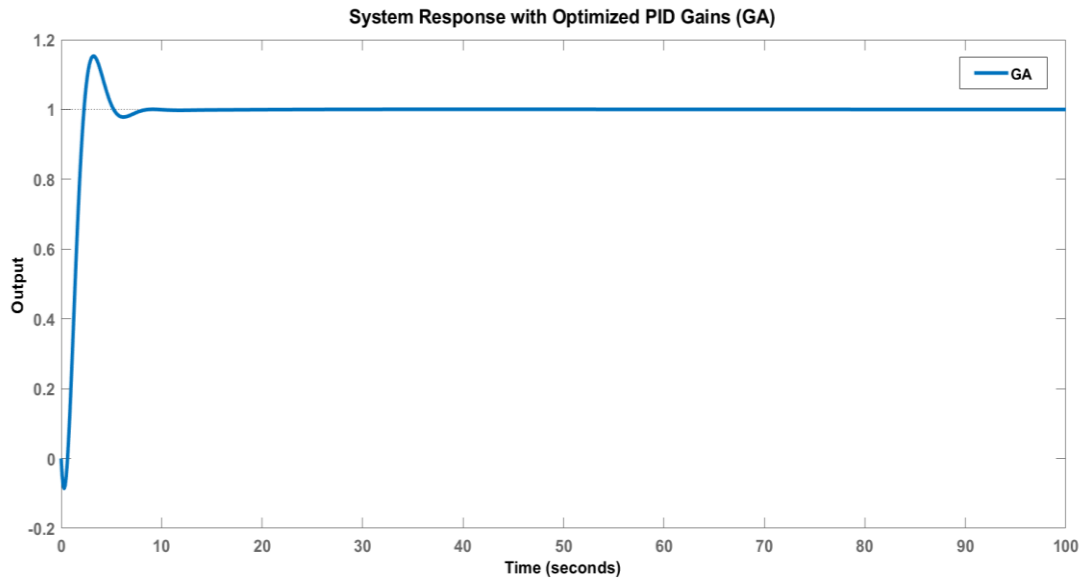
#### **4.3 Performance Analysis**

The cascade PID control system was simulated using three different optimization strategies: Genetic Algorithm (GA), Ant Lion Optimizer (ALO), and the Hybrid GA-ALO model. Table V shows the performance analysis of G.A, A.L.O, and G.A-A.L.O. The performance of the system was evaluated based on key dynamic response parameters including rise time, settling time, maximum overshoot, and Integral of Absolute Error (IAE). Table 4.1 summarizes the values obtained from the unit step system's response by using G.A, G.W.O, and G.A-G.W.O hybrid optimization algorithms to tune PID controller.

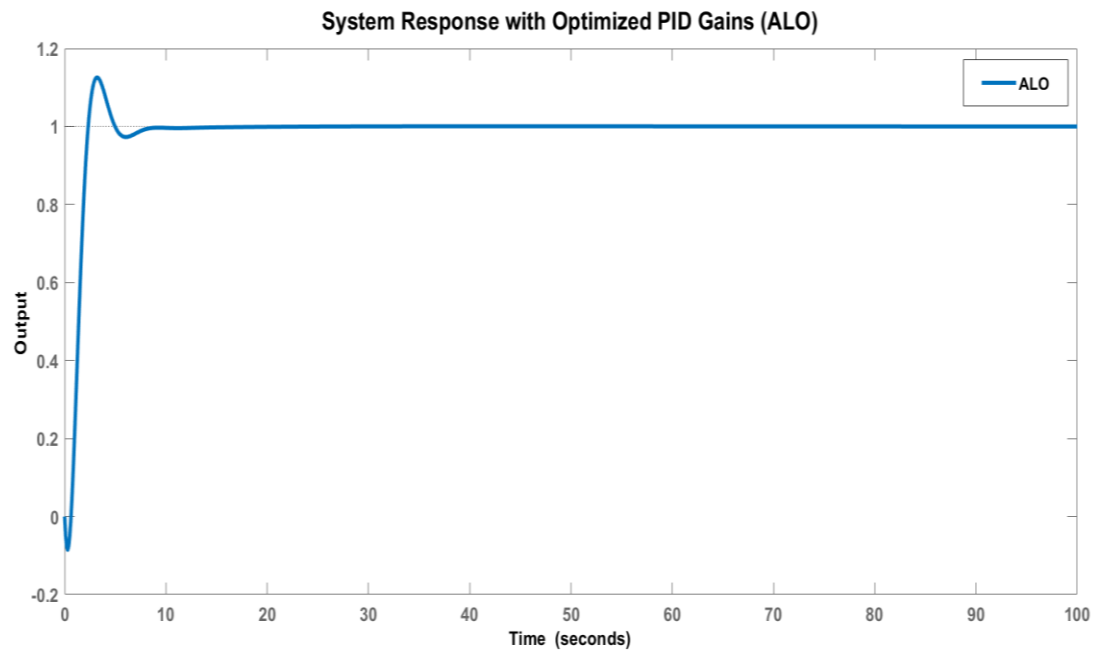
**Table 4.1:** Performance Analysis of G.A, A.L.O, And G.A-A.L.O

S.No.	Parameter	G.A	A.L.O	G.A – A.L.O
1	Rise Time (s)	1.66	1.25	1.32
2	Settling Time (s)	7.14	6.84	4.75
3	Maximum Overshoot	1.13	1.14	1.07
4	Best Function f(x)	2.023	1.827	1.201

