FINAL YEAR PROJECT 2017-2018

PROGRESS REPORT

OPTIMIZATION OF PID CONTROLLER USING GREY WOLF OPTIMZER AND DRAGONFLY ALGORITHM

Student Name	:	Nik Muhammad Aiman Bin Nik Mohamed Hazli
Student ID	:	126441
Supervisor's Name	:	Dr. Wan Amir Fuad Wajdi Bin Othman
Course Code	:	EEM 449
Program	:	Mechatronic Engineering
School	:	Electrical & Electronic Engineering
University	:	Universiti Sains Malaysia

Acknowledgement

Alhamdulillah, all praises to Allah for endowing me with health, strength and knowledge to complete this research. Here, I would to express my gratitude to everyone, who gave me the opportunity to gain invaluable experience during my final year project.

Foremost, I would like to express my deepest gratitude and appreciation to my supervisor, Dr. Wan Amir Fuad Wajdi Bin Othman. He continuously supervise and support me. Without his help, it is impossible for me to complete my final year project. His guidance are very helpful.

Finally, thanks to my beloved parents, Nik Mohamed Hazli and Azidah Binti Abdul Kadir as well as my siblings for the moral supports. I also need to acknowledge my friends for supporting me throughout my studies. This accomplishment would not been possible without them.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF SYMBOLS AND ABBREVATIONS

SYMBOLS

K_p	Proportional Gain
K_i	Integral Gain
K_d	Differential Gain
M_p	Peak Overshoot
e_{ss}	steady-state error
T_r	Rising time
T_s	Settling time

ABREVATIONS

ABM	Agent Based Modelling
CSO	Cat Swarm Optimisation
DA	Dragonfly Algorithm
DF	Dragonfly
FOPID	Fractional Order Proportional-Integral-Differential
GW	Grey Wolf
GWO	Grey Wolf Optimizer
PID	Proportional-Integral-Differential
SI	Swarm Intelligence

PENGOPTIMUMAM PENGENDALIAN PID MENGGUNAKAN PENGOPTIMUM SERIGALA PUTIH DAN ALGORITMA PEPATUNG

ABSTRAK

Optimisasi adalah satu cara untuk mencari keseimbangan dalam reka bentuk yang perlu ditoleransi antara faktor yang tertentu seperti kecerdasan dan kos. Dalam bidang kejuruteraan, salah satu optimisasi yang biasa dijumpai adalah optimisasi pengawal kadaran-kamiran-bezaan (PID). Optimisasi pengawal berkadar-integral-berbeza (PID) adalah susah kerana terdapat tiga parameter yang hendak dikawal, parameter kadaran, K_p, parameter kamiran, K_i, dan parameter bezaan, K_d. Dalam kajian ini, kepintaran berkumpulan digunakan untuk menyelesaikan masalah optimisasi. Algoritma serigala kelabu dan algoritma pepatung dipilih. Tiga sistem pangkal dipilih. Sistem pertama adalah berdasarkan sistem bola dan gelung, dan sistem kedua pula berdasarkan sistem servo motor arus terus. Sistem yang terakhir berdasarkan sistem motor berberus arus terus. Fungsi objektif untuk kajian ini adalah fungsi kos. Kriteria untuk fungsi kos adalah M_p, puncak lanjakan, e_{ss},ralat keadaan mantap,T_s, masa penetapan and masa kenaikan, T_r. Walau bagaimanapun, untuk mengunakan algoritma sepenuhnya, parameter algoritma perlu ditetapkan sebaiknya. Dalam kajian ini, nombor agen pencarian untuk kedua-dua algoritma. Kriteria berhenti juga perlu ditetapkan. Dalam kajian ini, nombor maksima kitaran untuk kedua-dua algoritma. Keputusan yang dijangkan adalah kedua-dua algoritma dapat mengoptimumkan pengawal berkadar-integral-berbeza (PID). Namun begitu, prestasi untuk setiap sistem dijangkan berbeza daripada algoritma yang berbeza.

OPTIMISATION OF PID CONTROLLER USING GREY WOLF OPTIMIZER AND DRAGONFLY ALGORITHM

ABSTRACT

Optimisation is a method to find a balance performance when the design has to compromise between a certain factors, which affects fitness and cost. In engineering field, one of the common optimisation problem is optimisation of PID controller. Optimisation is difficult to optimise as there are three parameters that need to be tuned, K_p, Integral parameter, K_i, and derivative parameter, K_d. In this research, swarming intelligence is used to solve optimisation problem. Grey Wolf Optimizer and Dragonfly Algorithm were chosen. Three plant system were used in this study. First system is based on the ball and hoop system and second system is based on the DC servo motor. Last system is based on the brushed DC motor. Objective function in this research, cost function was chosen. The criteria of the cost function are low peak overshoot, M_p, low steady-state error, e_{ss}, low settling time, T_s, and low rise time, T_r. However, to fully utilize the algorithm, the parameter of the algorithm need to be set properly. In this case, the right number of the search agents for both algorithm. The stopping criteria also need to be identified. In this study, maximum number of iterations is the stopping criteria. The expected result is the algorithms are able to optimise the PID controller. However, the performance of system is expected to be different from different algorithm.

CHAPTER 1

INTRODUCTION

1.1 Research Background

Search algorithm is an algorithm that used to achieve a certain objective in problem domain [1]. The appropriate search algorithm is chosen depending on the problem domain. The problem can be optimisation, classification or satisfaction problem. Optimisation problem is the problem of searching the finest solution from all possible solutions. The problems can be split into two categories depending on the variables either it is continuous or discrete. The discrete optimization is searching for an object such as an integer, permutation or graph from a finite while the continuous optimization is usually involving constrained problems and multimodal problems. Essentially, the aim of objective optimisation is to minimise or to maximise the value of a function [2].

For this comparative study, meta-heuristic algorithm which is part of stochastic optimisation algorithms was studied. The algorithm used in this research was Grey Wolf Optimizer (GWO) and Dragonfly Algorithm (DA). It is one of the subclass of meta-heuristics, Swarm Intelligence (SI) methods [3]. For the example, one of the swarm intelligence is Cat Swarm Optimization. Cat Swarm Optimization (CSO) is actually an algorithm influenced based on the natural behaviour of the cat. Cats is one of the animal that seen to spend their time in resting but actually they have high alertness and curiosity about their surroundings and moving objects in their environment. This behaviour helps cats in tracking and hunting them down. While resting, they actually conserve their energy. According to Chu and Tsai, there are 2 main mode of the cats, which are seeking mode and tracing mode. During seeking mode, the cat is resting while keeping an eye on its environment. They decide to move when they sense a prey or danger. If the cat decides to move, it does that slowly and cautiously. The second mode is tracing mode. The tracing mode simulates the cat chasing a prey. After finding a prey while seeking mode, the cat decides its movement speed and direction based on the prey‘s position and speed. This algorithm is terminated if the cat hunt down a prey [4].