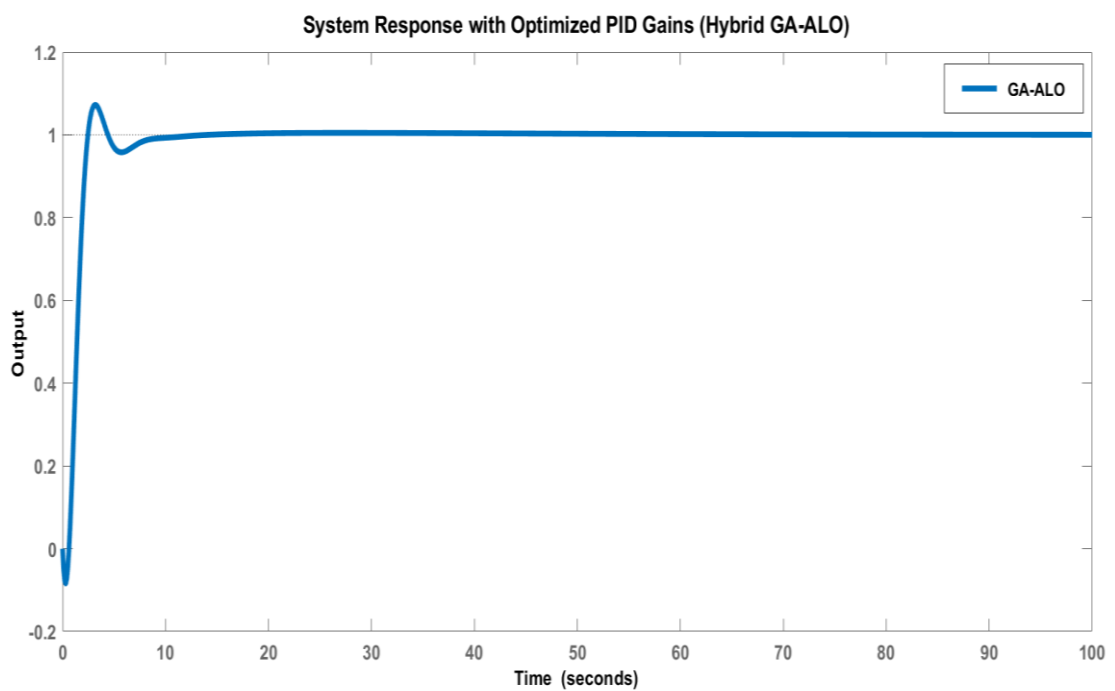
Figure 4.1, 4.2, and 4.3 shows the system response with optimized PID gains using G.A, A.L.O, and hybrid G.A-A.L.O respectively.



**Figure 4.1:** System Response with Optimized PID Gains using G.A



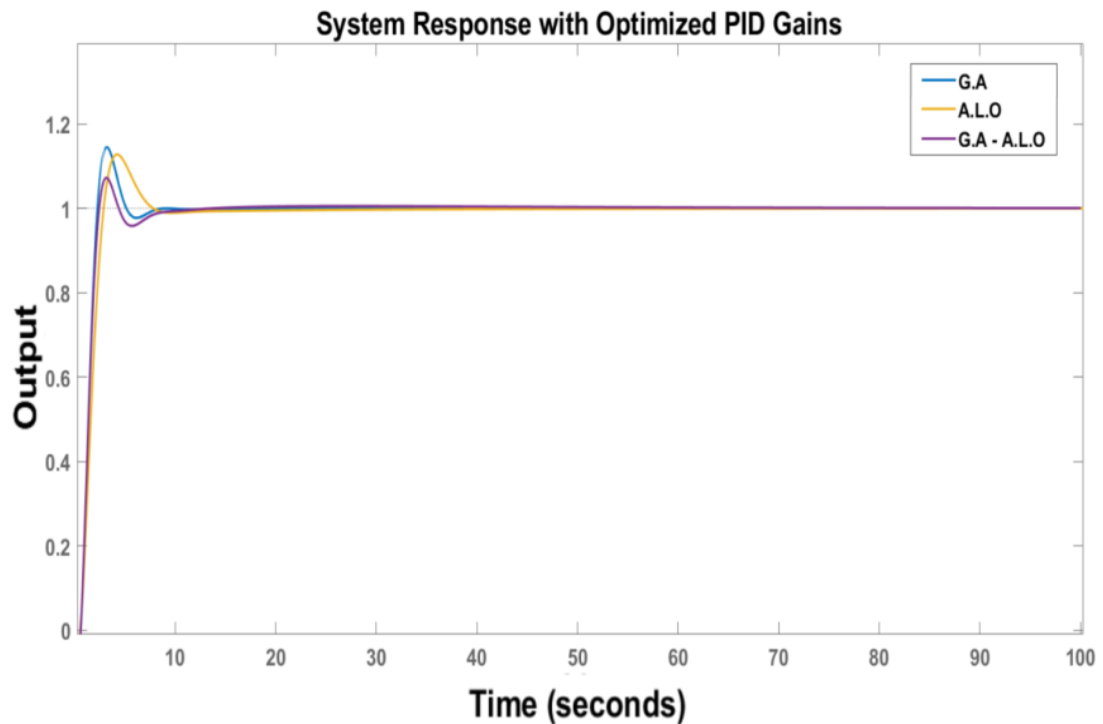
**Figure 4.2:** System Response with Optimized PID Gains using A.L.O



**Figure 4.3:** System Response with Optimized PID Gains using Hybrid G.A – A.L.O

Figure 9 shows the comparison of system response with optimized PID gains using G.A, A.L.O, and G.A – A.L.O.





**Figure 4.4:** Comparison of System Response with Optimized PID Gains using G.A, A.L.O, and G.A - A.L.O

#### 4.3.1 Rise Time

Rise Time is the time taken for the system output to rise from 10% to 90% of the desired final value.

##### Observations:

- GA achieved a rise time of 1.66 seconds.
- ALO provided a slightly faster rise at 1.25 seconds.
- The Hybrid GA-ALO method resulted in a rise time of 1.32 seconds.

Although ALO showed the fastest rise, the Hybrid GA-ALO achieved a more balanced rise time with better stability, preventing aggressive responses that could cause overshoot.

#### 4.3.2 Settling Time

Settling Time is the duration required for the system to settle within a specified tolerance band (typically  $\pm 2\%$ ) around the final value.

**Observations:**

- GA tuned system settled in 7.14 seconds.
- ALO tuned system settled in 6.84 seconds.
- The Hybrid GA-ALO method resulted in the shortest settling time of 4.75 seconds.

The hybrid approach significantly reduced the settling time by approximately 33% compared to GA alone, enabling faster stabilization and quicker response to changes.

**4.3.3 Maximum Overshoot**

Maximum Overshoot measures how much the system output exceeds the final desired value, typically expressed as a percentage.

**Observations:**

- GA-based system showed a 13% overshoot.
- ALO-based system showed a slightly higher 14% overshoot.
- The Hybrid GA-ALO based system achieved only 7% overshoot.

Lower overshoot achieved by the hybrid technique ensures better system safety and avoids unnecessary actuator stress, making it more suitable for sensitive industrial applications.

**4.3.4 Integral of Absolute Error (IAE)**

Integral of Absolute Error (IAE) quantifies the total accumulated absolute error over time, serving as the primary performance index in this study.