The performance of the algorithm is measured in terms of optimising Proportional –Integral- Derivative (PID) controller. Especially in the industry, Proportional-Integral-Derivative (PID) controller is one of the common controller that have been used. Proportional-Integral-Derivative (PID) controller consists of a few tuning parameters that need to be optimised [5].The tuning parameters are Proportional gain (K_p), Integral gain (K_i) and Derivative gain (K_d). The summation of proportional, integral and derivative terms as shown in Equation 1.1 is the output of the PID controller.

$$u(t) = K_p e(t) + K_i \int_0^t e(t)dt + K_d \frac{d}{dt} e(t) \qquad 1.1$$

Tuning a Proportional-Integral-Derivative (PID) controller is quite a hassle even though it only has 3 parameters. These parameters need to tune properly as it will the transient response of the system. For example, overshoot (M_p), rise time (T_r), steady state error (e_{ss}) and settling time (T_s) in the step response of a closed-loop time system. The selection of the parameters is important so that it meets the design requirements [6]. There is a lot of methods that have been proposed to optimise PID controller such as manual tuning, Ziegler- Nichols and Tyreus Luyben method. However, these methods also have certain limitation. For example, manual tuning requires personal experience and, some trial and error. Ziegler-Nichols adjusts the controller for one fourth of amplitude damping response, which overshoots and fluctuate a bit, leaves the controller with very little sturdiness, which can lead to loop instability [7].

Since there are some shortcomings of the mentioned above methods, meta-heuristic optimization techniques is another option. The algorithms inspired from the behaviour of animals such as spider, ant and bees [8]. The reason why they are preferable because meta-heuristics are simple. They are related to simple concept such as physical phenomena, animals' behaviours or evolutionary concepts. The directness of them allow scientists to learn, stimulate, propose, combine, improve and apply them. Second, they are so flexible that they can be apply to different problem. Third, they are suitable for real problems with fancy and unexplored derivative information. Finally, they are good for avoiding to local optima and optimizing hard real-time problem [8].

The effectiveness of the algorithm was compared by thoroughly analysing the minimisation of objective function. In this study, the objective is to reduce the cost function. The algorithm must be able to find the optimal solution in order to reduce cost function. Minimal overshoot, fast rise time and settling time and smaller steady state error. These criteria is known as cost function [9]. It will be used to evaluate the solutions in order to find the best solutions among the feasible solutions.

1.2 Problem Statement

In the engineering world, the optimisation problem is always a challenge to the engineer to solve it as the size and complexity are also increased. However, as the technology is developing, more algorithms are developed such as Genetic Algorithm, Simulated Annealing and Ant Colony Optimizer. Lately, nature-inspired metaheuristics is getting a lot attention. Many recent metaheuristics are being developed. These metaheuristics depend on how the nature phenomenon simulated. Nonetheless, these approaches are not very accurate and they not always returning an optimal result. As they are simulating the nature phenomenon, the simulation is not perfect as they are bound to the rules of Agent Based Modelling (ABM). Recent years, new algorithms are developed to compensate these drawbacks. In order to review the performance of selected algorithm, parameters of PID controller are used to test the algorithms in real world application.

1.3 Objectives of research

The aim of this research to study the performance of Grey Wolf Optimiser (GWO) and Dragonfly Algorithm (DA) for tuning the parameters K_p ,K_i and K_d of PID controller. Below are the objectives to be achieved from this study:

- To optimise PID controller using Grey Wolf Optimiser (GWO)
- To optimise PID controller using Dragonfly Algorithm (DA)
- To compare the performance of Grey Wolf Optimizer and Dragonfly Algorithm

1.4 Scope of research

This study solely focus on optimising the parameters of PID controller to achieve optimum result. Proportional-Integral-Derivative (PID) controller consists of a few tuning parameters that need to be optimised. The tuning parameters are Proportional gain (K_p), Integral gain (K_i) and Derivative gain (K_d). But, tuning a Proportional-Integral-Derivative (PID) controller is a difficult problem even though it only has 3 parameters. The PID controller parameters must be properly chosen because the selection of parameter will be affecting the transient response of the system. For example, overshoot (M_p), rise time (T_r), steady state error (e_{ss}) and settling time (T_s) in the step response of a closed-loop time system.

The objective of this optimisation is to reduce the cost function of the PID controller. The best possible results of PID controller parameters depend on cost function. The cost function is the transient response of the system. The criteria are minimal peak overshoot, M_p, steady state error, e_{ss}, rise time , $T_{r,}$ and settling time, T_s [7].

1.5 Thesis Outline

This thesis consists of five chapters which include Introduction, Literature Review, Methodology, Result and Discussion, Conclusion and Future Work. Chapter 1 mainly explain about the project overview, problem statement and objectives of this study. Next, chapter 2 consists of basic concept of optimisation, optimisation algorithms and categories of optimisation methods such as heuristics and meta-heuristic methods. This part elaborates the development and implantation of Grey Wolf Optimiser (GWO) Algorithm in optimisation PID controller. In addition, explained the impact of controller parameters to optimize the cost and time by utilizing GWO. Chapter 3 presents about the methodology, design and implementation of GWO in this problem. Also, includes the set of variables and control parameters used as well procedures in order to design solution via MATLAB. Chapter 4 simply presents the results of simulations and analyses the optimum solution for this system. Besides, this section discuss about the minimisation of cost function as objective function. Chapter 5 presents the conclusion that can be drawn based on the carried out results. Finally, considerate the future work which related to this study.

CHAPTER 2

LITERATURE REVIEW

2.1 Introduction

In this chapter, several research papers related to search algorithm, concept of optimisation, metaheuristic and swarming intelligence are reviewed in section 2.2 until section 2.4.1. Those two algorithms that used in this research, Grey Wolf Algorithm (GWO) and Dragonfly Algorithm (DA) are briefly explained in section 2.4.2 and 2.4.3. Section 2.6 is the summary of chapter 2.