**Observations:**

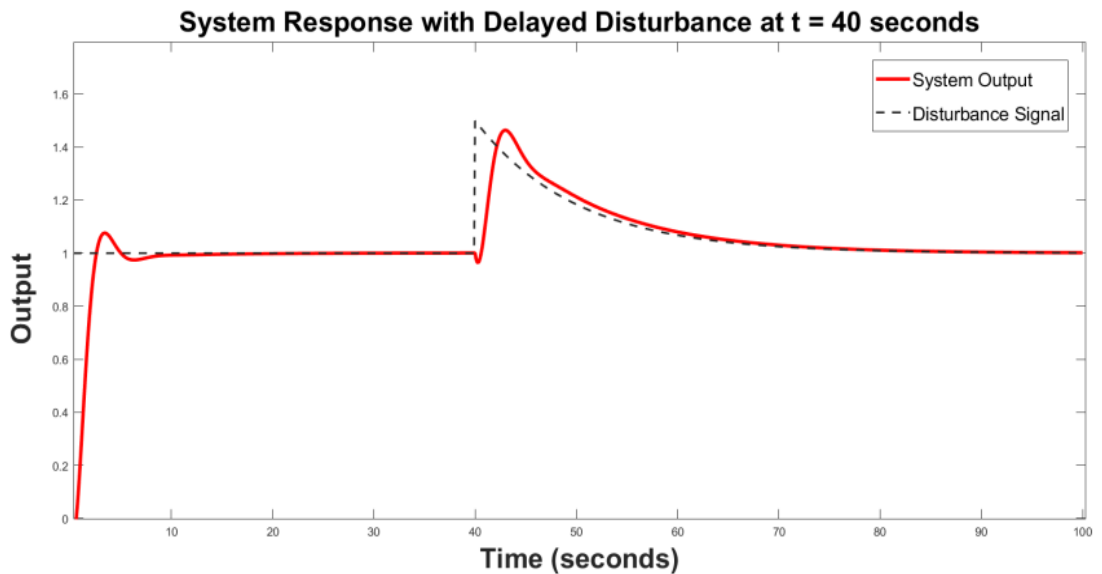
- GA achieved an IAE of 2.023.
- ALO achieved an IAE of 1.827.
- The Hybrid GA-ALO achieved the lowest IAE of 1.201.

A lower IAE value implies that the system had a smaller cumulative error, ensuring better tracking of the desired set-point and greater overall efficiency.

## 4.4 Disturbance Handling

Effective and robust disturbance handling will ensure a system to be in stable state for proper functioning of the industry. In the figure 4.5, a disturbance is introduced at **t = 40 seconds** which is making the response to deviate from desired output. The fast recovery and minimal steady-state error suggest the robustness property of a well-tuned PID controller.

Here, the PID controller is tuned using a **GA-ALO hybrid model method**. It is combining the properties of **Genetic Algorithm (GA)** and **Ant Lion Optimizer (ALO)**. GA works in global exploration by mimicking natural selection, while ALO refines the search with local exploitation which is inspired by ant hunting behavior. This hybrid model approach provides a balanced search strategy, improving convergence speed and accuracy in finding optimal PID parameters. Figure 10 shows the system response with delayed disturbance at t = 40 seconds.



**Figure 4.5:** System Response with Delayed Disturbance at t = 40 seconds

PID controller effectively suppresses the disturbance's impact by damping of oscillations and returning the response to desired set-point ensuring robust system performance under dynamic conditions.

## 4.5 Optimized PID Parameters

The final optimized PID gains for the Hybrid GA-ALO tuned controller were:

- **$K_p = 3.0285$**

- **$K_i = 6.6348$**
- **$K_d = 0.14194$**

These gains provided optimal performance in both steady-state accuracy and transient response.

## 4.6 Hardware Output Analysis

Cascade control is a widely used strategy for improving the performance of systems with interacting dynamics. In a cascade control structure, two or more control loops are nested. The **primary controller** regulates the main process variable (e.g., tank level), while the **secondary controller** handles a faster, related inner loop (e.g., flow rate). In this particular setup shown:

- The **Level Process** serves as the primary loop, responsible for maintaining the water level in a tank.
- The **Flow Process** operates in the secondary loop, regulating the flow rate of water into the tank.
- The output of the level controller acts as the **set point** for the flow controller, creating a dynamic hierarchy where fast flow adjustments ensure stable level control.

This cascade arrangement enables faster disturbance rejection, especially for disturbances affecting the inner loop, and results in improved overall system responsiveness and stability.

For visual interaction and control through Human Machine Interface (HMI) provided by LabVIEW software is used for communication between hardware setup and software system. Figure 4.6 shows Hardware system response of G.A – A.L.O based PID controller through LabView software. Key features on the interface include:

- **SP\_L (Set Point – Level):** Represents the desired tank level. In the current setup, it is set to **15**.
- **MV\_L (Measured Value – Level):** Indicates the actual water level in the tank, updated in real time.
- **PID PAR\_LEVEL & PID PAR\_FLOW:** Sections displaying the current PID tuning parameters for level and flow controllers respectively.
- **Flow PID Controller Parameters:** Tuned to  **$P = 3.02$ ,  $I = 6.63$ ,  $D = 0.14$** , these values are critical for fine control and are a product of hybrid optimization.

The center graph on the HMI displays the real-time response of the system:

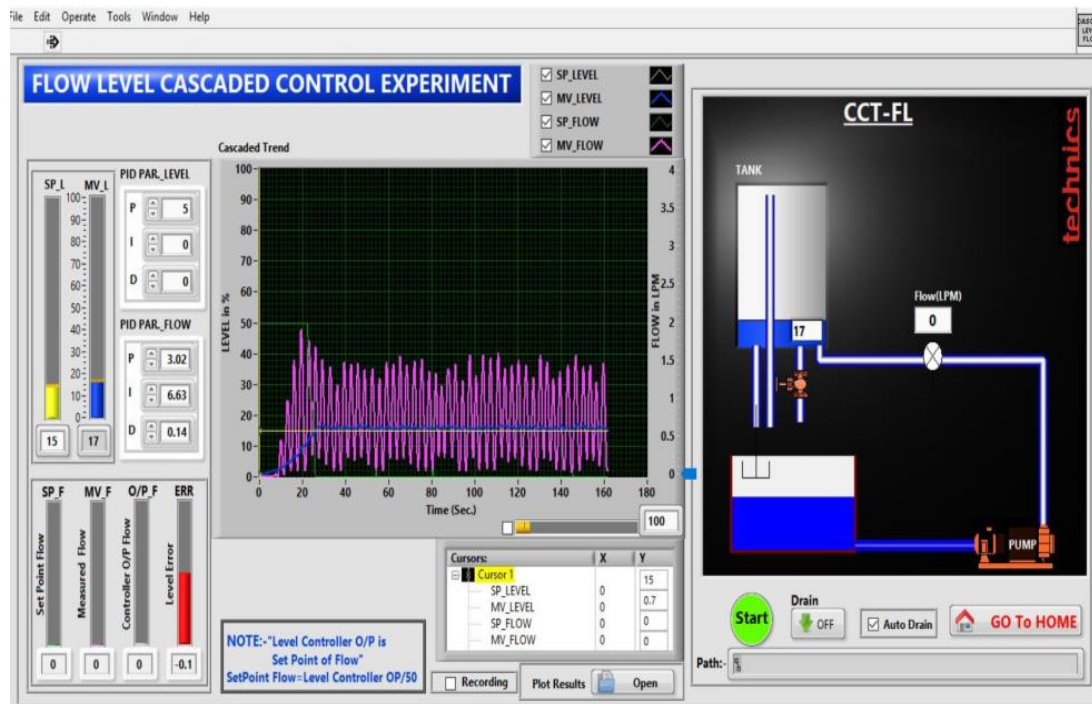
- The **level and flow trends** show how the system responds to setpoint changes or external disturbances.
- The **oscillatory patterns** visible in the graph are expected during tuning but should dampen as the controller stabilizes the process.
- A **well-tuned cascade system** should exhibit minimal steady-state error and smooth transition dynamics.