2.2 Search Algorithm

Search algorithm is an algorithm that used to find a target or more within a search space. It is also called as “Storage and Retrieval Information” or “Table Look-up”. The search process is mimic the process of gathering information in a computer memory in a way that it can be obtained instantly. The data can be saved for next call or deleted for computational purposes. The saved data need to be organized for a fast retrieval. A very simple search algorithm is a way to find the data that has been saved with a given clue. The problem is to find a precise one in the given search space. The search space is usually the space that contains potential candidate for solution. Each candidate or data is unique. The data will be tested for identification purpose. After the search is done, the result can be only successful or unsuccessful [10]. A very simple and famous search algorithm is Brute Force Algorithm. Brute Force Algorithm is basically sequential searching. It start the search at the very beginning, and go on until a suitable solution is found. This searching procedure is an obvious way to search something. The problem is if the search space or number of candidates is very large, it take long time and waste of computational power. However, it is a useful algorithm as it is a base for search algorithms[11].

2.3 Concept of Optimisation

Optimisation is one of the important process in engineering world. Optimisation is a method to find a balance performance when the design has to compromise between a certain factors, which affects fitness and cost [12]. Practically, optimisation must be able to produce low cost design without sacrifice the fitness. So, criterion must be set in order to define the optimised solution. The criterion is also called as objective function. The objective can be either minimise cost function or maximise fitness function [13]. Objective function is important in order to compare all possible solutions. The best solutions will be selected. This is why sometimes optimisation has more than one objective function, which is also known as multiobjective function. Cost function is a function of error, which usually is difference between desired output and actual output [9]. Fitness function is usually a function to determine the degree of achieving the intended target. Cost function and fitness function are reciprocals of each other. The algorithm that used for optimisation is called optimisation algorithm [14].

2.4 Metaheuristic

One of the solution to optimisation problem is metaheuristic. Metaheuristic is a high level heuristic invented to find a solution to an optimisation problem. Metaheuristic is surprisingly useful for solving optimisation problem that without complete information or enough computational power. Lately, nature-inspired metaheuristics are getting a lot of attention. Nature-inspired metaheuristic are inspired by system of nature. There are a few reasons why metaheuristics are worth getting attention [8]. First, they are simple. Mostly they are inspired by simple concept that can be easily understand. The inspirations are usually related to physical phenomena, animals' behaviour or evolutionary concepts [15]. The simplicity allow the computer scientists to simulate the phenomena using software. Second, mostly metaheuristics are flexible as they are easily can be apply to any structure. They are not bound to any special conditions to make them work since problems usually will be assumed as black boxes problems. Third, they have derivation-free mechanism. They mostly start the solutions randomly without enough information regarding the solution. Finally, they have abilities to avoid or trap in local optima compared to the other optimisation methods. This is due to stochastic nature of metaheuristics which let them to avoid local solutions and search the entire search space [5].

2.4.1 Swarming Intelligence

Swarm is defined by a huge quantity of alike, simple agents communicating locally among themselves, and their habitat, with no central control to let a global beautiful behaviour to combine. Swarm-based algorithms usually is inspired by nature. These algorithms are capable of generating low cost, fast and robust solutions to certain problems. The agents of swarming intelligence usually are insects, birds or swarm individuals. They are modelled using Agent Based Modelling (ABM) to simulate the interaction the swarming behaviour with limited capabilities [16]. The interaction can be categorised into two, direct and indirect. Indirect interaction usually occurs when the agent communicating each other through visually or audio [5]. For example, in Bees Colony Algorithm, the bees communicate each other through dance. Indirect interaction happens when the agent change the environment and other agent react to that new environment such in Ant Colony Optimisation, the ants release pheromone to indicate the trails from food sources to their colony. As for Lion Optimization, it was based on the special lifestyle of lions and their cooperation characteristics [17]. Swarm Intelligence have been successfully applied in different type of problems such as optimisation problems, optimal routes, scheduling and structural optimisation [8]. Although Swarming Intelligences have a lot of potential, it also has a few limitations. First, it is advisable to not apply swarming intelligence in time critical applications as it consumes a lot of time to apply. Second, the tuning parameters are unpredictable and might react differently based on problem domain. Third, it is possible for the intelligence to stagnate. They might trap in a same solution or local solution no matter what tuning parameters are [11].

2.4.2 Grey Wolf Optimizer (GWO)

Grey Wolf Optimizer is a nature-inspired metaheuristic proposed by Seyedali Mirjalili [18]. Grey Wolf Optimizer is basically an algorithm influenced based on the social behaviour of grey wolves, group hunting in addition to the social hierarchy of wolves in the pack. Based on the figure 2.1, the pack is dominated by alphas, followed by beta, delta and omega. The alpha wolves are the leader and responsible for making decisions for the pack. The betas are subordinate wolves that advice alpha in decision making. The omega plays the role of scapegoat and must submit to all the other dominant wolves. In order to model the social hierarchy of wolves, the fittest solution is considered as alpha followed by beta, delta and omega respectively.

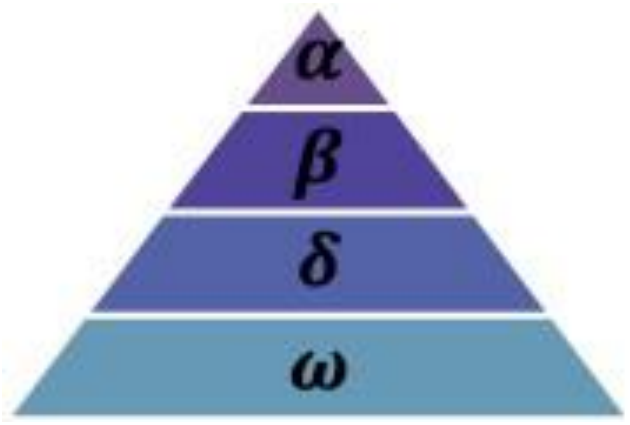

Figure 2.1: Hierarchy of grey wolf (dominance decreases from top to bottom) [18]

From the figure 2.2, the main phases of grey wolves hunting are searching for prey, encircling prey and attacking prey. In the first phase, they will move in the pack and search for suitable prey. They usually preferred ungulates, large hoofed animals such as deer and elk. As they found their prey, they may trail their pray before moving into the next phase, encircling prey. In this phase, they will encircle prey. They will encircle the prey for a few times to adjust their position accordingly to prey in order to estimate the position of the prey. They will constantly update their position randomly around the prey. Lastly, the final phase, attacking prey. They will finish hunt by attacking the prey when it stops moving. The prey position is the best solution. Figure 2.3 shows the pseudo code of Grey Wolf Optimizer.