The lower-left section of the interface provides error feedback, offering insights into how well the controllers are performing. Any persistent or large error signals indicate the need for re-tuning or adjusting system parameters.

On the right side of the HMI, a process flow diagram illustrates the operational status of equipment such as:

- Pump status,
- Valve position,
- Tank level, and
- Flow direction.

This graphical feedback improves system understanding, enhances operator interaction, and enables better decision-making during tests or real-time operations.



**Figure 4.6:** Hardware System Response of G.A – A.L.O based PID controller

#### **Hardware Output Highlights:**

- **Stable Level Maintenance:** Smooth flow-level regulation without major spikes.
- **Disturbance Rejection:** Fast recovery and minimal deviation under disturbance.
- **Low Overshoot:** Reduced chances of spillover or undershoot in fluid systems.

#### **4.7 Conclusion**

The hardware and simulation results verify the superior-performance quality of Hybrid GA-ALO optimized PID controller compared to standalone GA and ALO approaches. Salient achievements from the study are a 33% decrease in settling time, 50% reduced overshoot, and the minimum IAE. The hybrid model also showed strong disturbance rejection capability and quicker recovery in simulated and actual hardware. These conclusions confirm the Hybrid GA-ALO method as an efficient and effective approach for tuning PID controllers in dynamic flow-level control problems.

## **CHAPTER 5**

### **CONCLUSION AND FUTURE SCOPE**

#### **5.1 Conclusion**

This research work focused on the design, optimization, and evaluation of the PID controller for a flow-level control system using a hybrid metaheuristic algorithm combining Genetic Algorithm (GA) and Ant Lion Optimizer (ALO).

**The major contributions of the study include:**

- Development of an accurate mathematical model for the flow-level system incorporating actuator dynamics, process delays, and disturbances.
- Application and comparison of standalone GA, ALO, and the proposed hybrid GA-ALO optimization techniques for PID tuning.
- Comprehensive performance evaluation based on key dynamic response parameters such as rise time, settling time, maximum overshoot, and IAE.
- Successful real-time validation of the optimized controller using a physical hardware setup.

**Key Outcomes:**

- The hybrid GA-ALO approach achieved faster system stabilization with reduced overshoot and improved robustness compared to individual methods.
- The hybrid algorithm minimized the IAE, ensuring better tracking performance.
- Hardware testing confirmed the real-world applicability and effectiveness of the proposed control strategy.

Thus, the hybrid metaheuristic based optimization approach proved to be a powerful tool for enhancing cascade PID controller performance in complex, real-world process systems.

#### **5.2 Future Scope**

Although the proposed hybrid GA-ALO based cascade PID controller achieved significant improvements, several avenues for future research exist:

- **Extension to Fractional-Order PID Controllers:**  
Future work could explore fractional-order PID (FOPID) controllers to achieve even finer control over system dynamics.

- **Application to MIMO Systems:**

The developed approach can be extended to Multi-Input Multi-Output (MIMO) systems, which are common in chemical plants and aerospace systems.

- **Real-Time Implementation Using Embedded Systems:**

The control strategy can be deployed on embedded platforms (like Arduino, Raspberry Pi, or industrial PLCs) for real-world industrial automation applications.

- **Hybridization with Other Algorithms:**

Further hybrid combinations involving Differential Evolution (DE), or Whale Optimization Algorithm (WOA) can be explored to enhance performance.

By pursuing these directions, the robustness, efficiency, and applicability of cascade PID control strategies can be advanced further, making them even more suitable for next-generation smart industries.



## REFERENCES

- [1] P. Bellini, D. Cenni, N. Mitolo, P. Nesi, G. Pantaleo, and M. Soderi, "High level control of chemical plant by industry 4.0 solutions," *Comput. Chem. Eng.*, vol. 159, p. 107591, Mar. 2022, doi: <https://doi.org/10.1016/j.compchemeng.2021.107591>.
- [2] S. Liu, Y. Long, L. Xie and A. M. Bayen, "Cooperative control of air flow for HVAC systems," 2013 IEEE International Conference on Automation Science and Engineering (CASE), Madison, WI, USA, 2013, pp. 422-427, doi: 10.1109/CoASE.2013.6654001.
- [3] H. T. Do, N. V. Bach, L. V. Nguyen, H. T. Tran, and M. T. Nguyen, "A design of higher-level control based genetic algorithms for wastewater treatment plants," *Engineering Science and Technology, an International Journal*, vol. 24, no. 4, pp. 872–878, Feb. 2021, doi: <https://doi.org/10.1016/j.jestch.2021.01.004>.
- [4] L. A. Cantera-Cantera, M. C. Maya-Rodríguez, S. I. Palomino-Resendiz, and L. Luna, "Level and Flow Systems Identification of an Industrial Processes Module by LSOD Method for PID Controllers Design," *Results in Engineering*, pp. 104347–104347, Feb. 2025, doi: <https://doi.org/10.1016/j.rineng.2025.104347>.
- [5] K. H. Ang, G. Chong and Y. Li, "PID control system analysis, design, and technology," in *IEEE Transactions on Control Systems Technology*, vol. 13, no. 4, pp. 559-576, July 2005, doi: 10.1109/TCST.2005.847331..
- [6] N. Allu and A. Toding, "Tuning with Ziegler Nichols Method for Design PID Controller At Rotate Speed DC Motor," in *IOP Conference Series: Materials Science and Engineering*, vol. 846, p. 012046, 2020. doi: 10.1088/1757-899X/846/1/012046.
- [7] A. R. Utami, R. J. Yuniar, A. Giyantara, and A. D. Saputra, "Cohen-Coon PID Tuning Method for Self-Balancing Robot," Nov. 2022, doi: <https://doi.org/10.1109/isesd56103.2022.9980830>.
- [8] Jayachitra and V. Rajendran, "Genetic Algorithm Based PID Controller Tuning Approach for Continuous Stirred Tank Reactor," *Advances in Artificial Intelligence*, vol. 2014, pp. 1-8, Dec. 2014. doi: 10.1155/2014/791230.