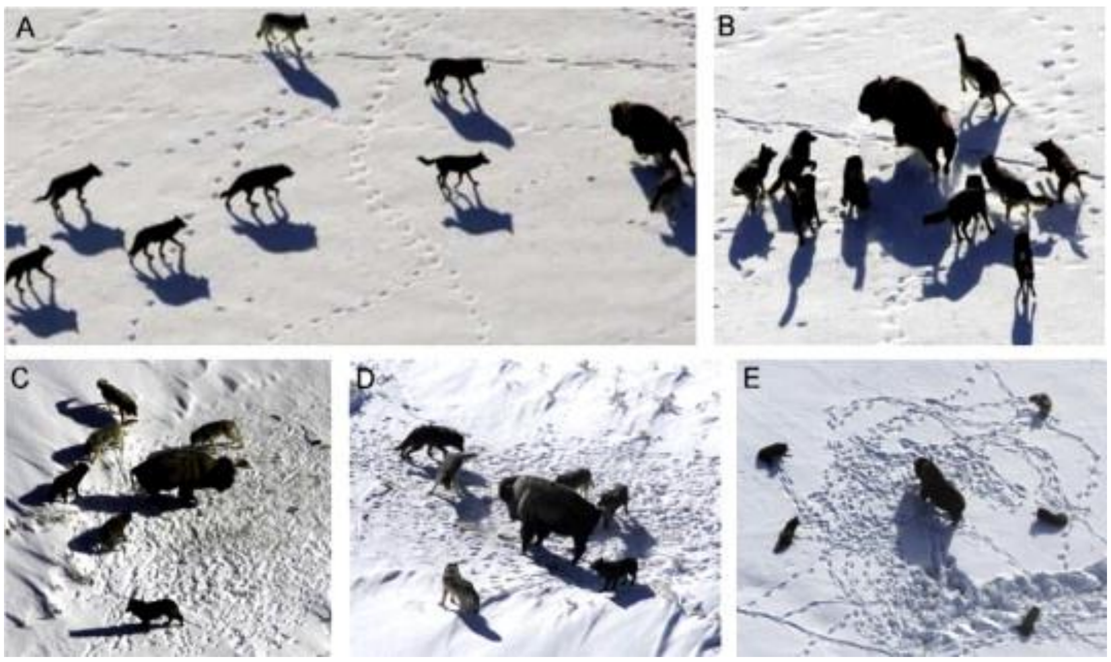

Figure 2.2: Hunting behaviour of grey wolves [18]

Grey Wolf Optimizer starts the algorithm by initializing the population of grey wolf. Then, the basic parameter of the algorithm (refer Section 3.31) will be initialized. The fitness of each search agent will be calculated. The best agent will be set as alpha, followed by beta and delta. The search agent will start to move according to hunting behaviour. For every iteration, the search agent will be updating the position. The fitness of each agents will be calculated again. The fitness by alpha, beta and delta will be updated. This operation will be repeated until the algorithm meets the maximum number of iterations.

```
Initialize the grey wolf population
Initialize a, A , and C
Calculate the fitness of each search agent
    $X_a$ = the best search agent
    $X_B$ = the second-best search agent
    $X_d$ = the third search agent
while (t < Max number of iterations)
    for each search agent
        Update the position search agent
    end for
    Update a, A and C
    Calculate the fitness of all search agents
    Update $X_a$ , $X_B$ and $X_d$
    t=t+1
end while
return $X_a$
```

Figure 2.3: Pseudo code of Grey Wolf Optimizer [19]

2.4.3 Dragonfly Algorithm (DA)

Dragonfly algorithm is another algorithm that designed by Seyedali Mirjalili [20] . Dragonfly is basically an algorithm inspired from the static and dynamic swarming behaviours of dragonflies. The modelling is designed based on the action, finding for the foods and by passing enemies when swarming dynamically or statically. Based on the figure 2.4, the former is called static swarm (feeding) while latter is called dynamic swarm (migration). In static swarming, dragonflies make small- scaled groups and fly and back and forth over a narrow are to hunt the preys. Local movements and abrupt changes in the flying path are the major characteristic of static swarm. In dynamic swarm, a huge quantity of dragonflies make the swarm for migrating in one direction over expanded distances. These two behaviours are very alike to two main phases of optimization, which are exploration and exploitation. In exploration, dragonflies make small groups and fly over different areas in a static swarm. In exploitation phases, dragonflies fly in large group and fly along one direction.

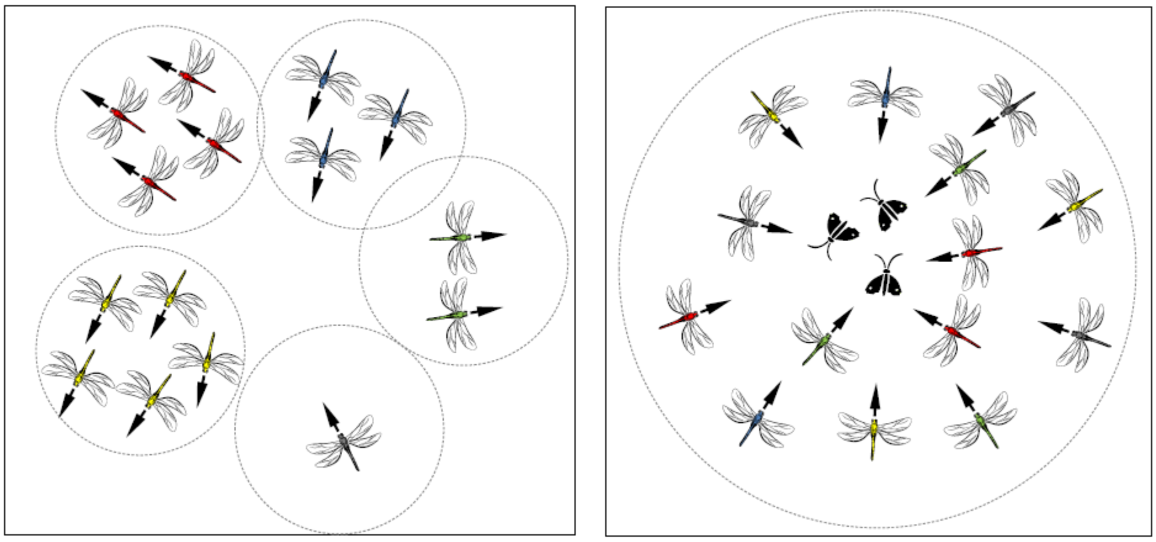

Figure 2.4: Static versus dynamic dragonfly swarm [20]

From the figure 2.5, the behaviour of swarms is following three basic principles; separation, alignment and cohesion. Separation refers to the static collision prevention of the individuals from the others in the neighbourhood. Alignment signifies velocity matching of individuals to that of other individuals in neighbourhood. Cohesion refers to the percentage leaning of individuals towards the centre of the mass of the neighbourhood. The nature of any animal is survival, so all individuals will be attracted to food sources and distracted outward enemies [21]. Figure 2.6 shows the pseudo code of Dragonfly Algorithm.