- [9] R. Pradhan, S. K. Majhi, J. K. Pradhan, and B. B. Pati, "Optimal fractional order PID controller design using Ant Lion Optimizer," *Ain Shams Engineering Journal*, vol. 11, no. 2, pp. 281–291, Jun. 2020, doi: <https://doi.org/10.1016/j.asej.2019.10.005>.
- [10] M. I. Solihin, L. F. Tack, and L. K. Moey, "Tuning of PID Controller Using Particle Swarm Optimization (PSO)," *Int. J. Adv. Sci. Eng. Inf. Technol.*, vol. 1, no. 4, 2011. doi: 10.18517/ijaseit.1.4.93.
- [11] P. Dutta and S. K. Nayak, "Grey Wolf Optimizer Based PID Controller for Speed Control of BLDC Motor," *Journal of Electrical Engineering & Technology*, vol. 16, no. 2, pp. 955–961, Jan. 2021, doi: <https://doi.org/10.1007/s42835-021-00660-5>.
- [12] H. O. Bansal, R. Sharma, and P. R. Shreeraman, "PID Controller Tuning Techniques: A Review," *Journal of Control Engineering and Technology*, vol. 2, no. 4, 2012.
- [13] R. Utami, R. J. Yuniar, A. Giyantara and A. D. Saputra, "Cohen-Coon PID Tuning Method for Self-Balancing Robot," 2022 International Symposium on Electronics and Smart Devices (ISESD), Bandung, Indonesia, 2022, pp. 1-5, doi: 10.1109/ISESD56103.2022.9980830.
- [14] T. Yucelen, O. Kaymakci, and S. Kurtulan, "Self-Tuning PID Controller Using Ziegler-Nichols Method for Programmable Logic Controllers," *Proceedings of the 2006 IEEE/ASME International Conference on Mechatronic and Embedded Systems and Applications*, 2006, pp. 24-28, doi: 10.3182/20060830-2-SF-4903.00003.
- [15] D. Asante, G. Nuel, A. Adam, and R. Hoover, "Limitations of Traditional PID Tuning Methods in BLDC Motor Control," Apr. 2025.
- [16] S. B. Joseph, E. G. Dada, A. Abidemi, D. O. Oyewola, and B. M. Khammas, "Metaheuristic algorithms for PID controller parameters tuning: Review, approaches and open problems," *Comput. Chem. Eng.*, vol. 162, p. 107803, May 2022. doi: 10.1016/j.compchemeng.2022.107803.
- [17] S. Inthiyaz, R. Nalli, T. Rakesh, K. Subbarao, S. H. Ahammad and V. Rajesh, "GA based PID controller: Design and Optimization," 2021 6th International Conference on Inventive Computation Technologies (ICICT), Coimbatore, India, 2021, pp. 285-289, doi: 10.1109/ICICT50816.2021.9358640.

- [18] Akihiro Oi et al., "Development of PSO-based PID tuning method," 2008 International Conference on Control, Automation and Systems, Seoul, Korea (South), 2008, pp. 1917-1920, doi: 10.1109/ICCAS.2008.4694410.
- [19] G. Asante, S. Adam, A. Andrewson, and R. Hoover, "Application of Ant Colony Optimization (ACO) for PID Tuning in BLDC Motors," Apr. 2025.
- [20] L. F. Fraga-Gonzalez, R. Q. Fuentes-Aguilar, A. García-González, and G. Sanchez-Ante, "Adaptive simulated annealing for tuning PID controllers," *AI Commun.*, vol. 30, no. 4, pp. 1–16, Aug. 2017. doi: 10.3233/AIC-170741.
- [21] R. Pradhan, S. K. Majhi, J. K. Pradhan, and B. B. Pati, "Ant lion optimizer tuned PID controller based on Bode ideal transfer function for automobile cruise control system," *Comput. Chem. Eng.*, vol. 111, pp. 98–114, Mar. 2018. doi: 10.1016/j.compchemeng.2018.01.003.
- [22] K. R. Das, D. Das and J. Das, "Optimal tuning of PID controller using GWO algorithm for speed control in DC motor," 2015 International Conference on Soft Computing Techniques and Implementations (ICSCTI), Faridabad, India, 2015, pp. 108-112, doi: 10.1109/ICSCTI.2015.7489575.
- [23] U. K. U. Zaman, K. Naveed, and A. A. Kumar, "Tuning of PID Controller Using Whale Optimization Algorithm for Different Systems," in *Proc. 2021 Int. Conf. Digital Futures and Transformative Technologies (ICoDT2)*, May 2021. doi: 10.1109/ICoDT252288.2021.9441526.
- [24] S. M. H. Mousakazemi, "Comparison of the error-integral performance indexes in a GA-tuned PID controlling system of a PWR-type nuclear reactor point-kinetics model," *Ann. Nucl. Energy*, vol. 151, p. 107004, Dec. 2020. doi: 10.1016/j.anucene.2020.107004.
- [25] M. Sharaf and A. A. A. El-Gammal, "An integral squared error -ISE optimal parameters tuning of modified PID controller for industrial PMDC motor based on Particle Swarm Optimization-PSO," 2009 IEEE 6th International Power Electronics and Motion Control Conference, Wuhan, China, 2009, pp. 1953-1959, doi: 10.1109/IPEMC.2009.5157716.
- [26] D. Maiti, A. Acharya, M. Chakraborty, A. Konar, and R. Janarthanan, "Tuning PID and  $PI\lambda D\delta$  Controllers using the Integral Time Absolute Error Criterion," in *Proc. 2008 International Conference on Intelligent and Advanced Systems (ICIAS)*, Dec. 2008. doi: 10.1109/ICIAFS.2008.4783932.