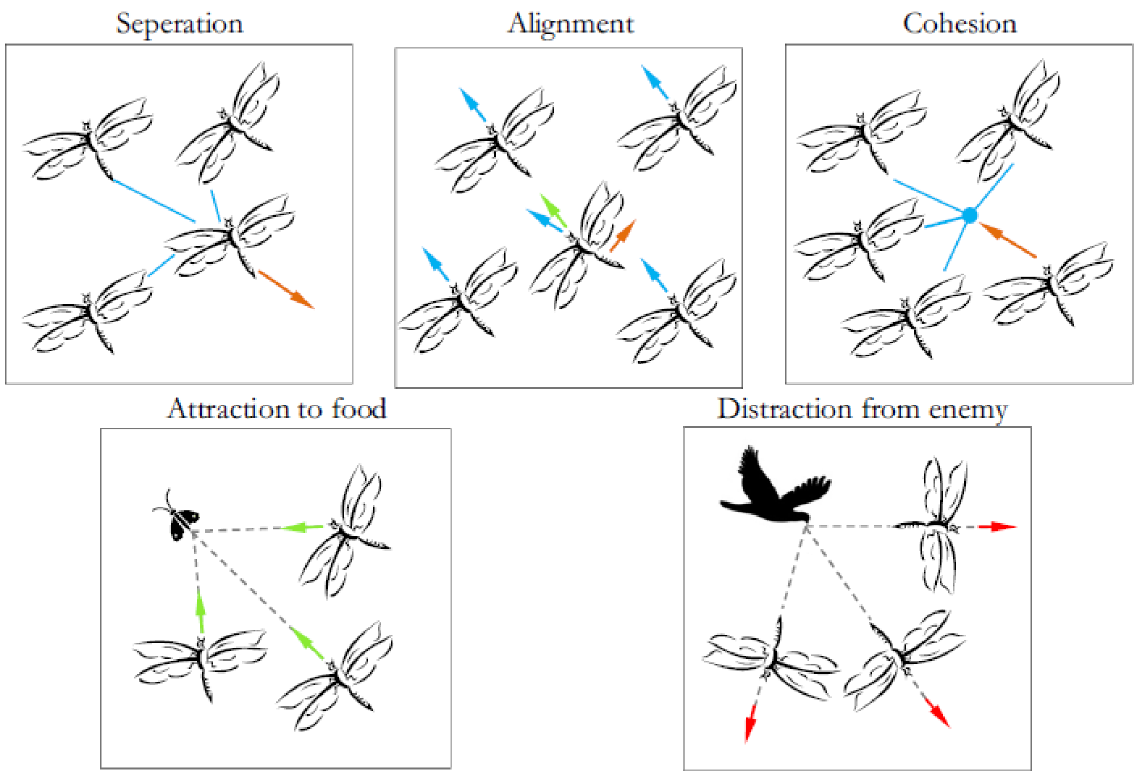

Figure 2.5: Primitive pattern of behaviour of dragonflies in a swarm [20]

The algorithm starts with initializing the dragonflies' population and step vectors. For every iteration, the fitness of each dragonfly. The position of the food source and enemy will be updated. Update the dragonfly algorithm parameter (refer Section 3.4.1). Calculate the score of each primitive pattern of behaviour. If the number of dragonfly in a neighbourhood is more than one, update vector position and velocity. If the number of dragonfly in a neighbourhood is less than one, perform update vector position. Each dragonfly update the position. This operation will be repeated until the algorithm meets the maximum number of iterations.

```
Initialize the dragonflies population $X_i$ ( i = 1,2,…n)
Initialize step vectors $\Delta X_i$ ( i = 1,2,….n)
while the end condition is not satisfied
    Calculate the objective values of all dragonflies
    Update the food source and enemy
    Update w, s, a, c, f and e
    Calculate S, A, C, F, and E using Eqs. (3.1) to (3.5)
    Update neighbouring radius
    If a dragonfly has at least one neighbouring dragonfly
        Update velocity vector using Eq. (3.6)
        Update position vector using Eq. (3.7)
    else
        Update position vector using Eq. (3.8)
    end if
    Check and correct the new positions based on the boundaries of variables
end while
```

Figure 2.6: Pseudo code of Dragonfly Algorithm (DA) [20]

2.5 Related research work on PID controller tuning

Optimization of PID controller using swarming intelligence have been done and numerous related research paper regarding it. They have been applied in different kinds of subject areas and systems. Various research papers were deeply reviewed to enhance the understanding of concept of optimization. Different algorithm has different advantages and drawbacks on certain types of problems.

2.5.1 Optimization PID controller using GWO algorithm for speed control in DC motor

A research proposed by Kaushik Ranjan Das [22]. The project is conducted to study the performance of a PID controller used in DC motor speed control, which optimizes by a nature inspired metaheuristic, GWO. Adaptive PID controller design to adopt a second order DC motor system. The optimization algorithm used in this study is Grey Wolf Optimizer to optimize the controller. Using MATLAB, the best set of PID parameters is obtained from the optimization with a step input to the DC motor to get the transient response specifications. The transient response specification such as rise time, settling time, maximum overshoot and steady state error, was recorded. These results are compared with other results from the other conventional techniques. The other techniques were Ziegler Nichols (ZN), Particle Swarm Optimization (PSO) and Artificial Bee Colony (ABC). Compared to the other techniques, GWO was able to give the best results for the rise time, settling time and peak overshoot, which the values are the minimum compare others. But, steady state error for GWO is higher than PSO and ZN.

2.5.2 Optimization of Fractional Order PID (FOPID) based temperature control of bioreactor

A project proposed by Dharmendra Tiwari. The project is focused on precise temperature control of bioreactor. The control of the process is difficult because of its highly nonlinear and dynamic nature. The proposed solution is using Fractional Order PID (FOPID) controller for the temperature control of bio reactor. Three algorithm were used to evaluate the design parameters, which are Dragonfly Algorithm, Genetic Algorithm and Simulated Annealing (SA). The performance of the design of each algorithm were evaluated [23]. The transient responses from each algorithm are observed. The result is DA converges significantly faster as compared to GA and SA. DA also minimizes the cost function slightly better compared to the other two algorithms [24]. The performance of FOPID designed by DA is better than the others two algorithms. The FOPID gave minimum value of set point tracking and disturbance rejection. The conclusion from this project that FOPID is better solved by DA rather than Genetic Algorithm and Simulated Annealing. Dragonfly algorithm is more robust and efficient.