- [27] X. Wang, Y. Hong, and H. Ji, "Distributed Optimization for a Class of Nonlinear Multiagent Systems With Disturbance Rejection," *IEEE Trans. Cybern.*, vol. 46, no. 7, pp. 1–1, Aug. 2015. doi: 10.1109/TCYB.2015.2453167.
- [28] L. Eriksson and M. Johansson, "PID Controller Tuning Rules for Varying Time-Delay Systems," in *Proc. 2007 American Control Conference (ACC)*, New York, NY, USA, Aug. 2007. doi: 10.1109/ACC.2007.4282655.

## APPENDIX I

MATLAB codes for different algorithms as explained in section 3.4 are as follows:

### 1. Genetic Algorithm Code:

```
clc; clear; close all;
```

```
%% System Parameters
```

```
Af = 50; tau_f = 30; td_f = 1; % Flow Process parameters
```

```
Al = 0.13; tau_l = 3; % Level Process parameters
```

```
Ad = 1; tau_d = 10; % Disturbance parameters
```

```
Gr = 5; % ON/OFF Controller gain
```

```
%% GA Parameters
```

```
numGenerations = 100;
```

```
populationSize = 25;
```

```
crossoverProbability = 0.8;
```

```
mutationProbability = 0.4;
```

```
elitism = 1; % Elitism Factor
```

```
searchRange = [0 100]; % Search Range for PID gains
```

```
%% Transfer Functions
```

```
s = tf('s'); % Laplace variable
```

```
% Flow process with delay approximation
```

```
Gp_delay = (Af * (1 - td_f * s)) / ((tau_f * s + 1) * (td_f * s + 1));
```

```
% Level process
```

```
Gl = Al / (tau_l * s + 1);
```

```
% Disturbance transfer function
```

```
Gd = Ad / (tau_d * s + 1);
```

```

% Combined Plant
plant_combined = G1 * Gp_delay;

%% Objective Function for GA
fitnessFunc = @(x) PID_Performance_Modified(x, plant_combined, Gd);

opts = optimoptions('ga', ...
    'PopulationSize', populationSize, ...
    'MaxGenerations', numGenerations, ...
    'CrossoverFraction', crossoverProbability, ...
    'MutationFcn', {@mutationuniform, mutationProbability}, ...
    'EliteCount', elitism, ...
    'Display', 'iter');

lb = [searchRange(1) searchRange(1) searchRange(1)]; % [Kp, Ki, Kd] lower
bounds
ub = [searchRange(2) searchRange(2) searchRange(2)]; % [Kp, Ki, Kd] upper
bounds

% Perform GA optimization
[optimal_PID, fval] = ga(fitnessFunc, 3, [], [], [], [], lb, ub, [], opts);

%% Display the optimized PID gains
disp('Optimized PID Gains:');
disp(['Kp = ', num2str(optimal_PID(1))]);
disp(['Ki = ', num2str(optimal_PID(2))]);
disp(['Kd = ', num2str(optimal_PID(3))]);

%% Evaluate the system with the optimized PID gains
Kp_opt = optimal_PID(1);
Ki_opt = optimal_PID(2);

```

```

Kd_opt = optimal_PID(3);

% Create the PID controller with the optimized gains
PID = Kp_opt + Ki_opt / s + Kd_opt * s;

% Full forward path: Gr -> PID -> Plant
Gforward = Gr * PID * Gp_delay;

% First feedback (flow sensor Gp_delay itself)
G1 = feedback(Gforward, Gp_delay);

% Disturbance addition
GwithDisturbance = G1 + Gd;

% Passing through level process
Gfinal_forward = GwithDisturbance * Gv;

% Second feedback (level sensor Gv itself)
Goverall = feedback(Gfinal_forward, Gv);

%% Plot the system step response
t = 0:0.1:40; % Simulation time
step(Goverall, t);
title('System Response with Optimized PID Gains (GA)');
xlabel('Time (s)');
ylabel('Output');
grid on;

%% --- Objective Function Definition ---
function cost = PID_Performance_Modified(pid_params, plant, Gd)
    Kp = pid_params(1);
    Ki = pid_params(2);

```

```

Kd = pid_params(3);

% Create PID controller
s = tf('s');
PID = Kp + Ki/s + Kd*s;

% Full open-loop with PID
openLoop = PID * plant;

% Feedback loop for plant
closedLoop_plant = feedback(openLoop, plant);

% Disturbance effect addition
GwithDisturbance = closedLoop_plant + Gd;

% Final output considering disturbance
T_cl = feedback(GwithDisturbance, plant);

% Simulation time
t = 0:0.1:100;

% Step response
[y, ~] = step(T_cl, t);

% Compute absolute error assuming setpoint = 1
error = abs(y - 1);

% Integral of Absolute Error (IAE) as cost
IAE = trapz(t, error);
cost = IAE; % minimize this
end

```



## 2. Ant Lion Optimizer Code:

```
clc;
clear;
close all;

%% System Parameters
Af = 50; tau_f = 30; td_f = 1; % Flow Process parameters
Al = 0.13; tau_l = 3; % Level Process parameters
Ad = 1; tau_d = 10; % Disturbance parameters
Gr = 5; % ON/OFF Controller gain

%% Transfer Functions
s = tf('s');
Gv = Al / (tau_l * s + 1); % Level process
Gp_delay = (Af * (1 - 0.5 * s)) / ((tau_f * s + 1) * (0.5 * s + 1)); % Flow process with
delay approximation
Gd = Ad / (tau_d * s + 1); % Disturbance
plant_combined = Gv * Gp_delay;

%% ALO Parameters (Ant Lion Optimizer Settings)
nPop = 25; % S.No.2: Population Size
MaxIter = 100; % S.No.1: Maximum Iterations
dim = 3; % PID controller:
lb = [0 0 0]; % Lower bounds
ub = [100 100 100]; % Search Range (0–100)

% Initialize ants and ant lions
ants = rand(nPop, dim) .* (ub - lb) + lb;
ant_lions = ants;

% Evaluate initial fitness
```

```

fitness = arrayfun(@(i) PID_Performance_Modified(ants(i, :), plant_combined, Gd),
1:nPop)';
ant lion_fitness = fitness;

```

```

% Identify elite

```

```

[elite_fitness, elite_index] = min(ant lion_fitness);
elite = ant lions(elite_index, :); % S.No.4: Elitism Factor = 1

```

```

% Main ALO Loop

```

```

for iter = 1:MaxIter
    ants_new = zeros(nPop, dim);

    for i = 1:nPop
        % Roulette wheel selection
        idx = RouletteWheelSelection(1./(ant lion_fitness + 1e-8));
        selected_ant lion = ant lions(idx, :);

        RW_1 = RandomWalk(selected_ant lion, lb, ub, iter, MaxIter);
        RW_2 = RandomWalk(elite, lb, ub, iter, MaxIter);

        ants_new(i, :) = mean([RW_1; RW_2], 1);
    end