2.6 Chapter Summary

In this chapter, it can be summarised that engineering problems especially optimisation problem need right optimisation technique to obtain optimum results. Optimisation problems need to be optimised with respect to objective function either fitness function or cost function. Based on objective function, it will be used to measure the performance of the system that need to be optimised. The right optimisation algorithm able to give an optimum results. Metaheuristics is one of the preferred optimisation technique as they are simple, flexible, free-derivation mechanism and ability to avoid local optima. One of the metaheuristics subclass is nature-inspired metaheuristic. One of the branch of the nature-inspired metaheuristic is swarming intelligence. Swarming intelligence is preferable because it is low cost, fast and robust to certain problems. The swarming algorithms used in this study are Grey Wolf Optimizer and Dragonfly Algorithm. The concept and idea of these algorithms were explained briefly. Based on the concepts of optimisation, these algorithms is applied to optimise and tune PID controller parameters.

CHAPTER 3

METHODOLOGY

3.1 Introduction

This chapter presents the optimisation of tuning PID controller parameters by implementing Grey Wolf Optimizer (GWO) and Dragonfly Algorithm (DA). In this section, the variables and parameters set in the algorithm is included as well as procedures to create basic design using MATLAB. The methods to evaluate the performance of the algorithms are also included.

3.2 Closed-Loop System and Optimisation Problem Design

Optimisation problems usually associated to specific domain of problems. Choosing a right method for optimisation is essential as different method give different output. Aside from method of optimisation, there are still a few factors that need to be considered as they also can affect the performance. They are parameter settings, robustness of objective function and scalability of constraints.

3.2.1 System Plant

The closed-loop plant response of the PID controller was modelled in MATLAB R2017b and executed using workstation powered by Intel Core i7 8700K, Base Clock @ 3.70 GHz. A closed-loop response system was designed so that the step response can be compared with that of the closed-loop system that optimised with Grey Wolf Optimizer and Dragonfly Algorithm. A plant transfer function is needed to design the closed- loop system. Three different plant transfer function were chosen from research papers. According to Pareek et al. [5], G_{p1} (System A) was used while one from a research paper, G_{p2} (system B) by Wadhwani and Verma [7]. Another one, G_{p3} (System C) from a research paper by A.A.M Zahir and S.S.N. Alhady [25]. The System A is based on the ball and hoop system [5]. The System B is based on the DC servo motor. The last system, System C is represent the brushed DC motor for cart follower system [25].The plant transfer functions, System A, System B and System C are representing as Equation 3.1, Equation 3.2 and Equation 3.3 respectively. The transfer function of PID controller, G_c (s) in Laplace form is represented by Equation 3.4 since PID controller is integrated in this design.

$$G_{p1}(s) = \frac{1}{s^4+6s^3+11s^2+6s} \tag{3.1}$$

$$G_{p2}(s) = \frac{0.01}{0.005s^3+0.06s^2+0.1001s} \tag{3.2}$$

$$G_{p3}(s) = \frac{104.9}{s^2+103.5s+2617} \tag{3.3}$$

$$G_c(s) = K_p + K_d s + \frac{K_i}{s} \tag{3.4}$$

The general transfer function, T(s) of the systems are as follows:

$$T(s) = \frac{G_p(s)G_c(s)}{1+G_p(s)G_c(s)H(s)} \tag{3.5}$$

Hence the transfer function of system A and system B are represented by Equation 3.6, Equation 3.7 and Equation 3.8.

$$T_1(s) = \frac{K_d s^2+K_p s+K_i}{s^4+6s^3+(11+K_d)s^2+(6+K_p)s+K_i} \tag{3.7}$$

$$T_2(s) = \frac{0.01K_d s^2+0.01K_p s+0.01K_i}{0.005s^3+(0.06+0.01K_d)s^2+(0.1001+0.01K_p)s+0.01K_i} \tag{3.8}$$

$$T_2(s) = \frac{104.9K_d s^2+104.9K_p s+104.9K_i}{(1+104.9K_d)s^2+(103.5+104.9K_p)s+(2617+104.9K_i)} \tag{3.9}$$

Where H(s) is a unity feedback.

The block diagram of PID controller is shown as figure 3.1. The system was constructed using Simulink. The input of this system is step input. The output was observed using scope.

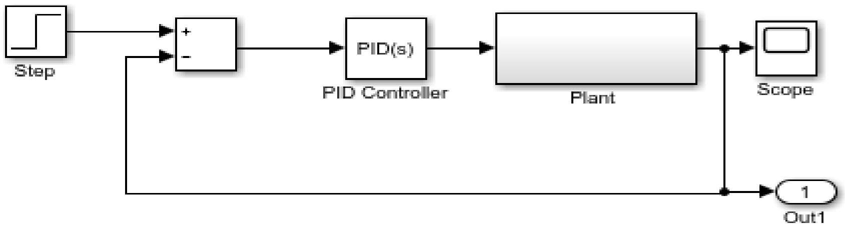

Figure 3.1 Closed-loop system with PID controller

In order to find the step response of the system, the PID controller parameters were set up. All of these parameters were selected using PID Tuner App, which one of the available application from Control System Toolbox, MATLAB. This app automatically tune those three parameter as it was fed by plant transfer function. This process was repeated for three times as for each system. The PID Tuner App implemented Ziegler Nichols auto-tune method [26]. This method is tuning PID parameters rapidly until it reaches the optimum result. For this study, the lower limit and upper limit were not set as need to identify the range of parameter for each system [27]. Table 3.1 shows the values of K_p, K_i, and K_d parameters. For system A, the highest value is 6.844, so the upper boundary was set to 10. For system B, the highest value is 18.625, so the upper boundary was set to 50. For system C, the highest value is 1662, however the upper boundary was set to 1500 because computational power and time of execution need to be considered. The dynamic performance specification for each system will be discussed in Section 4.2.1.

Table 3.1: Values of K_p, K_i, and K_d parameters

Parameters	System A	System B	System C
K_p	4.818	18.220	46.99
K_i	0.335	12.697	1662
K_d	6.844	18.625	0.28

3.2.2 Theory of Cost Function as Objective Function

This research is focussing mainly on optimising PID controller by tuning PID parameters. In other word, the optimised PID parameters should be able to produce step response, which closely similar the step input. In order to identify whether the PID parameters are optimised or not, certain criteria were observed using cost function. Optimised PID must be able to give minimal cost function [9].

The search space was defined by PID controller parameters. Three PID parameters indicate three dimensional search space. Each axis was represented by K_p, Ki, and K_d respectively. The combination of these three parameter were indicated by a single point, which represent the step response. The performance will be evaluated using cost function. The cost function consists of several element which can described the performance. The cost function is represented by Equation 3.10. The elements that can be obtained from cost function were peak overshoot, M_p, steady-state error, e_{ss}, rise time, T_r and settling time, T_s. This cost function was chosen because it is well defined what a good PID controller supposed to be. Minimal peak overshoot, minimal steady-state error, short rise time and short settling time [28].

$$F = (1 - e^{-\rho})\left(M_p + \ e_{ss}\right) + (e^{-\rho})(T_s - T_r) \qquad (3.10)$$

Where ρ is the scaling factor of designer's choice

According to Zwe-Lee, the value of ρ can be set larger than 0.7 for low overshoot and steady-state error while for short rise time and settling time, the value of ρ can be set less than 0.7 [29]. For this study, two values of ρ were chosen, which are 0.5 and 1.5.

3.3 Grey Wolf Optimizer

3.3.1 Idealization of Grey Wolf Optimizer

To simulate the grey wolf behaviour, the behaviours need to be expressed mathematically. In order to model the encircling behaviour mathematically, equations are as follow.

$$\vec{D} = |\ \vec{C}\,.\vec{X}_p(t) - \vec{X}(t)| \qquad (3.11)$$

$$\vec{X}(t+1) = \vec{X}_P(t) - \vec{A}\,.\vec{D} \qquad (3.12)$$

Where *t* indicates the current iteration, $\vec{A}$ and $\vec{C}$ are coefficient vectors, $\vec{X}_p$ is the position vector of the prey, and $\vec{X}$ indicates the position vector of a grey wolf. The vectors $\vec{A}$ and $\vec{C}$ are calculated as follows.

$$\vec{A} = 2\vec{a} \,. \vec{r}_1 - \vec{a} \tag{3.13}$$

$$\vec{C} = 2 \,. \vec{r}_2 \tag{3.14}$$

Where components of $\vec{a}$ are linearly decreased from 2 to 0 over the course of iterations and r_1 , r_2 are random vectors in [0, 1].

In order to model the hunting behaviour mathematically, alpha is assumed to have better knowledge about the location of prey followed by beta and delta. So, the first 3 best solutions obtained are saved and oblige the others agents. The equations are as follow.

$$\vec{D}_a = |\vec{C}_1 . \vec{X}_a - \vec{X}|, \vec{D}_B = |\vec{C}_2 \,. \vec{X}_B - \vec{X}|, \vec{D}_d = |\vec{C}_3 \,. \vec{X}_d - \vec{X}| \tag{3.15}$$

$$\vec{X}_1 = \vec{X}_a - \vec{A}_1 . (\vec{D}_a), \vec{X}_2 = X_B - A_2 \,. (D_B), \vec{X}_3 = \vec{X}_d - \vec{A}_3 \,. (\vec{D}_d) \tag{3.16}$$

$$\vec{X}(t+1) = \frac{\vec{X}_1 + \vec{X}_2 + \vec{X}_3}{3} \tag{3.17}$$

Alpha, beta and delta estimate the position of the prey, and the other agents update their position randomly around the prey.

In order to model the attacking behaviour mathematically, the value of $\vec{a}$ in order to mimic the grey wolves approaches the prey. $\vec{A}$ is a random value in the interval [-2a,2a] where a is decreased from 2 to 0 over the course of iteration. When random values of $\vec{A}$ is between [1, 1], it indicates the wolves attack the prey.

In order to mathematically model divergence, $\vec{A}$ is random values greater than 1 or less than 1 to oblige the agent to diverge from the prey. This forces the agents to search globally. When $\vec{A}$ less than 1, the wolves will attack the prey. But, if the values of $\vec{A}$ is larger than 1, the wolves will diverge from the prey.

Another component is $\vec{C}$, which contains random values in [0, 2]. This component provides random weight for prey to emphasize (C >1) or deemphasize (C<1) the effect of prey in defining the distance in equation (3.1). This is important in order to create a more random behaviour through optimization and avoid local optima. The prey position is the best solution. Grey Wolves Optimizer uses the basic parameters listed in table 3.2. Table 3.2 shows the basic parameters that exist in Grey Wolf Optimizer.

Table 3.2: Basic parameters in Grey Wolf Optimizer (GWO)

Parameters	Symbols
Number of search agents (grey wolf)	SearchAgents_no
Parameter for equation 2.4	C
Parameter for equation 2.3	a
Cost function	fobj
Number of variables	dim
Maximum number of iterations	Max_iterations
Lower boundary for each variable	lbn
Upper boundary for each variable	ubn
Number of boundaries	Boundary_no
Upper boundary	ub
Lower boundary	lb

In Grey Wolf Optimizer, *SearchAgents_no* refer to number of grey wolf in the population. *C* and *a* parameter refer to equation 2.4 and 2.3. *fobj* refer the declared cost function. *dim* refer to the number of variables of the search space. *Max_iterations* refers to maximum number of iterations that the GWO will be executed. *lbn* and *ubn* refer the lower upper boundary for each variable. *Boundary_no* specify number of boundaries. Figure 3.2 show the flowchart of Grey Wolf Optimizer (GWO). The flowchart show the flow on how the Grey Wolf Optimizer (GWO) works.