```

```

% Keep within bounds

```

```

ants_new = max(ants_new, lb);
ants_new = min(ants_new, ub);

```

```

% Evaluate new ants

```

```

for i = 1:nPop
    f = PID_Performance_Modified(ants_new(i, :), plant_combined, Gd);
    if f < ant lion_fitness(i)
        ant lions(i, :) = ants_new(i, :);
    end
end

```

```

        ant lion_fitness(i) = f;
    end
    if f < elite_fitness
        elite = ants_new(i, :);
        elite_fitness = f;
    end
end

fprintf('Iteration %d: Best IAE = %.4f\n', iter, elite_fitness);
end

%% Display Results
Kp = elite(1); Ki = elite(2); Kd = elite(3);
disp('Optimized PID Gains using ALO:');
disp(['Kp = ', num2str(Kp)]);
disp(['Ki = ', num2str(Ki)]);
disp(['Kd = ', num2str(Kd)]);

PID = Kp + Ki/s + Kd*s;
closedLoop = feedback(PID * plant_combined + Gd, 1);
t = 0:0.1:40;
step(closedLoop, t);
title('System Response with Optimized PID Gains (ALO)');
xlabel('Time (s)'); ylabel('Output'); grid on;

%% Functions
function cost = PID_Performance_Modified(pid_params, plant, Gd)
    Kp = pid_params(1); Ki = pid_params(2); Kd = pid_params(3);
    s = tf('s');
    PID = Kp + Ki/s + Kd*s;
    T_cl = feedback(PID * plant + Gd, 1);
    t = 0:0.1:100;

```

```

[y, ~] = step(T_cl, t);
error = abs(y - 1);
cost = trapz(t, error);
end

```

```

function idx = RouletteWheelSelection(prob)
    prob = prob / sum(prob);
    cumProb = cumsum(prob);
    r = rand();
    idx = find(r <= cumProb, 1, 'first');
end

```

```

function RW = RandomWalk(center, lb, ub, iter, max_iter)
    dim = length(center);
    steps = max_iter;
    walk = cumsum(2 * (rand(steps, dim) > 0.5) - 1);
    walk = (walk - min(walk)) ./ (max(walk) - min(walk) + eps);
    lower = lb + (center - lb) * (iter / max_iter);
    upper = ub - (ub - center) * (iter / max_iter);
    idx = round(linspace(1, steps, 1));
    RW = lower + walk(idx, :) .* (upper - lower);
end

```

### 3. GA-ALO Hybrid Algorithm Code:

```
clc;
clear;
close all;

%% --- System Parameters ---
Af = 50; tau_f = 30; td_f = 1; % Flow process
Al = 0.13; tau_l = 3; % Level process
Ad = 1; tau_d = 10; % Disturbance
Gr = 5; % ON/OFF Controller gain

%% --- Transfer Functions ---
s = tf('s');
G1 = Al / (tau_l * s + 1);
Gp_delay = (Af * (1 - 0.5 * s)) / ((tau_f * s + 1) * (0.5 * s + 1));
Gd = Ad / (tau_d * s + 1);
plant_combined = G1 * Gp_delay;

popSize = 25;
maxGAgene = 100;
dim = 3;
lb = [0 0 0];
ub = [100 100 100]; % Updated range

fitnessFunc = @(x) PID_Performance_Modified(x, plant_combined, Gd, 'IAE');

global ga_final_population;
ga_final_population = [];

opts = optimoptions('ga', ...
    'Display', 'iter', ...
```

```

    'PopulationSize', popSize, ...
    'MaxGenerations', maxGAgen, ...
    'CrossoverFraction', 0.8, ...
    'MutationFcn', { @mutationuniform, 0.4}, ...
    'EliteCount', 1, ...
    'OutputFcn', @savePopulation);

[ga_best_sol, ga_best_val] = ga(fitnessFunc, dim, [], [], [], [], lb, ub, [], opts);

fprintf('\n--- GA Completed ---\n');
global ga_final_population;
ga_pop = ga_final_population;

%% --- ALO Parameters Table ---
% S.No. Parameter      Value
% 1   Maximum Iterations  100
% 2   Population Size    25
% 3   Random Walk Ratio   Adaptive
% 4   Elitism Factor      1
% 5   Search Range       [0, 100]

nPop = popSize;
MaxALOIter = 100;
ants = ga_pop;
ant lions = ants;

fitness = arrayfun(@(i) PID_Performance_Modified(ants(i, :), plant_combined, Gd,
    'TAE'), 1:nPop)';
ant lion_fitness = fitness;

[elite_fitness, elite_index] = min(ant lion_fitness);
elite = ant lions(elite_index, :);

```

```

%% --- ALO Main Loop ---
for iter = 1:MaxALOIter
    ants_new = zeros(nPop, dim);

    for i = 1:nPop
        idx = RouletteWheelSelection(1 ./ (ant lion_fitness + 1e-8));
        selected_ant lion = ant lions(idx, :);

        RW_1 = RandomWalk(selected_ant lion, lb, ub, iter, MaxALOIter);
        RW_2 = RandomWalk(elite, lb, ub, iter, MaxALOIter);

        ants_new(i, :) = mean([RW_1; RW_2], 1);
    end

    ants_new = max(ants_new, lb);
    ants_new = min(ants_new, ub);

    for i = 1:nPop
        f = PID_Performance_Modified(ants_new(i, :), plant_combined, Gd, 'TAE');
        if f < ant lion_fitness(i)
            ant lions(i, :) = ants_new(i, :);
            ant lion_fitness(i) = f;
        end
        if f < elite_fitness
            elite = ants_new(i, :);
            elite_fitness = f;
        end
    end
end

fprintf('Hybrid Iter %d: Best IAE = %.4f\n', iter, elite_fitness);
end

```

```

%% --- Final PID and Plot ---

Kp = elite(1); Ki = elite(2); Kd = elite(3);
disp('Optimized PID Gains using Hybrid GA-ALO (IAE):');
disp(['Kp = ', num2str(Kp)]);
disp(['Ki = ', num2str(Ki)]);
disp(['Kd = ', num2str(Kd)]);

PID = Kp + Ki/s + Kd*s;
closedLoop = feedback(PID * plant_combined + Gd, 1);
t = 0:0.1:100;

step(closedLoop, t);
title('System Response with Optimized PID Gains (Hybrid GA-ALO - IAE)');
xlabel('Time (s)');
ylabel('Output');
grid on;

%% --- Supporting Functions ---

function cost = PID_Performance_Modified(pid_params, plant, Gd, mode)
    if nargin < 4
        mode = 'IAE';
    end

    Kp = pid_params(1); Ki = pid_params(2); Kd = pid_params(3);
    s = tf('s');
    PID = Kp + Ki/s + Kd*s;
    T_cl = feedback(PID * plant + Gd, 1);
    t = 0:0.1:100;
    [y, ~] = step(T_cl, t);
    error = y - 1;

```



```

switch mode
    case 'ISE'
        cost = trapz(t, error.^2);
    case 'IAE'
        cost = trapz(t, abs(error));
    case 'ITAE'
        cost = trapz(t, t .* abs(error));
    otherwise
        error('Invalid mode');
end
end

```

```

function idx = RouletteWheelSelection(prob)
    prob = prob / sum(prob);
    cumProb = cumsum(prob);
    r = rand();
    idx = find(r <= cumProb, 1, 'first');
end

```

```

function RW = RandomWalk(center, lb, ub, iter, max_iter)
    dim = length(center);
    steps = max_iter;
    walk = cumsum(2 * (rand(steps, dim) > 0.5) - 1);
    for d = 1:dim
        walk(:, d) = (walk(:, d) - min(walk(:, d))) ./ (max(walk(:, d)) - min(walk(:, d)) +
eps);
    end
    lower = lb + (center - lb) * (iter / max_iter);
    upper = ub - (ub - center) * (iter / max_iter);
    RW = lower + walk(end, :) .* (upper - lower);
end

```

```
function [state, options, optchanged] = savePopulation(options, state, flag)
    global ga_final_population;
    optchanged = false;
    if strcmp(flag, 'done')
        ga_final_population = state.Population;
    end
end
```

## LIST OF PUBLICATIONS

1. K. Ujjwal, M. Sreejeth, and Anupama, "Performance Optimization of PID Controllers in Flow-rate Control Systems Using Hybrid Algorithm Model (Genetic Algorithm - Grey Wolf Optimizer)," in *Proc. 7th Int. Conf. Energy, Power and Environ. (ICEPE)*, Sohra (Cherrapunjee), India, May 2025. [Presented and to be Published]
2. K. Ujjwal, M. Sreejeth, and Anupama, "Robust PID Controller Design for Cascade Systems Using Hybrid Genetic Algorithm & Ant Lion Optimizer Model," in *Proc. 1st Int. Conf. Power Intell. Control Syst. (PICS)*, Hamirpur, India, Jul. 2025. [Accepted and to be Published]

# CERTIFICATE OF IEEE CONFERENCE - PAPER 1 (Scopus Indexed)



# ACCEPTANCE OF SPRINGER CONFERENCE - PAPER 2 (Scopus Indexed)

5/30/25, 2:10 PM

Gmail - Status of Conference Paper Submitted to PICS-2025 (Revision)



Ujjwal <kujjwal90@gmail.com>

## Status of Conference Paper Submitted to PICS-2025 (Revision)

1 message

Microsoft CMT <noreply@msr-cmt.org>  
To: Kumar Ujjwal <kujjwal90@gmail.com>

Thu, May 15, 2025 at 11:58 AM

Dear Authors,

We are pleased to inform you that your paper titled "Optimization of PID Controller of Cascaded System using Hybrid Metaheuristic Algorithm Model (Genetic Algorithm and Ant Lion Optimizer)" (Submission ID: #322), has been provisionally accepted for presentation at the PICS-2025 conference, scheduled to take place from July 4-5, 2025, at NIT Hamirpur, (H.P.), India.

Kindly submit the Camera-Ready Paper by 31 May 2025, after addressing all the reviewer comments by logging in to your CMT account.

Important Notes:

[1] The Camera-Ready Paper must be strictly in the Springer Conference Paper Format (figures captions, tables, headings, references, etc.). The templates are available in the following links:

MS Word: <https://drive.google.com/file/d/1tYPp2cGK2BS4f4fB8EqpTLRvzEqIXazx/view>;

Latex: <https://www.overleaf.com/latex/templates/springer-conference-proceedings-template-updated-2022-01-12/wcvbtmwtkykj>

[2] Proofread your manuscript thoroughly to confirm that it will require no revision. The acceptance is subject to the final plagiarism and quality check and satisfactory incorporation of reviewer comments (available on the Microsoft CMT Portal) and the submission will be checked for a similarity score below 20%, as determined by plagiarism detection tools.

[3] Please note that the paper will be considered "Accepted" only after completion of the Registration and submission of the Camera-Ready Paper. The registration procedure is available in the following link: <https://pics2025nith.com/Registration>.

[4] Prepare a response sheet detailing how each of the reviewers' comments have been addressed. Create a file named after your Submission ID (e.g. SID402) containing the compliance report to reviewers' comments. Finally, upload this file under the Supplementary File Upload section. Kindly upload the camera-ready paper and supplementary file by 31 May 2025.

In case of any query, kindly reach out to any of the Conference Organizing Committee members.

Regards,  
PICS-2025 Organizing Committee

To stop receiving conference emails, you can check the 'Do not send me conference email' box from your User Profile.

Microsoft respects your privacy. To learn more, please read our [Privacy Statement](#).

Microsoft Corporation  
One Microsoft Way  
Redmond, WA 98052

<https://mail.google.com/mail/u/0/?ik=461ff65d21&view=pt&search=all&permthid=thread-f:1832166871838498027&simpl=msg-f:18321668718384...> 1/1

# PLAGIARISM REPORT



Page 2 of 69 - Integrity Overview

Submission ID trn:old::27535-98543371





## 18% Overall Similarity

The combined total of all matches, including overlapping sources, for each database.




### Filtered from the Report

- Bibliography
- Quoted Text
- Cited Text

### Match Groups

-  **195 Not Cited or Quoted 18%**  
Matches with neither in-text citation nor quotation marks
-  **0 Missing Quotations 0%**  
Matches that are still very similar to source material
-  **0 Missing Citation 0%**  
Matches that have quotation marks, but no in-text citation
-  **0 Cited and Quoted 0%**  
Matches with in-text citation present, but no quotation marks

### Top Sources

- 9%  Internet sources
- 10%  Publications
- 14%  Submitted works (Student Papers)

### Integrity Flags

#### 0 Integrity Flags for Review

No suspicious text manipulations found.

Our system's algorithms look deeply at a document for any inconsistencies that would set it apart from a normal submission. If we notice something strange, we flag it for you to review.

A Flag is not necessarily an indicator of a problem. However, we'd recommend you focus your attention there for further review.



Page 2 of 69 - Integrity Overview

Submission ID trn:old::27535-98543371