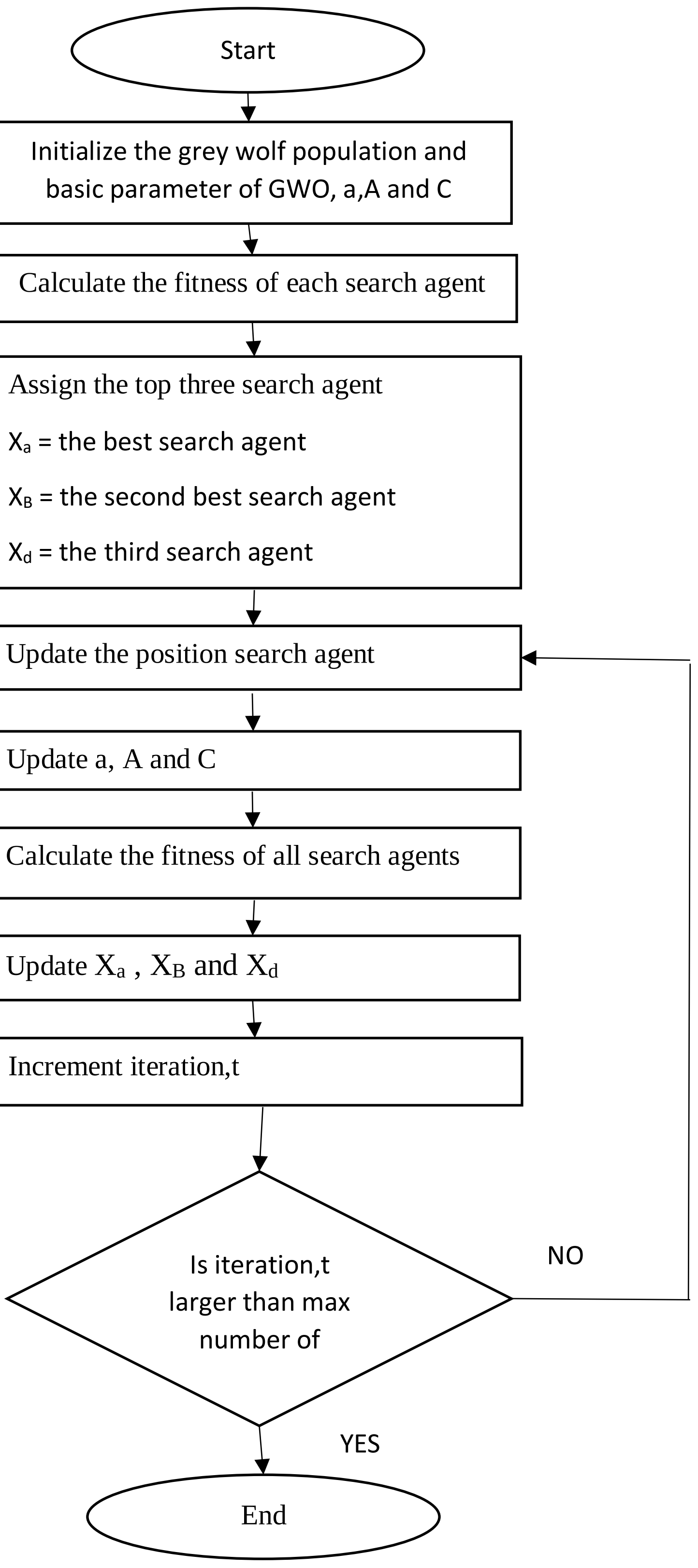
Start
Initialize the grey wolf population and basic parameter of GWO, a,A and C
Calculate the fitness of each search agent
Assign the top three search agent
X_a = the best search agent
X_B = the second best search agent
X_d = the third search agent
Update the position search agent
Update a, A and C
Calculate the fitness of all search agents
Update X_a , X_B and X_d
Increment iteration,t
Is iteration,t larger than max number of
NO
YES
End

Figure 3.2: Flowchart of Grey Wolf Optimizer (GWO)

Refer to the flowchart (figure 3.2), the algorithm start with initialize the grey wolf population and basic parameter of GWO, which are *A* and *C*. The fitness of each search agent will be calculated. The top three search will be assigned as alpha, followed by beta and delta. Alpha is the fittest solution. The rest of the search agent will be assigned as omega. The search agents will updated the position from prey. Update *a, A,* and *C.* the fitness of all search agents will be calculated. The position of alpha, beta and delta. Increment the value of current iterations. The algorithm will check the stopping criteria, which is maximum number of iteration. If the number of the maximum of the iteration is equal to the number of current iteration, the algorithm will stop. If not, the operation will be continued with update the position of each search agent [30].

3.3.2 Setting of Grey Wolf Optimizer

The algorithm has its own specialised control parameters that can be tuned based on the optimisation problem. This parameters is like an interface of the algorithm. The control parameters are listed in Table 3.3.

Table 3.3: Set of control parameters (GWO)

Description	Parameters	System A	System B	System C
Number of search agents (grey wolf)	SearchAgents_no	30	30	30
Number of variables	dim	3	3	3
Maximum number of iterations	Max_iterations	200	200	200
Upper boundary	ub	10	50	1500
Lower boundary	lb	0	0	0

Based on table 3.3, the number of grey wolf was set to 30. The number of variables, *dim* was set to 3 because of three PID controller parameters, K_p, K_i and K_d. The value of upper boundary, ub for each system was set differently. For system A, upper boundary was set to 10. For system B, it was set to 50. For system C, it was set to 1500. The value of upper boundary was set in such way because each system is different. Referring to the Table 3.1, the value of K_p, K_i and K_d were different for each system. The value of upper boundary need to set in a way that it was bigger than these values, so the algorithm can cover these points when executed except for System C. As for System C, it was capped to 1500 because if the value of upper boundary is set higher than that, it need to take more times and computational power to executed. Lower boundary for each system was set to 0 as the value of K_p, K_i and K_d need to be positive. Max iterations was set to 200. The reason why this value was chosen discussed in Section 4.4.

All of these values were set up by declaring all those values in the beginning of file initialisation.m. The objective function was saved as tracklsq.m file. This file was interfaced with the m-file of Grey Wolf Optimizer. So, the variables were directly read by algorithm.

3.4 Dragonfly Algorithm

3.4.1 Idealization of Dragonfly Algorithm

To simulate the dragonfly behaviour, their behaviour need to be expressed mathematically. From the figure 3.3, the behaviour of swarms is following three basic principles; separation, alignment and cohesion. Separation refers to the static collision prevention of the individuals from the others in the neighbourhood. The separation behaviour is mathematically represented as follows:

$$S_i = \sum_{j=i}^{N} X - X_j \qquad (3.18)$$

Where X is the position of the current individual, X_j presents the position j-th neighbouring individual and N is the number of neighbouring individuals.

Alignment signifies velocity matching of individuals to that of other individuals in neighbourhood. The alignment behaviour is mathematically represented as follows:

$$A_i = \frac{\sum_{j=1}^{N} V_j}{N} - X \qquad (3.19)$$

Where V_j shows the velocity of the j-th neighbouring individual.

Cohesion refers to the percentage leaning of individuals towards the centre of the mass of the neighbourhood. The cohesion behaviour is mathematically represented as follows: