# Designing of Optimal position Control systems using intelligent optimization techniques

**By**

**Shreya Naskar**

Class Roll No**. -** 001711103003

Reg. No. -140933 of 2017-18

Examination Roll No. – M4IEE19003

Thesis submitted in partial fulfilment of the requirement for the Degree of Master of Technology

In the Department of Instrumentation and Electronics Engineering, Faculty of Engineering & Technology

Under supervision of

**Prof. Rajanikanta Mudi**

Jadavpur University

Kolkata -700032

2019

## *Certificate of Recommendation*

I hereby recommend that the thesis titled "**Designing of Optimal position Control systems using intelligent optimization techniques**" carried out under my supervision by Shreya Naskar (Registration no. 140933 of 2017-18 ) may be accepted in partial fulfilment of the requirement for the degree of "Master of Technology in Instrumentation & Electronics Engineering' of Jadavpur University.

...............................................
(Prof. Rajanikanta Mudi)

Thesis Supervisor
Department of
Instrumentation & Electronics Engineering
Jadavpur University
Salt Lake Campus
Kolkata- 700098

.............................................
(Prof. Runu Banerjee Roy)

Head of Department
Instrumentation & Electronics Engineering
Jadavpur University
Kolkata- 700098

..............................................
(Prof. Chiranjib Bhattacharjee)

Dean
Faculty of Engineering and Technology
Jadavpur University
Kolkata- 700032

## ***Certificate of Approval***

(*Only in case the thesis is approved)

The thesis at instance is hereby approved as a credible study of an Engineering subject carried out and presented in a manner satisfactory to warrant its acceptance as a prerequisite to the degree for which it has been submitted. It is understood that by this approval the undersigned does not necessarily endorse or approve any statement made, opinion expressed or conclusion drawn therein, but approve this thesis for the purpose for which it is submitted.

| | |
|---|---|
| ----------------------------------- | ----------------------------------- |
| Signature of the examiner | Signature of the supervisor |

## *Declaration of originality and compliance of academic ethics*

I hereby declare that this thesis contains literature survey and original research work by me, as a part of my Master of Technology in Instrumentation & Electronics engineering studies.

All information in this document have been obtained and presented in accordance with academic rules and ethical conduct.

I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.

| | |
|---|---|
| Name: | Shreya Naskar |
| Roll Number: | 001711103003 |
| Thesis title: | Designing of Optimal position Control systems using intelligent optimization techniques |

Signature with Date:

# *Acknowledgement*

I would like to thank a lot of people who gave me unending support and inspiration from the beginning and without whose help this thesis and project would not have been completed.

First and foremost, I would like to thank my guide Prof. Rajanikanta Mudi, whose suggestions, guidance and encouragement have helped me immensely in understanding the subject.

I would also like to thank all my classmates and the staffs of the department for the constant support and help they provided me all the time.

Regards
Shreya Naskar

# CONTENTS

# LIST OF FIGURES

# LIST OF TABLES

# Chapter 1

Introduction & Scope of the Thesis

## 1.1 Introduction

A control system can be thought of as any system where additional hardware is added to regulate the behaviour of a dynamic system. Control systems can either be open loop or closed loop. A closed loop system implies the use of feedback in the system. We will see that using feedback allows us more freedom to specify the desired output behaviour of the system. For continuously modulated control, a feedback controller is used to automatically control a process or operation. The control system compares the value or status of the process variable (PV) being controlled with the desired value or set point (SP), and applies the difference as a control signal to bring the process variable output of the plant to the same value as the set point. Control Engineering is concerned with techniques that are used to solve the following six problems in the most efficient manner possible.

(a)The identification problem: to measure the variables and convert data for analysis.
(b)The representation problem: to describe a system by an analytical form or mathematical model
(c)The solution problem: to determine the above system model response.
(d)The stability problem: general qualitative analysis of the system.
(e)The Controller design problem: modification of an existing system or develop a new one.
(f)The optimization problem: from a variety of design to choose the best.

In studying control systems, one must be able to model dynamic systems in mathematical terms and analyze their dynamic characteristics. A mathematical model of a dynamic system is defined as a set of equations that represents the dynamics of the system accurately, or at least fairly well. The mathematical model is not unique to a given system. A system may be represented in many different ways and, therefore, may have many mathematical models, depending on one's perspective. The dynamics of many systems, whether they are mechanical, electrical, thermal, economic, biological, and so on, may be described in terms of differential equations. Such differential equations may be obtained by using physical laws governing a particular system—for example, Newton’s laws for mechanical systems and Kirchhoff’s laws for electrical systems. We must always keep in mind that deriving reasonable mathematical models is the most important part of the entire analysis of control systems [1].

## 1.2 Process Control and Basic Terminology

A collection of components that interact with one another and with their environment is known as system. A control system is a collection of components that is designed to drive a given system (plant) with a given input to a desired output [1].

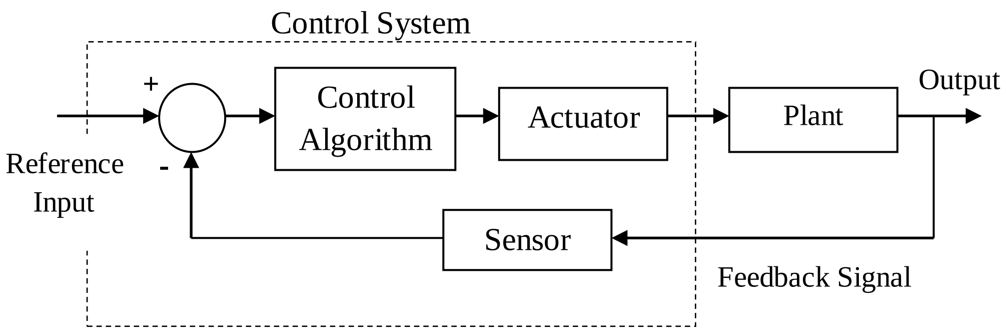

**Fig 1.1:** Process control block diagram with feedback loop

In block diagrams of SISO systems, a solid line represents a single scalar signal. In MIMO systems, a single line may represent multiple signals. The circle in the figure represents a summing junction, which combines its inputs by addition or subtraction depending on the + and – signs next to each input.

The contents of the dashed box in Fig 1.1 are the control system components. The controller inputs are the reference input (also called a set point) and the plant output signal (measured by the sensor), which is used as feedback. The controller output is the actuator signal that drives the plant.

A block in a diagram can represent something as simple as a constant value that multiplies the block input, or as complex as a nonlinear system with unknown mathematical representation.

Figure 1.2 is a block diagram of a simple linear feedback control system. Lower case characters identify the signals in this system.

- *r* is the reference input, also called the set point.
- *e* is the error signal, computed by subtracting the sensor measurement from the reference input.
- *y* is the system output, which is measured and used as the feedback signal.

The blocks in the diagram represent linear system components. Each block can represent dynamic behaviour with any degree of complexity as long as the requirement of linearity is satisfied.

- $G_c$ is the linear controller algorithm.
- $G_p$ is the linear plant model (including actuator dynamics.)
- $H$ is a linear model of the sensor, which can be modelled as a constant such as 1 if the sensor dynamics are negligible.

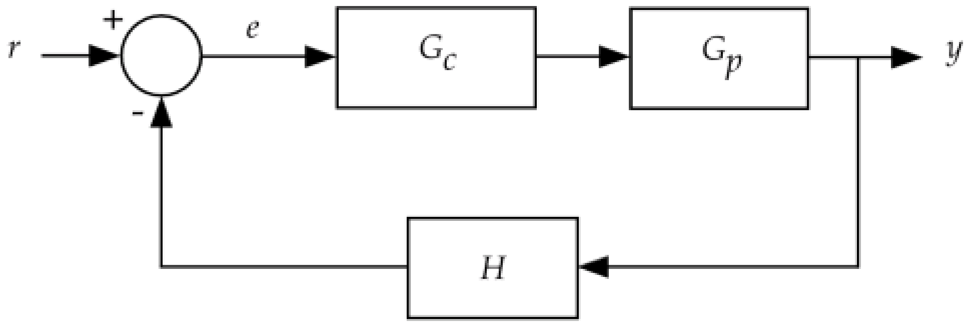

**Fig 1.2:** Linear feedback control system

The fundamental rule of block diagram algebra states that the output of a block equals the block input multiplied by the block transfer function [2]. Applying this rule twice results in Eq. 1.1. In words, Eq. 1.1 says the system output $y$ is the error signal $e$ multiplied by the controller transfer function $G_c$, and then multiplied again by the plant transfer function $G_p$.

$$y = (eG_c)G_p \tag{1.1}$$

Block diagram algebra obeys the usual rules of algebra. Multiplication and addition are commutative, so the parentheses in Eq. 1.1 are unnecessary. The error signal $e$ is the output of a summing junction subtracting the sensor measurement from the reference input $r$. The sensor measurement is the system output $y$ multiplied by the sensor transfer function $H$. This relationship appears in Eq. 1.2.

$$e = r - yH \tag{1.2}$$

Substituting Eq. 1.2 into Eq. 1.1 and rearranging algebraically results in Eq. 1.3.

$$\frac{y}{r} = \frac{G_cG_p}{1+G_cG_pH} \tag{1.3}$$

Eq. 1.3 is a transfer function giving the ratio of the system output to its reference input. This form of system model is suitable for use in numerous control system analysis and design tasks.

Using the relation of Eq. 1.3, the entire system in Fig. 1.2 can be replaced by the equivalent system shown in Fig. 1.3. Remember, these manipulations are only valid when the components of the block diagram are all linear.

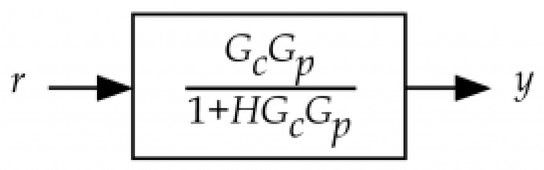

**Fig 1.3:** Equivalent system to the system shown in Fig. 1.2.

### 1.2.1 Process

Any equipment that serves the targeted physical or chemical operation of the plant is termed as a process. Reactors, separators, exchangers, pressure vessels, tanks, etc. are examples of a process. Typically these processes are connected in a logical fashion and the output of one process becomes input to the other [3]. Any disturbance or malfunction of one process may affect other processes in the downstream side (and upstream too, in case recycle streams are used). Process variables are primarily pressure, temperature, flow rate, level, composition, etc. From the process control perspective, it is crucial to study how the changes in one process variable affect the other, so that an educated measure of control action on one variable can be taken in order to maintain the other [4].

### 1.2.2 Measuring Instruments or Sensors

The success of any feedback control operation depends largely on accurate measurement of process variables through appropriate sensors. There are a large number of commercial sensors available in the market. They differ in their measuring principle(s) and or their construction characteristics [4].

### 1.2.3 Controllers [4]

A controller is basically a mathematical function block that reads the error between desired set point and the measured output and then computes the corrective action for the manipulated input that would steer process towards the desired set point. There are three basic types of feedback controllers which are widely used in the industry.

- Proportional (P) controller

- Proportional Integral (PI) controller

- Proportional Integral Derivative (PID) controller

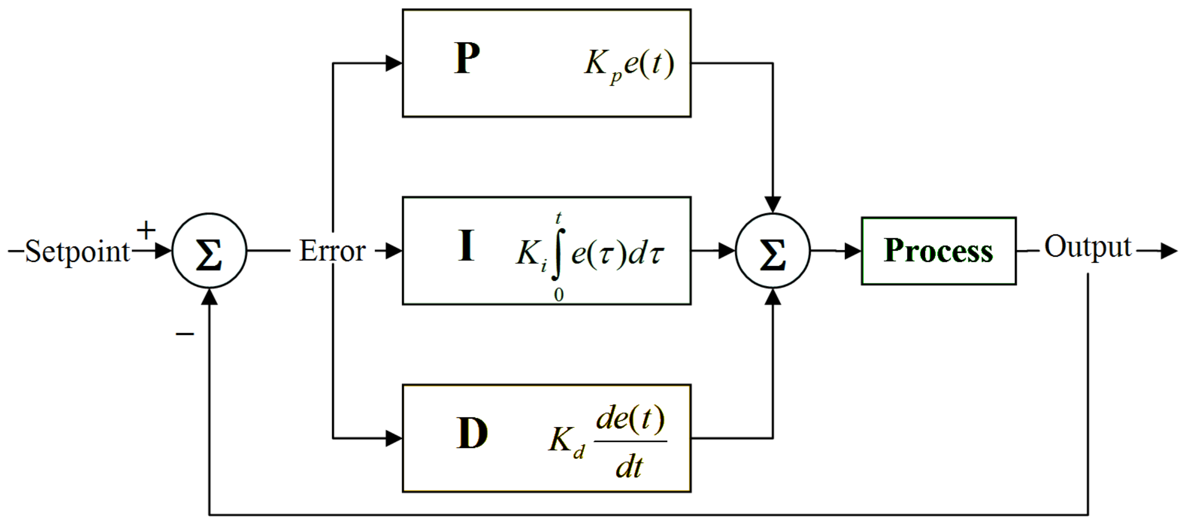

**Fig 1.4:** Physical Realization of PID Controller

Let us discuss each one separately.

### 1.2.3.1 Proportional Controllers

The actuating output of a P controller is proportional to the error between the set point and process output. Higher the error, higher will be the control action. The control law is given as:

$$C(t) = K_c \; e(t) + C_s \tag{1.4}$$

where $K_c$ is called the gain of the controller and $C_s$ is the bias signal. When error signal is zero (i.e., the process output reaches its desired set point), the control signal $C(t)$ stabilizes at its bias value $C_s$. The deviation form of actuating signal is

$$\bar{C}(t) = C(t) - C_s = K_c\, e(t) \tag{1.5}$$

Hence the transfer function of the proportional controller is

$$\frac{\overline{C}(s)}{e(s)} = G_c = K_c \tag{1.6}$$

The proportional controller is also termed as "Gain" controller. Equivalent representation of proportional gain is proportional band. It is the amount of change in error that will cause the control action to go from full OFF to full ON. The amount of change in error is calculated as a percentage of full-scale error,

$$PB\ (\text{in }\%) = \frac{100}{K_C} \tag{1.7}$$

e.g. consider a level controller acting on a tank where we measure the level from bottom to top as 0 to 100%. A control valve on the outlet of the tank maintains the level in the tank. The PB is defined as the range of level over which the control valve will go from fully closed to fully open.

### 1.2.3.2 Proportional Integral Controllers

The actuating output of a PI controller is given as:

$$C(t) = K_c(e(t) + \frac{1}{\tau}\int_0^t e(t)dt) + C_s \tag{1.8}$$

where $\tau_1$ is the integral time constant (or the reset time) in minutes. The PI controller not only actuates on the basis of current error, *e(t)* but it also accounts for the history of all the past errors that has been encountered since the control action has started. From Eq.1.8, the transfer function of the PI controller is

$$G_c = \frac{\overline{C(s)}}{e(s)} = K_c\ (1 + \frac{1}{\tau s}) \tag{1.9}$$

In industrial lingo, the PI Controller is also termed as “Gain-Preset” controller. At this point, it is worth explaining the significance of the term reset. Suppose the error between desired set point and process output changes by a constant step of magnitude *e (t).* The effect of integral term of Eqn. 1.8 after every $\tau$ minutes is given as

$$K_c(\frac{1}{\tau}\int_0^t e(t)) = K_c\ (\frac{e}{\tau}\int_0^\tau dt) = K_c\ (\frac{e}{\tau}\tau) = K_c\, e \tag{2.0}$$

In other words, the integral action repeats the response of the proportional action every $\tau$ minutes and “resets” itself for an integral action. Sometimes the controllers are calibrated in terms of reciprocal of reset time, $1/\tau$ (repeats per minute). This is known as reset rate.

The reset term causes the control action changing as long as there exists a non-zero error in the system. Often this error cannot be eliminated quickly and given enough time, they produce larger values for integral terms. Such situation is often observed when the system undergoes a large change in set point (say a positive change) and the integral term accumulates a significant error during the rise. This condition is termed as Integral Windup. The control action in turn keeps on increasing until it reaches the control valve saturation (i.e. control valve fully open or fully closed). Even if the error changes its sign (as the process output overshoots the desired set point), this accumulated error has to unwind completely before control action is reversed. Various measures can be taken to address the issue of integral windup such as:

• Re-initializing the integral action to a desired value

• Increasing the set point in a suitable ramp (rather than a single step jump)

• Disabling the integral action until the process output enters the controllable region

• Preventing the integral term from accumulating above or below pre-determined bounds

### 1.2.3.3 Proportional Integral Differential Controllers

The actuating output of a PID controller is given as:

$$C(t) = K_c\left(e(t) + \frac{1}{\tau}\int_0^t e(t)dt + \tau_D\frac{de(t)}{dt}\right) + C_s \qquad (2.1)$$

where $\tau_D$ is the derivative time constant (or the react time) in minutes. The PID controller not only actuates on the basis of current and past errors but it also anticipates the error in immediate future and applies an additional control action which is proportional to the current rate of change of error. Hence the transfer function of the PID controller is

$$G_c = \frac{\overline{C(s)}}{e(s)} = K_c\left(1 + \frac{1}{\tau s} + \tau_D s\right) \qquad (2.2)$$

The PID Controller is also termed as “Gain-Reset-Preact” controller.

The major drawback of a PID controller is that for a noisy response in a process, the controller can erroneously actuate a high derivative control action.

## 1.3 Performance Specifications

One of the first steps in the control system development process is the definition of a suitable set of system performance specifications. Performance specifications guide the design process and provide the means for determining when a controller design is satisfactory. Controller performance specifications can be stated in both the time domain and in the frequency domain [5].

Time domain specifications usually relate to performance in response to a step change in the reference input. An example of such a step input is instantaneously changing the reference input from 0 to 1. Time domain specifications include, but are not limited to, the following parameters:

- Rise time from 10% to 90% of the commanded value, $t_r$.
- Time to peak magnitude, $t_p$.
- Peak magnitude, $M_p$. This is often expressed as the peak percentage by which the output signal overshoots the step input command.
- Settling time to within some fraction (such as 1%) of the step input command value, $t_s$.

Examples of these parameters appear in Fig. 1.5. This figure shows the response of a hypothetical plant plus controller to a step input command with an amplitude of one. The time axis zero location is the instant of application of the step input.

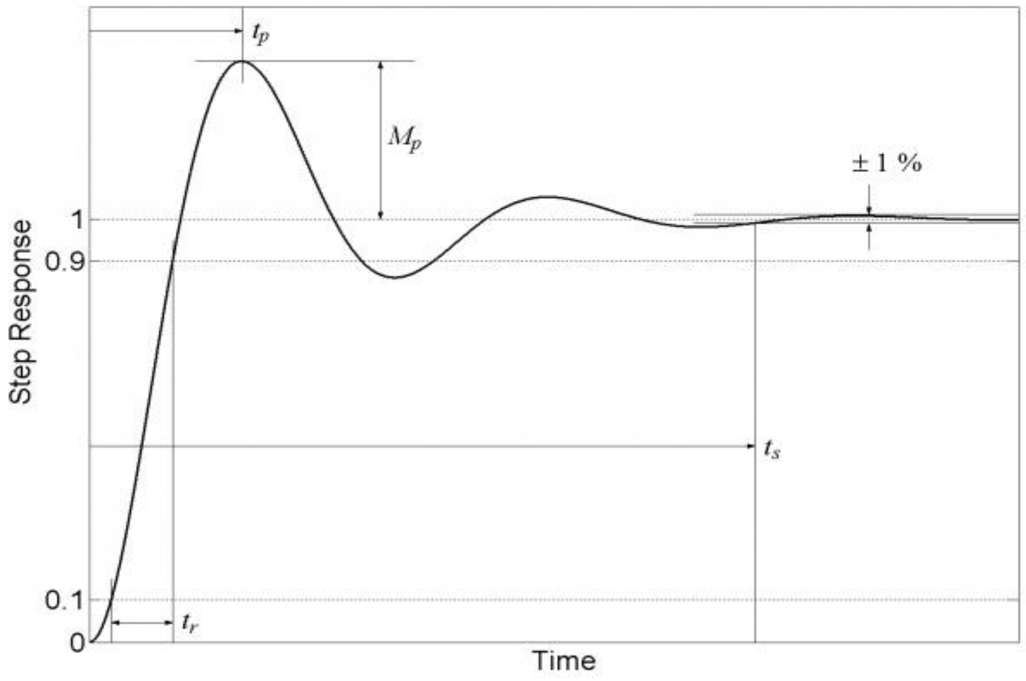

**Fig 1.5** Time Domain control system performance parameters

The step response in Fig. 1.5 represents a system with a fair amount of overshoot (in terms of $M_p$) and oscillation before converging to the reference input. Sometimes the step response has no overshoot at all. When no overshoot occurs, the $t_p$ parameter becomes meaningless and $M_p$ is zero.

Tracking error is the error in the output that remains after the input function has been applied for a long time and all transients have died out. It is common to specify the steady-state controller tracking error characteristics in response to different commanded input functions such as steps, ramps, and parabolas.

Here are some example specifications of tracking error in response to different input functions:

- Zero tracking error in response to a step input.
- Less than 'X' tracking error magnitude in response to a ramp input, where *X* is some nonzero value.

In addition to the time domain specifications discussed above, performance specifications can be specified in the frequency domain. The controller reference input is usually a low frequency signal, while noise in the sensor measurement used by the controller often contains high frequency components. It is normally desirable for the control system to suppress the high frequency components related to sensor noise while responding to changes in the reference input. Performance specifications capturing these low and high frequency requirements would look similar to these:

- For sinusoidal reference input signals with frequencies below a cut-off point, the amplitude of the closed loop (plant plus controller) response must be within *X*% of the commanded amplitude.
- For sinusoidal reference input signals with frequencies above a higher cut-off point, the amplitude of the closed loop response must be reduced by at least *Y*%.

In other words, the frequency domain performance requirements given above say that the system response to expected changes in the reference input must be acceptable while simultaneously attenuating the effects of noise in the sensor measurement. Looked at in this way, the closed loop system exhibits the characteristics of a low pass filter.

## 1.4 System Stability

Stability is a critical issue throughout the control system design process. A stable controller produces appropriate responses to changes in the reference input. If the system stops responding properly to changes in the reference input and does something else instead, it has become unstable.

Fig. 1.6 shows an example of unstable system behavior. The initial response to the step input overshoots the commanded value by a large amount. The response to that overshoot is an even larger overshoot in the other direction [6].

This pattern continues, with increasing output amplitude over time. In a real system, an unstable oscillation like this grows in amplitude until some nonlinearity such as actuator saturation (or a system breakdown) limits the response.

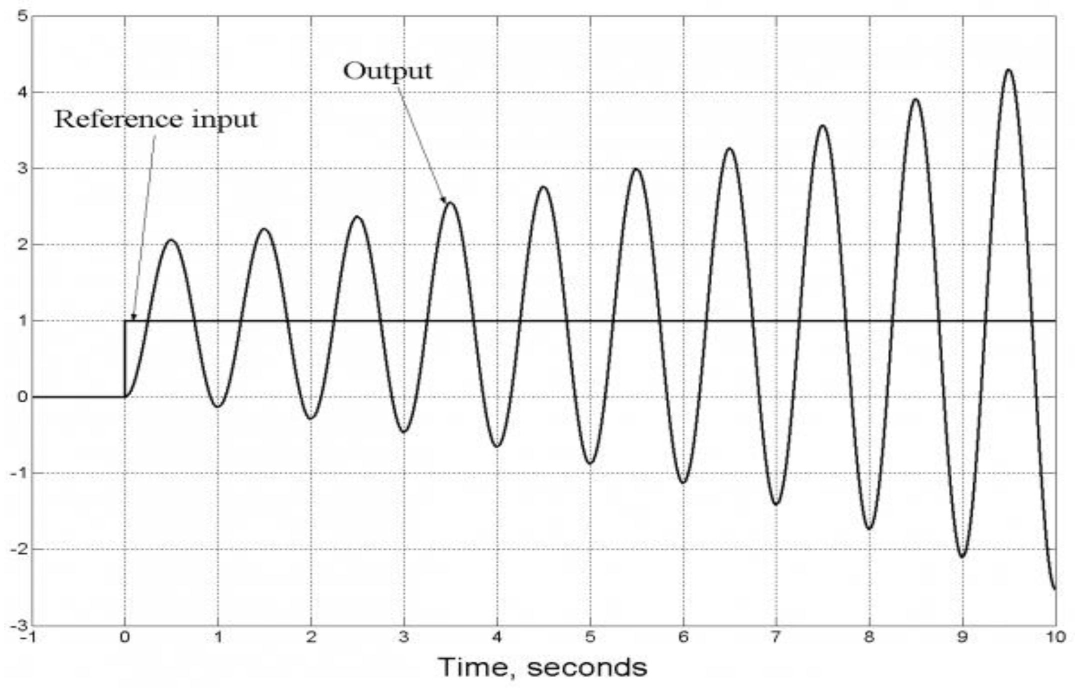

**Fig 1.6:** System with an unstable oscillatory response

In addition to achieving a bare minimum degree of stability, a control system must possess a degree of robustness. A robust controller can tolerate limited changes to the parameters of the plant and its operating environment while continuing to provide satisfactory, stable performance. For example, an automotive cruise control must maintain the desired vehicle speed by adjusting the throttle position in response to changes in road grade (an environmental change.) The cruise control must also perform properly whether or not the vehicle is pulling a trailer (a change in system parameters).

Determining the allowable ranges of system and environmental parameter changes is part of the controller specification and design process. To demonstrate robustness, the designer must evaluate controller stability under worst-case combinations of expected plant and environment parameter variations [6]. For each combination of parameter values, a robust controller must satisfy all of its performance requirements. When working with linear models of plants and controllers it is possible to precisely determine whether a particular plant and controller form a stable system. If no mathematical model for the plant exists, stability can only be evaluated by testing the plant and controller under a variety of operating conditions.

## 1.5 Control System Testing

Testing is an integral part of the control system design process. Many of the design methods rely on the use of a linear plant model. Creating a linear model always involves approximation and simplification of the true plant behavior. The implementation of a controller using an embedded processor introduces nonlinear effects such as quantization. As a result, both the plant and the controller contain nonlinear effects that are not accounted for in a linear control system design.

The ideal way to demonstrate correct operation of the nonlinear plant and controller over the full range of system behavior is by performing thorough testing with an actual plant. This type of system-level testing normally occurs late in the product development process when prototype hardware becomes available. Problems found at this stage of the development cycle tend to be very expensive to fix.

Because of this, it is highly desirable to perform thorough testing at a much earlier stage of the development cycle [5]. Early testing enables discovery and repair of problems when they are relatively easy and inexpensive to fix. However, testing the controller early in the product development process may not be easy if a prototype plant does not exist on which to perform tests.

System simulation provides a solution to this problem. A simulation containing detailed models of the plant and controller is extremely valuable for performing early-stage control system testing. This simulation should include all relevant nonlinear effects present in the actual plant and controller implementations. While the simulation model of the plant must necessarily be a simplified approximation of the actual system, it should be a much more authentic representation than the linear plant model used in the controller design.

When using a simulation in a product development process, it is imperative to perform thorough simulation verification and validation [6].

- Verification demonstrates the simulation has been implemented correctly according to its design specifications.
- Validation demonstrates that the simulation accurately represents the behaviour of the simulated system and its environment for the intended purposes of the simulation.

The verification step is relevant for any software development process, and simply shows that the software performs as its designers intended. In simulation work, verification can occur in the early stages of a product development project. It is possible to perform verification for a simulation of a system that does not yet exist. This consists of making sure that the models used in the simulation are correctly implemented and produce the expected results.

Verification allows the construction and application of a simulation in the earliest phases of a product development project.

Validation is a demonstration that the simulation models the embedded system and the real world operational environment with acceptable accuracy. A standard approach for validation is to use the results of system operational tests for comparison against simulation results. This involves running the simulation in a scenario that is identical to a test that was performed by the actual system in a real world environment. The results of the two tests are compared and the differences are analyzed to determine if they represent significant deviations between the simulation and the actual system.

A drawback of this approach to validation is that it cannot happen until a complete system prototype is available. Even when a prototype does not exist, it may be possible to perform validation at an earlier project phase at the component and subsystem level. You can perform tests on those system elements in a laboratory environment and duplicate the tests with the simulation. Comparing the results of the two tests provides confidence in the validity of the component or subsystem model.

## 1.6 Computer-Aided Control System Design

The classical control system analysis and design methods were originally developed and have been implemented for years as techniques that rely on hand-drawn sketches. While this approach leads to a level of design intuition, it takes significant time and practice to develop the necessary skills.

Since it intends to rapidly apply a variety of control system design techniques, automated approaches will be emphasized rather than manual methods [5]. Several software packages are commercially available that perform control system analysis and design functions as well as complete nonlinear system simulation.

- MATLAB Control System Toolbox. This is a collection of algorithms that implement common control system analysis, design and modelling techniques. It covers classical design techniques as well as modern state-space methods. This is an add-on to the MATLAB product, which integrates mathematical computing, visualization, and a programming language to enable the development and application of sophisticated algorithms to large sets of data.

Here we have used MATLAB, the Control System Toolbox, and other MATLAB add-on products to demonstrate a variety of control system modelling, design, and simulation techniques. These tools provide efficient, numerically robust algorithms to solve a variety of control system engineering problems. The

MATLAB environment also provides powerful graphics capabilities for displaying the results of control system analysis and simulation procedures.

## 1.7 Method Based on Performance Criteria

It is based on minimizing an appropriate performance criterion, either for optimum regularity or for optimum servo performance. Based on the minimum ITAE value, settings for PID Controllers are derived. These settings are expected to provide desirable performance for time delay to time constant ratio from 0.1 to 1. In 1993 Zhuang and Atherton suggested PI and PID settings based on minimization of ISE, ISTE and ITAE [7]. For the process model, repeated optimization is carried out for different values of time delay to time constant ratio.

The ISE criterion penalises large errors, while the ITAE criterion penalises error that persists for longer periods of time. In general, the ITAE criterion is the preferred criterion in practise, because it usually results in the most conservative controller settings [7]. By contrast, the ISE criterion provides the most aggressive setting, while the IAE criterion tends to produce controller settings that are between those for the ITAE and ISE criteria.

In all the above tuning rules, the optimum controller settings are different for set point changes in comparison to those for step load disturbances. In general, the controller settings for set point changes are more conservative. The performance criteria for the controller chosen here is ITAE.

## 1.8 Literature Survey

Feedback control systems measure attributes of the system being controlled and use that information to determine the control actuator signal [7]. Feedback control provides superior performance compared to open loop control when environmental or system parameters change. The system to be controlled is called a plant.

The two fundamental steps in control system design are:

1. Specify the controller structure.
2. Determine the value of the design parameters within that structure.

The control system design process usually involves the iterative application of these two steps. In the first step, a candidate controller structure is selected. In the second step, a design method is used to determine suitable parameter values for that structure. If the resulting system performance is inadequate, the cycle is repeated with a new, usually more complex and controller structure.

A block diagram of a plant and controller graphically represents the structure of a controller design and its interaction with the plant. It is possible to perform algebraic operations on the components of a block diagram to reduce the diagram to a simpler form.

Performance specifications guide the design process and provide the means for determining when controller performance is satisfactory. Controller performance specifications can be stated in both the time domain and the frequency domain. The PID controller is the most widely used control algorithm in the process industry, and that improvements in tuning PID controllers will have a significant practical impact. The objective of the study in [8] is to find a simple model based tuning rules that give insight into how the tuning depends on the process parameters based on very simple process information .These rules may then be used to assist in retuning the controller if, for example, the production rate is changed. Another related objective is that the rules should be so simple that they can be memorized. There has been previous work along these lines; most noteworthy the early paper by Ziegler and Nichols (1942), the IMC PID-tuning paper by Rivera, Morari and Skogestad (1986), and the book by Smith and Corripio (1985) [8]. The Ziegler-Nichols tunings result in a very good disturbance response for integrating processes [9].On the other hand, the IMC-tunings of Rivera et al. (1986) are known to result in poor disturbance response for integrating processes (e.g., Chien and Fruehauf (1990), Horn et al. (1996), but generally give very good responses for set point changes [9]. Derivative action is primarily recommended for a process with dominant second order dynamics. The derivative time is selected so as to cancel the second-largest process time constant.

## 1.9 Scope of the Thesis

Our literature survey reveals that a lot of work has been done towards improving the performance of PD Controllers with increased robustness. In a broad sense, such development works on the controller tuning are mostly dependent on the process model. However, for a practical process it is very difficult to find its exact model, as a result, most of the theoretical developments have limitation from practical implementation point of view.

Along with the mathematical complexity in finding out the appropriate process model, there is always a certain amount of uncertainty in model parameters. Model parameters are also changing with time due to natural phenomena like aging, scaling, erosion etc. So obtaining the desired performance from PD controller is not the only goal. Additionally it has to be robust enough to withstand the model uncertainties as well as process nonlinearities. At the same time, it is found that an optimally tuned controller is more prone to fragile. So depending on the area of application, there should be a compromise between optimality and robustness of selected parameters.

Soft computing tools like fuzzy logics, neural networks and different optimization techniques are also used by the researchers to obtain optimal settings of PD parameters. In such cases the engineers have tried to incorporate the human intelligence in the controller behaviour. Certain improvements are found in the controller performances on making them more intelligent but at the cost of higher computational complexity [10]. A controller designed to reduce the initial overshoot during set-point change usually fails to offer good load rejection behaviour. On the other hand, a controller with better load regulation cannot restrict the overshoot in the set point response. Although in some cases improvements in the process behaviour are observed during both set-point and load disturbance responses.

In our experimental purpose initially we have identified the process model by using Bump test method. Then the classical PD Controller has been designed for identified model satisfying some performance indices (here Percentage overshoot method). Then the optimal PD Controllers have been developed using different optimization techniques for the same identified process model. Then we have studied the performances of PD controller and optimized controllers through simulation experiments with identified model as well as real time experiments with actual process. For the optimization purpose we have chosen ITAE as objective functions, because they provide the overall improved performance, ITAE indicates improved set point and good load rejection respectively.

In Chapter-2, we have presented the modelling and validation of the QUBE Servo 2 made by Quanser , Canada [11] and also calculated the PD parameters by Percentage overshoot method. The modelling of the Qube Servo 2 motor has been done by Bump test modelling method and the validation has been performed. According to the validation the first order process model has been derived on which the tuning of the PD parameters has been done.

In chapter-3, we have presented the same thing as above for another Rotary Servo base unit (SRV02), made by Quanser, Canada [11].

In Chapter-4, the detailed description is presented for the Particle Swarm Optimization (PSO) based PD controllers with respect to ITAE. We compared the performance with already tuned conventional PD controllers of the identified model. Then the performance is tested with two plant models of QUBE Servo2 and SRV02 through simulation experiments with identified models as well as real time experiments with actual process.

In Chapter-5, we have presented the detailed description of Moth Flame optimization (MFO) based PD Controllers with respect to the objective function, ITAE. Initially we have tuned the conventional PD controllers and compared it with algorithm tuned PD Controllers. These controllers are then tested with two plant models through simulation experiments with identified models as well as real time experiments with actual process.

In Chapter-6, the detailed description is presented for the Grey Wolf Optimization (GWO) based PD controllers with respect to ITAE. Initially we have found the conventional PD controllers manually based on percentage overshoot method and compared it with GWO algorithm based PD Controllers. Then the performance is tested with two plant models through simulation experiments with identified models as well as real time experiments with actual process.

In Chapter-7, we have presented the Ant Lion Optimization (ALO) based PD Controllers with respect to the objective function, ITAE. Initially we have tuned the conventional PD controllers and compared with ALO algorithm based PD controllers. Then the performance is tested with two models through simulation experiments with identified models as well as real time experiments with actual process.

In Chapter-8, all the outputs of PD controllers for different set of environment are compared. An effort to determine the best algorithm for the purpose of tuning controllers has been made in this chapter.

In Chapter-9, first we have provided a brief summary of the present study. Then we have discussed the implementation issues of the four optimization techniques, Particle Swarm Optimization, Moth Flame optimization, Grey Wolf Optimization, Ant Lion Algorithm while designing optimal logic controllers in the last chapters. We have also tried to learn the future scopes for further improvement.

## References

[1] Modern Control Engineering 5th Edition Ogata.

[2] Smith, C.A. and A.B. Corripio (1985). Principles and Practice of Automatic Process Control. John Wiley & Sons.

[3] Cohen, G.H. and G.A. Coon (1953). Theoretical consideration of retarded control. Trans. ASME 75, 827– 834.

[4] Seborg, D.E., T.F. Edgar and D.A. Mellichamp (1989). Process Dynamics and Control. John Wiley & Sons Shinskey, F.G. (1998). Personal communication.

[5] Ziegler, J.G. and N.B. Nichols (1942). Optimum settings for automatic controllers. Trans. of the A.S.M.E. 64, 759–768.

[6] Astrom, K.J., T. Hagglund, C.C. Hang and W.K Ho (1993). Automatic tuning and adaptation for PID controllers - A survey. Control Eng. Practice 1(4), 699–714.

[7] Holm, O. and A. Butler (1998). Robustness and performance analysis of PI and PID controller tunings. Technical report. 4th year project, Department of Chemical Engineering. Norwegian University of Science and Technology, Trondheim.

[8] Ho, W.K., K.W. Lim and W. Xu (1998). Optimal gain and phase margin tuning for PID controllers. Automatica 34(8), 1009–1014. See Automatica.

[9] Tyreus, B.D. and W.L. Luyben (1992). Tuning PI controllers for integrator/dead time processes. Ind. Eng. Chem. Res. pp. 2628–2631.

[10] Horn, I.G., J.R. Arulandu, J. Gombas, J.G. VanAntwerp and R.D. Braatz (1996). Improved filter design in internal model control. Ind. Eng. Chem. Res. 35(10), 3437–3441.

[11] Quanser Operating Manual; *Documentation for the* USER MANUAL Quanser, Ontario, Canada, 2016.

# Chapter 2

Modelling, validation and Position control of Quanser QUBE-Servo 2

## 2.1 Introduction

Direct-current motors are used in a variety of applications. As discussed in the QUBE-Servo 2 User Manual [1], the QUBE-Servo 2 has a brushed DC motor that is connected to a PWM amplifier. Encoders are used here to measure angular position. There are many types of encoders but we have used here in this experiment the rotary incremental optical encoder, the angle they measure depends on the last position and when it was last powered. The Quanser QUBE-Servo 2 is a direct-drive rotary servo system as shown in Fig. 2.1

The resolution of the encoder:- In order to measure the total counts per revolution, we moved the disc to the 0 degree position marked on the QUBE-Servo 2 and the controller is started and the disc rotates one full rotation. The encoder count reads 2048, which is in-line with the specifications given in the QUBE-Servo 2 User Manual [1]. The encoder resolution is 512 lines per revolution, but goes up to 2048 in quadrature mode (4 x 512 = 2048). To get a measurement in degrees we need a gain of 360 °/2048 cnts = 0.1758 °/cnts.

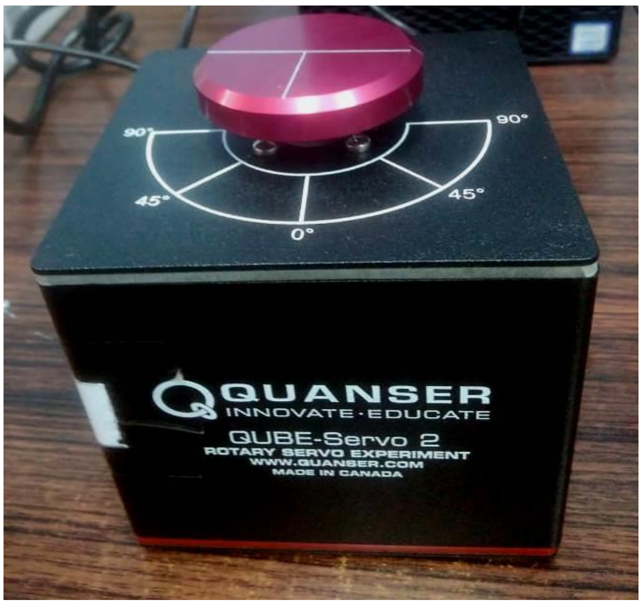

**Fig 2.1:** Quanser QUBE-Servo2 System

## 2.2 First Principle Modelling

The motor armature circuit schematic of QUBE-Servo 2 is shown in Fig 2.2 and the electrical and mechanical parameters are given in Table 2.1. The DC motor shaft is connected to the load hub. The hub is a metal disk used to mount the disk or rotary pendulum and has a moment of inertia of $J_h$. A disk load is attached to the output shaft with a moment of inertia of $J_d$.

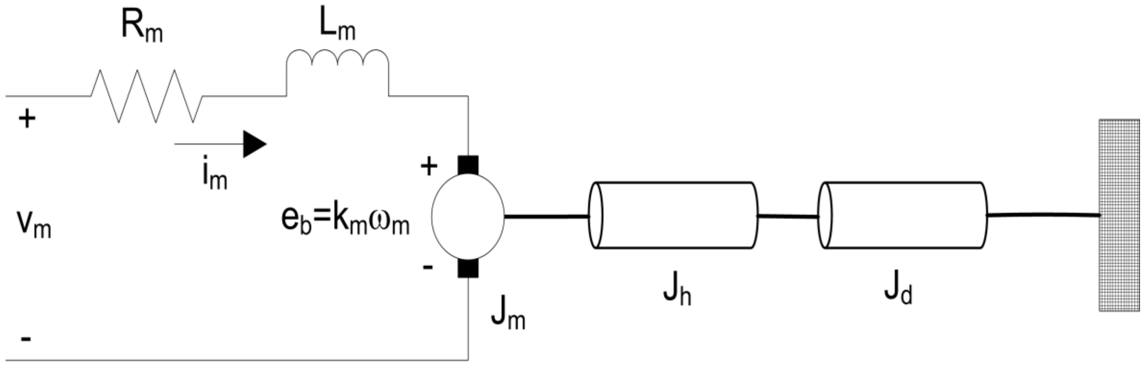

**Fig 2.2**: QUBE-Servo 2 DC motor and load

The back-emf (electromotive) voltage $e_b(t)$ depends on the speed of the motor shaft, $w_m$, and the back-emf constant of the motor, $k_m$. It opposes the current flow. The back emf voltage is given by: $e_b(t) = k_m \omega_m(t)$ (2.21)

| Symbol | Description | Value |
|---|---|---|
| **DC Motor** | | |
| *Rm* | Terminal resistance | 8.4Ω |
| $K_t$ | Torque Constant | 0.042N.m/A |
| $K_m$ | Motor back-emf constant | 0.042 V/(rad/s) |
| $J_m$ | Rotor inertia | 4.0 $x10^6$ kg:$m^2$ |
| $J_h$ | Load hub inertia | 0.6$x10^6$ kg.m2 |
| $r_h$ | Load hub mass | 0.0111 m |
| $m_h$ | Load hub mass | 0.0106 kg |
| $L_m$ | Rotor inductance | 1.16 mH |
| **Load Disk** | | |
| $M_d$ | Mass of disk load | 0.053 kg |
| $r_d$ | Radius of disk load | 0.0248 m |

**Table 2.1:** QUBE-Servo 2 system parameters

Using Kirchhoff's Voltage Law, we can write the following equation:

$$v_m(t) - R_m\, i_m\,(t) - L_m \frac{\partial}{\partial t} i_m\,(t) - k_m \omega_m\,(t) = 0 \qquad (2.2.2)$$

Since the motor inductance $L_m$ is much less than its resistance, it can be ignored. Then, the equation becomes $v_{\mathrm{m}}\,(t)$ - $R_m\, i_m\,(t)$ - $k_m \omega_m\,(t)$ =0 (2.2.3)

Solving for $i_m(t)$, the motor current can be found as:

$$i_m(t) = \frac{v_m(t) - k_m \omega_m(t)}{R_m} \qquad (2.2.4)$$

The motor shaft equation is expressed as $J_{eq}\, \omega_m(t) = \tau_m(t)$ (2.2.5)

where $J_{eq}$ is total moment of inertia acting on the motor shaft and $\tau_m$ is the applied torque from the DC motor. Based on the current applied, the torque is

$$\tau_m(t) = K_m i_m(t) \qquad (2.2.6)$$

The moment of inertia of a disk about its pivot, with mass *m* and radius *r*, is

$$J = (1/2)\, m\, r^2 \qquad (2.2.7)$$

Based on the models, we have designed a model that applies a 1 - 3 V, 0.4 Hz square wave to the motor and reads the servo velocity using the encoder.

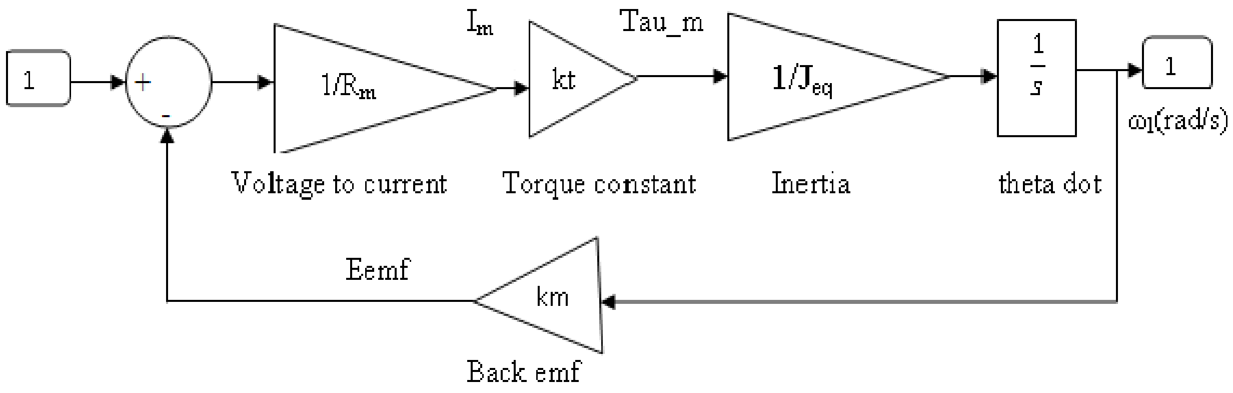

**Fig 2.3:** Completed QUBE-Servo 2 Model subsystem.

Based on the parameters given in the Table 2.1, we can calculate the total moment of inertia acting on the motor shaft which is the sum of the motor armature or rotor inertia $J_m$ the hub inertia $J_h$ and the disk inertia $J_d$. The equivalent moment of inertia is therefore

$$J_{eq} = J_m + J_h + J_d \qquad (2.2.8)$$

The moment of inertia of the hub and disk load are: $J_h = (1/2)\, m_h\, r_h^2$

And $J_d = (1/2) m_d\, r_d^2$ .

Therefore, $\mathrm{J_{eq}} = 2.09 \times 10^{-5}$

The QUARC controller and the model generated as in Fig 2.3 is run and obtained the response as in Fig 2.4.

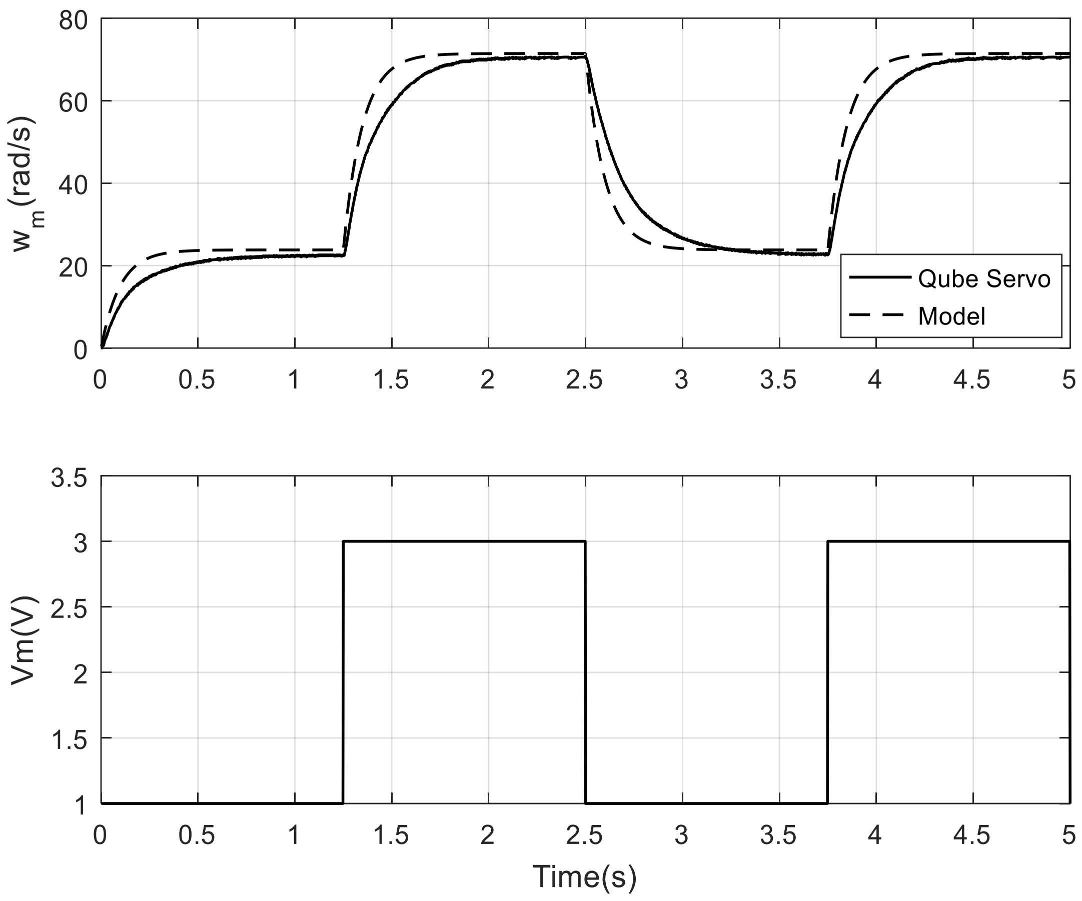

**Fig 2.4:** Motor Speed and Motor voltage of the QUBE-Servo2 and the model according to First principle modelling

## 2.3 Bump test Modelling

The bump test is a simple test based on the step response of a stable system. A step input is given to the system and its response is recorded. Considering a first order system given by the following transfer function:

$$\frac{Y(s)}{U(s)} = \frac{K}{\tau s+1}$$

The step input begins at time $t_0$. The input signal has a minimum value of $u_{min}$ and a maximum value of $u_{max}$. The resulting output signal is initially at $y_0$. Once the step is applied, the output tries to follow it and eventually settles at its steady-state value $y_{ss}$. From the output and input signals, the steady-state gain is

$$K = \frac{\Delta y}{\Delta u} \tag{2.3.1}$$

where $\Delta y = y_{ss} - y_0$ and $\Delta u = u_{max} - u_{min}$. The time constant of a system τ is defined as the time it takes the system to respond to the application of a step input to reach 63.2% of its steady-state value, i.e.

$$t_1 = t_0 + \tau \tag{2.3.2}$$

$$where \quad y(t_1) = 0.632\Delta y + y_0$$

Then, we can read the time $t_1$ that corresponds to $y\ (t_1)$ from the response data. From Eq. (2.3.2), the model time constant can be found as:

$$\tau = t_1 - t_0$$

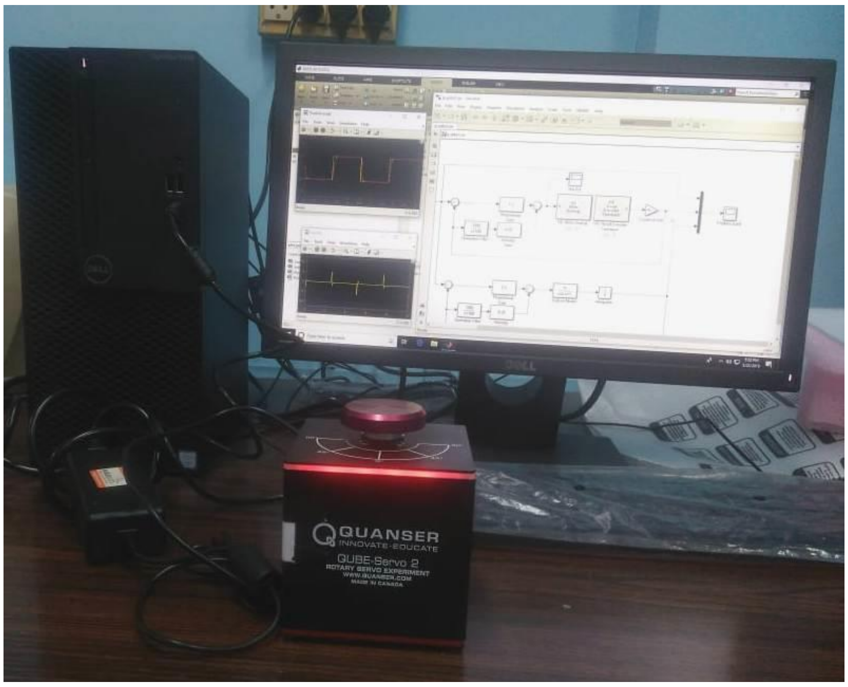

**Fig 2.5:** Experimental set up for Quanser QUBE-Servo2

Going back to the QUBE-Servo 2 system, the s-domain representation of a step input voltage with a time delay $t_0$ is given by

$$V_m(s) = \frac{A_v}{s} e^{-st_0} \tag{2.3.3}$$

where $A_v$ is the amplitude of the step and $t_0$ is the step time (i.e. the delay).
The voltage-to-speed transfer function is

$$\frac{\Omega_m(s)}{V_m(s)} = \frac{K}{\tau s + 1} \tag{2.3.4}$$

where $K$ is the model steady-state gain, T is the model time constant,
$\Omega_m(s) = \mathcal{L}[\omega_m(t)]$ is the load gear rate, and
$V_m(s) = \mathcal{L}[V_m(t)]$ is the applied motor voltage.
If we substitute input in Eq. 2.3.3 into the system transfer function in Eq.(2.3.4), we get:

$$\Omega_m(s) = \frac{K}{(\tau s+1)} \frac{A_v}{s} e^{-st_0} \tag{2.3.5}$$

We can then find the QUBE-Servo 2 motor speed step response in the time domain $\omega_m(t)$ by taking inverse Laplace of this equation

$$\omega_m(t) = K A_v (1 - e^{-(t-t_0)/\tau}) + \omega_m(t_0) \qquad (2.3.6)$$

noting the initial conditions $\omega_m(0-) = \omega_m(t_0)$.

Based on the models designed in QUBE-Servo 2, we desire to design a model that applies a step of 2 V to the motor for 2.5 seconds and reads the servo velocity using the encoder.

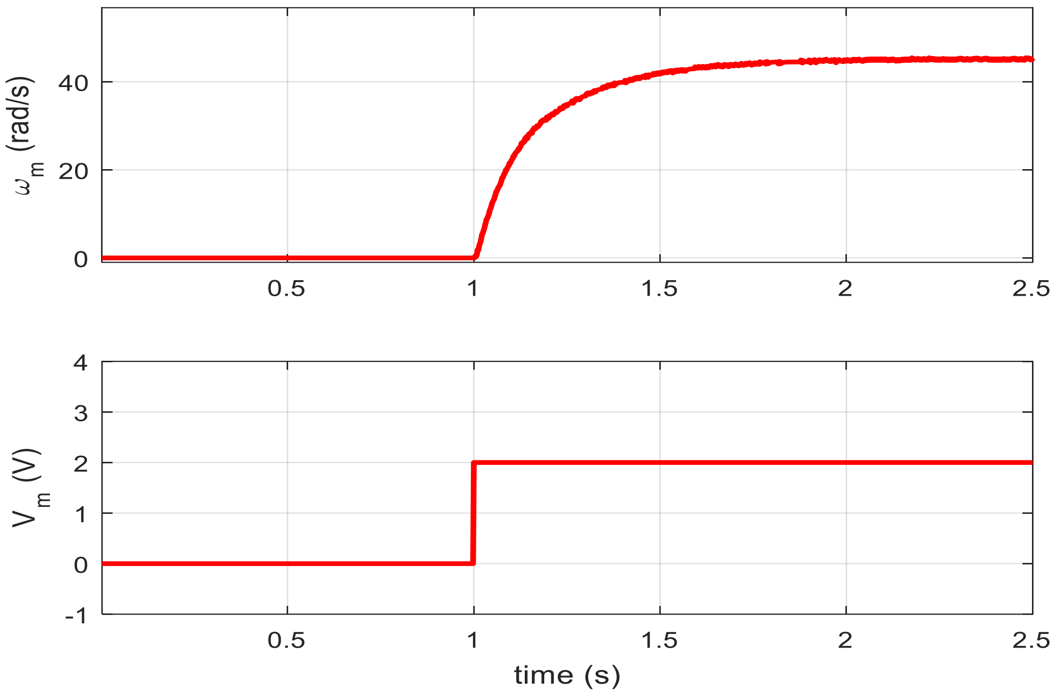

**Fig 2.6**: Motor Speed and Motor voltage of the QUBE-Servo 2 step response.

From Fig 2.6, the measured initial and steady-state load shaft speeds are

$\omega_m(t_0) = 0$ *rad/s* and

$\omega_{mss} = 45.6132$ *rad/s* and

the input voltage amplitude is $A_v = 2.0$ V

Using the above Eq. 2.3.1 with the collected data from the Fig 2.6, the resulting steady-state gain is: $K = 22.7$ rad/(Vs)

To find time of the first decay $t_1 = t_0 + \tau$ , the corresponding speed is measured. From Figure the time at the shaft speed

$$\omega_m(t_0 + \tau) = 28.33 \text{ rad/s} \quad \text{is } t_1 = 1.158 \text{ s}$$

The step start time is $t_0 = 1.0$ s

Given the step start time $t_0$ and decay time $t_1$ the time constant is

$$\tau = 0.158 \text{ s}$$

Therefore, voltage-to-speed transfer function of the identified model becomes

$$\frac{\Omega_m(s)}{V_m(s)} = \frac{22.7}{0.158s+1} \qquad (2.3.7)$$

And the voltage-to-position transfer function is

$$P(s) = \frac{\theta_m(s)}{V_m(s)} = \frac{22.7}{s(0.158s+1)} \tag{2.3.8}$$

To check if the derived model parameters $K$ and $\tau$ are correct, the Simulink diagram has been modified to include a Transfer Function block with the first-order model.

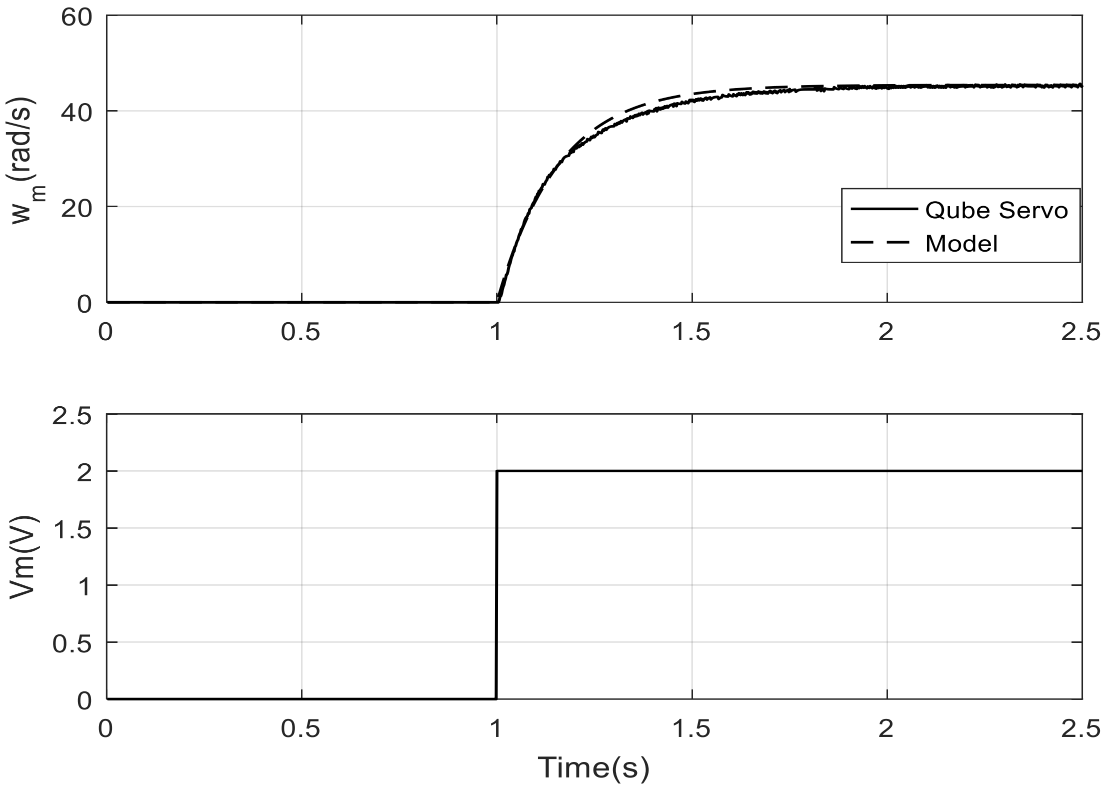

**Fig 2.7:**Motor Speed and Motor voltage of the QUBE-Servo2 and the model according to Bump test modelling (Validation)

The actual and model responses in Fig 2.7 match very closely. Given the model represents the actual system accurately, the parameters derived are correct.

### 2.4 Position Control

The QUBE-Servo 2 voltage-to-position transfer function is

$$P(s) = \frac{\theta_m(s)}{V_m(s)} = \frac{K}{s(\tau s+1)} \tag{2.4.1}$$

where $K$ = 22.7 rad/(V-s) is the model steady-state gain, $\tau$ = 0.158 s is the model time constant, $\theta_m(s)$ is the motor or disk position, and $V_m$(s) is the applied motor voltage. The value of K and $\tau$ has been found by the Bumptest modelling experiment mentioned in Section 2.3.

Proportional-Derivative (PD) controller is a distinct class of controller having a wide acceptance for industrial position control applications. Mostly, in industrial automation processes where robots and manipulators are extensively used, any oscillation in arm movement is highly undesirable [1, 2]. At the same time, good responsiveness in its behaviour during achieving a new position and simultaneously improved load rejection is expected. To satisfy these requirements appropriate amount of damping (D action) should be present in control action and at the same time suitable amount of sensitivity (P action) should also be present in the controller behaviour [3]. So, to get the best result from servo motor based position control applications, integral component (I action) of PID controller is usually kept off as it provides oscillatory responses with large overshoots or undershoots for such integrating processes. A variation of the classical PD control will be used: the proportional-velocity control as illustrated in Fig 2.8. Here, only the negative velocity is fed back (i.e. not the velocity of the error) and a low-pass filter is used along with the derivative term to suppress measurement noise. The combination of a first order low-pass filter and the derivative term results in a high-pass filter H(s) which will be used instead of a direct derivative.

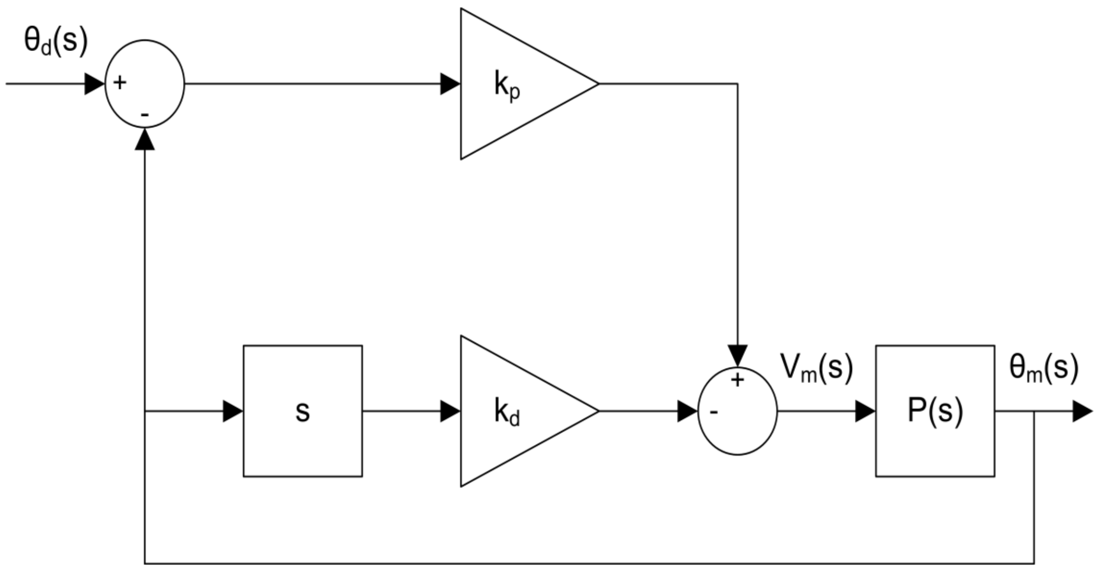

**Fig 2.8**: Block diagram of PV control

The proportional-velocity (PV) control has the following structure

$$u = ( k_p ( r(t) - y(t) ) - k_d \dot{y}(t) ) \qquad (2.4.2)$$

where $k_p$ is the proportional gain, $k_d$ is the derivative (velocity) gain, $r = \theta_d(t)$ is the set point or reference motor /load angle, $y = \theta_m(t)$ is the measured load shaft angle, and $u = V_m(t)$ is the control input (applied motor voltage).

The closed-loop transfer function of the QUBE-Servo 2 is denoted $\frac{Y(s)}{R(s)} = \frac{\theta_m(s)}{\theta_d(s)}$

Assume all initial conditions are zero, i.e. $\theta_m(0-) = 0$ and $\dot{\theta}_m(0-) = 0$, taking the Laplace transform of Eq.(2.4.2) yields

$$U(s) = (k_p \left( R(s) - Y(s) \right) - k_d s \, Y(s)) \qquad (2.4.3)$$

which can be substituted into Eq. (2.4.1) to result in

$$Y(s) = \frac{K(k_p\left(R(s) - Y(s)\right) - k_d s\, Y(s))}{s(\tau s+1)}$$

Solving for Y (s)/R(s), we obtain the closed-loop expression

$$\frac{Y(s)}{R(s)} = \frac{Kk_p}{\tau s^2 + (1+K\,k_d)s + Kk_p} \qquad (2.4.4)$$

This is a second-order transfer function. By comparing the standard second-order transfer function $\frac{Y(s)}{R(s)} = \frac{\omega_n^2}{s^2+2\xi\omega_n+\omega_n^2}$ (2.4.5)

The characteristic equation of the QUBE-Servo2 closed-loop transfer function in Eq. (2.4.4) is: $\tau s^2 + (1+K\,k_d)s + Kk_p = 0$ and can be re-structured into the form $s^2 + \frac{(1+K\,k_d)s}{\tau} + \frac{Kk_p}{\tau} = 0$

Equating this with the standard second order system Eq. (2.4.5) gives the expressions $\frac{Kk_p}{\tau} = \omega_n^2$ and $\frac{(1+K\,k_d)s}{\tau} = 2\xi\omega_n$

Solving for $k_p$ and $k_d$ to obtain the control gain equations

$$k_p = \frac{\omega_n^2}{\tau} \qquad (2.4.6)$$

$$\text{and } k_d = \frac{2\xi\omega_n - 1}{K} \qquad (2.4.7)$$

For the response to have a peak time of 0.15 s and a percentage overshoot of 5 %, the natural frequency and damping ratio needed are
$\omega_n$ = 28.9321 rad/s and $\xi$ = 0.6903.

Using the model parameters given above and the desired natural frequency with Eq.(2.4.6) generates the proportional control gain

$k_p$ = 5.8263 V/rad

Similarly, the derivative control gain is obtained by substituting the model parameters given above with the damping ratio specification into Eq.(2.4.7)

$k_d$ = 0.2340 V/(rad/s)

When these gains are used with the PD controller, the position response of the load gear on a QUBE-Servo 2 with a disk load should satisfy the specifications. The motor voltage and the position response of both the QUBE-Servo 2 and the designed model has been plotted in the following Fig 2.9

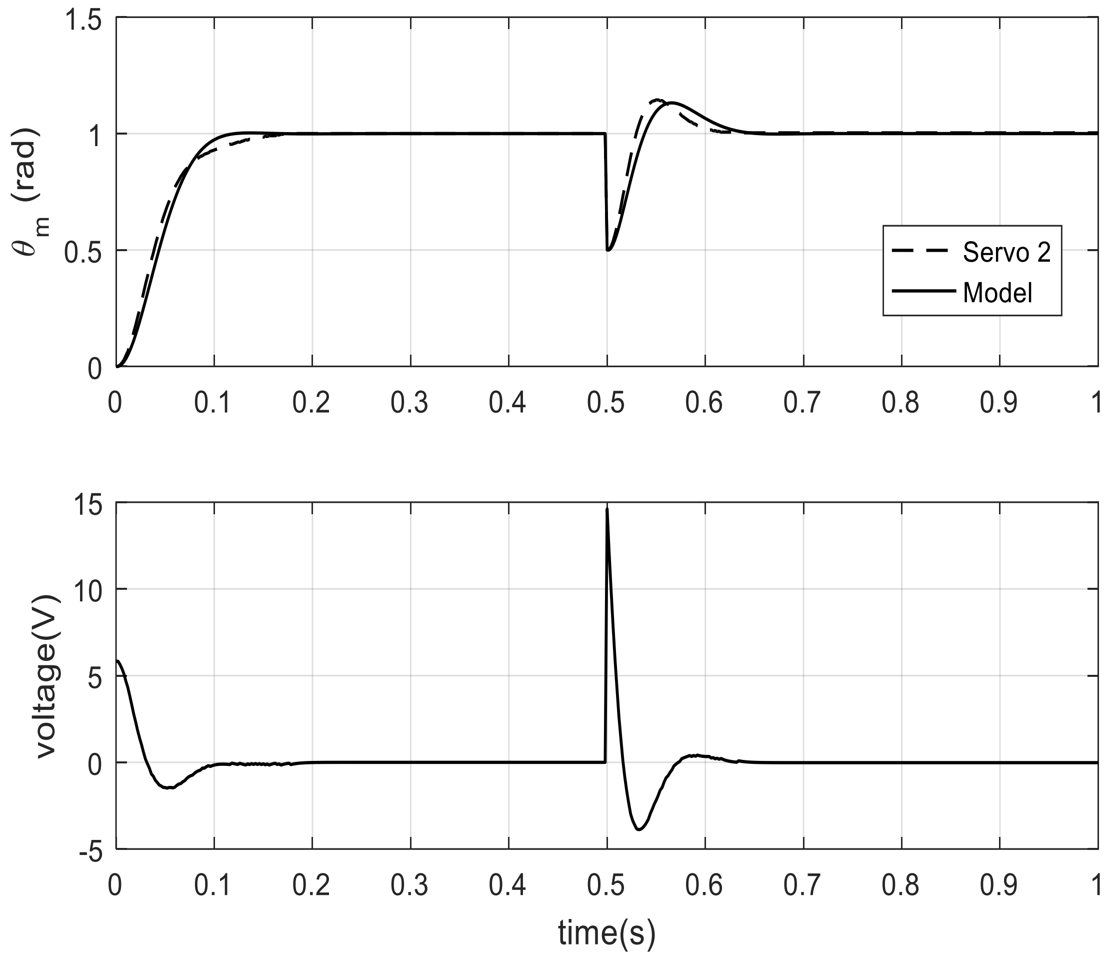

**Fig 2.9**: Motor Speed and Motor voltage of the QUBE-Servo2 and the model according to PD Control

| | **QUBE-Servo2** | **Model** |
|---|---|---|
| Rise Time (ms) | 70.762 | 64.010 |
| %Overshoot | -0.122 | 0.117 |
| Peak Time(sec) | 0.178 | 0.136 |
| ITAE | 0.01074 | 0.01135 |
| IAE | 0.06353 | 0.06633 |
| ISE | 0.03474 | 0.03891 |

**Table 2.2:** Performance table for PD Controller of QUBE-Servo2 and the designed model

The response satisfies the overshoot since the magnitude of the measured percent overshoot is less than 5 %. However, the measured peak time in Table

2.2 surpasses 0.15 s in the case of QUBE-Servo2, the response is slower. The QUBE-Servo 2 model response from the transfer function is ideal, and thus, has no steady state error. The QUBE-Servo 2 response is expected to have a small steady state error due to friction, which has not been modelled. To improve the response time (i.e. decrease the peak time), we can increase the proportional gain. Thus in order to get tuned result we have implemented various optimization technique. For the optimization purpose we have chosen ITAE as objective functions because they provide the overall improved performance, ITAE indicates improved set point response and good load rejection respectively.

**Refernces:**

[1] *Documentation for the* USER MANUAL QUBE-Servo 2. Quanser, Ontario, Canada, 2016.

[2] C. Aguilar-Ibáňez and Sira-Ramirez, “PD control for active vibration damping in an under actuated nonlinear system,” *Asian Journal of Control,* vol. 4, no. 4, pp. 502–588, 2002.

[3] K. R. Atia and M. P. Cartmell, “A new methodology for designing PD controllers,” *Robotica,* vol. 14, no. 3, pp. 267–273, 2001.

[4] K. Ogata, *Modern Control Engineering.* New Jersey: Prentice-Hall, 2002.

# Chapter 3

Modelling, validation and position control of Quanser SRV02 Rotary Servo Base unit

## 3.1 Introduction

The Quanser SRV02 rotary servo plant, pictured in Fig 3.1, consists of a DC motor that is enclosed in a solid aluminium frame and equipped with a planetary gearbox. The motor has its own internal gearbox that drives external gears. The SRV02 is equipped with three sensors: potentiometer, encoder, and tachometer. The potentiometer and encoder sensors measure the angular position of the load gear and the tachometer can be used to measure its velocity.

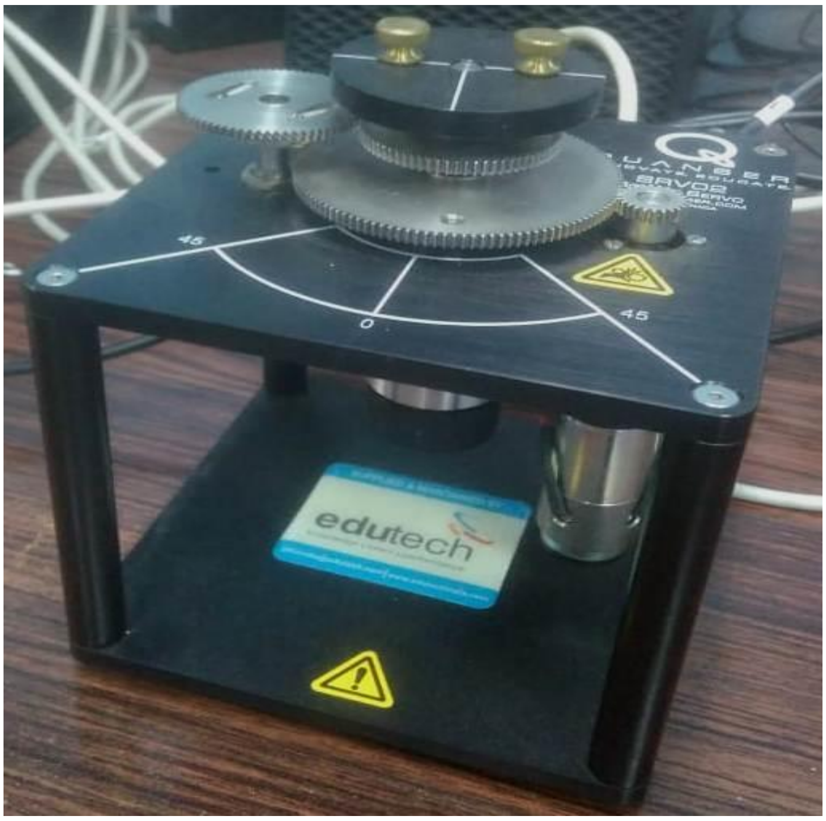

**Fig 3.1:** Quanser SRV02 system

Table 3.1 lists and characterizes the main parameters associated with the SRV02. Some of these are used in the mathematical model.

| **Symbol** | **Description** | **Value** |
|---|---|---|
| $V_{nom}$ | Motor nominal input voltage | 6.0 V |
| $R_m$ | Motor armature resistance | 2.6 Ω |
| $L_m$ | Motor armature inductance | 0.18 mH |
| $k_t$ | Motor current-torque constant | $7.68\ x\ 10^{-3}$ N-m/A |
| $k_m$ | Motor back-emf constant | $7.68x10^{-3}$ V/(rad/s) |
| $K_g$ | High-gear total gear ratio | 70 |
|  | High-gear total gear ratio | 14 |
| $\eta_m$ | Motor efficiency | 0.69 |
| $\eta_g$ | Gearbox efficiency | 0.90 |
| $J_{m,rotor}$ | Rotor moment of inertia | $3.90\ x10^{-7}$ kg-m$^2$ |
| $J_{tach}$ | Tachometer moment of inertia | $7.06\ x10^{-8}$ kg-m$^2$ |
| $J_{eq}$ | High-gear equivalent moment of inertia without external load | $2.087\ x10^{-3}$ kg-m$^2$ |
|  | Low-gear equivalent moment of inertia without external load | $9.785\ x10^{-5}$ kg-m$^2$ |
| $B_{eq}$ | High-gear Equivalent viscous damping coefficient | 0.015 N-m/(rad/s) |
| $m_b$ | Mass of bar load | 0.038 kg |
| $L_b$ | Length of bar load | 0.1525 m |
| $m_d$ | Mass of disc load | 0.04 kg |
| $r_d$ | Radius of disc load | 0.05 m |
| $m_{max}$ | Maximum load mass | 5 kg |
| $f_{max}$ | Maximum input voltage frequency | 50 Hz |
| $I_{max}$ | Maximum input current | 1 A |
| $w_{max}$ | Maximum motor speed | 628.3 rad/s |

**Table 3.1:** Main SRV02 Specifications

## 3.2 First Principle Modeling

### 3.2.1 Electrical Equations

The DC motor armature circuit schematic and gear train is illustrated in Fig 3.2. As specified $R_m$ is the motor resistance, $L_m$ is the inductance, and $k_m$ is the back-emf constant.

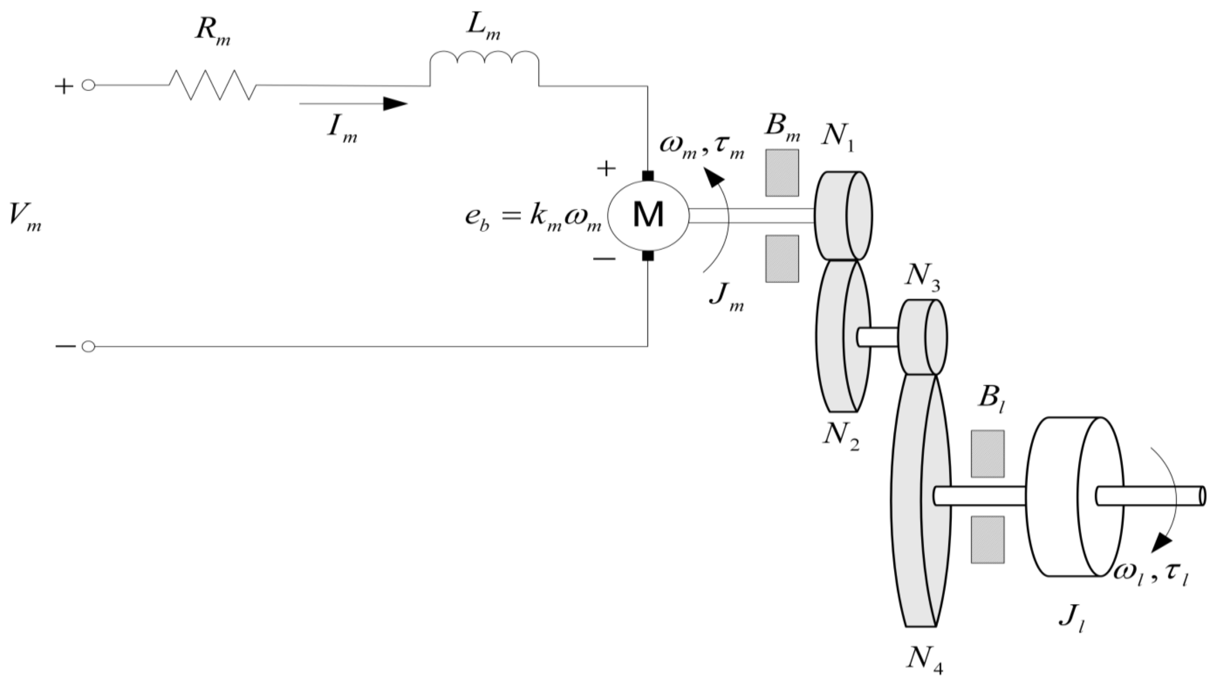

**Fig 3.2**: SRV02 DC motor armature circuit and gear train

The back-emf (electromotive) voltage $e_b(t)$ depends on the speed of the motor shaft, $\omega_m(t)$, and the back-emf constant of the motor, km. It opposes the current flow. The back emf voltage is given by: $e_b(t) = k_m\,\omega_m(t)$ (3.2.1)
Using Kirchhoff's Voltage Law, we can write the following equation:

$$v_m(t) - R_m\, i_m\,(t) - L_m \frac{\partial}{\partial t} i_m\,(t) - k_m\omega_m\,(t) = 0 \qquad (3.2.2)$$

Since the motor inductance $L_m$ is much less than its resistance, it can be ignored. Then, the equation becomes $v_m\,(t)$ - $R_m\, i_m\,(t)$ - $k_m\omega_m\,(t)$ =0 (3.2.3)
Solving for $I_m(t)$, the motor current can be found as:

$$i_m(t) = \frac{v_m(t) - k_m\omega_m(t)}{R_m} \qquad (3.2.4)$$

3.2.2 Mechanical Equations
The equation of motion describing the speed of the load shaft, $\omega_l$, with respect to the applied motor torque, $\tau_m$, is developed. Since the SRV02 is a one degree-of-freedom rotary system, Newton's Second Law of Motion can be written as:

$$J\,.\alpha = \tau$$

where $J$ is the moment of inertia of the body (about its center of mass), $\alpha$ is the angular acceleration of the system, and $\tau$ is the sum of the torques being applied to the body. As illustrated in Fig 3.2, the SRV02 gear train along with the viscous friction acting on the motor shaft, $B_m$, and the load shaft $B_l$ are considered. The load equation of motion is

$$J_l \frac{\partial}{\partial t}\omega_l(t) + B_l\,\omega_l\,(t) = \tau_l\,(t) \qquad (3.2.5)$$

where $J_l$ is the moment of inertia of the load and $\omega_l$ is the total torque applied on the load. The load inertia includes the inertia from the gear train and from any external loads attached, e.g. disc or bar. The motor shaft equation is expressed as: $J_m \frac{\partial}{\partial t}\omega_m(t) + B_m \omega_m(t) + \tau_{ml}(t) = \tau_m(t)$ (3.2.6)

where $J_m$ is the motor shaft moment of inertia and $\tau_{ml}$ is the resulting torque acting on the motor shaft from the load torque. The torque at the load shaft from an applied motor torque can be written as:

$$\tau_l(t) = \eta_g K_g \tau_{ml}(t) \qquad (3.2.7)$$

where $K_g$ is the gear ratio and $\eta_g$ is the gearbox efficiency. The planetary gearbox that is directly mounted on the SRV02 motor is represented by the $N_1$ and $N_2$ gears in Fig 3.1 and has a gear ratio of

$$K_{gi} = N_2/N_1 \qquad (3.2.8)$$

This is the internal gear box ratio. The motor gear $N_3$ and the load gear $N_4$ are directly meshed together and are visible from the outside. These gears comprise the external gear box which has an associated gear ratio of

$$K_{ge} = N_4/N_3 \qquad (3.2.9)$$

The gear ratio of the SRV02 gear train is then given by:

$$K_g = K_{ge}K_{gi} \qquad (3.2.10)$$

Thus, the torque seen at the motor shaft through the gears can be expressed as:

$$\tau_{ml}(t) = \frac{\tau_l(t)}{\eta_g K_g} \qquad (3.2.11)$$

Intuitively, the motor shaft must rotate $K_g$ times for the output shaft to rotate one revolution: $\theta_m(t) = K_g\, \theta_l(t)$ (3.2.12)

We can find the relationship between the angular speed of the motor shaft, $\omega_m$, and the angular speed of the load shaft, $\omega_l$ by taking the time derivative:

$$\omega_m(t) = K_g \omega_l(t) \qquad (3.2.13)$$

The differential equation that describes the motion of the load shaft with respect to an applied motor torque is as follows:

$$J_m K_g \frac{\partial}{\partial t}\omega_l(t) + B_m K_g \omega_l(t) + J_l \frac{\frac{\partial}{\partial t}\omega_l(t) + B_l\, Kg\, \omega_l(t)}{\eta_g K_g} = \tau_m(t) \qquad (3.2.14)$$

Collecting the coefficients in terms of the load shaft velocity and acceleration gives

$$(\eta_g K_g^2 J_M + J_l)\frac{\partial}{\partial t}\omega_l(t) + (\eta_g K_g^2 B_m + B_l)\omega_l(t) = \eta_g K_g \omega_m(t) \qquad (3.2.15)$$

Defining the following terms:

$$J_{eq} = \eta_g K_g^2 J_m + J_l \qquad (3.2.16)$$

$$B_{eq} = \eta_g K_g^2 B_m + B_l \qquad (3.2.17)$$

simplifies the equation as:

$$J_{eq}\frac{\partial}{\partial t}\omega_l(t) + B_{eq}\omega_l(t) = \eta_g K_g \tau_m(t) \qquad (3.2.18)$$

## 3.2.3 Combining the Electrical and Mechanical Equations

In this section the electrical equation derived and the mechanical equation are brought together to get an expression that represents the load shaft speed in terms of the applied motor voltage.

The motor torque is proportional to the voltage applied and is described as

$$T_m(t) = \eta_m k_t I_m(t) \qquad (3.2.19)$$

where $k_t$ is the current-torque constant (N.m/A), $\eta_m$is the motor efficiency, and $I_m$ is the armature current. We can express the motor torque with respect to the input voltage $V_m(t)$ and load shaft speed $\omega_l(t)$ by substituting the motor armature current into the current-torque relationship is as follows:

$$\tau_m\,(t) = \frac{\eta_m k_t (V_m(t) - k_m \omega_m(t))}{R_m} \qquad (3.2.20)$$

After substituting we can express this in terms of $V_m$ and $\omega_l$,

$$\tau_m(t) = \frac{\eta_m k_t (V_m(t) - k_m K_g \omega_l(t))}{R_m} \qquad (3.2.21)$$

$$J_{eq} \frac{\partial}{\partial t} \omega_l(t) + B_{eq} \omega_l(t) = \eta_m \eta_g K_g k_t \frac{(V_m(t) - k_m K_g \omega_l(t))}{R_m} \qquad (3.2.22)$$

After collecting the terms, the equation becomes

$$J_{eq} \frac{\partial}{\partial t} \omega_l(t) + \left(\frac{k_m \eta_g K_g^2 \eta_m k_t}{R_m} + B_{eq}\right) \omega_l(t) = \eta_m \eta_g K_g k_t \frac{V_m(t)}{R_m} \qquad (3.2.23)$$

This equation can be re-written as:

$$J_{eq} \frac{\partial}{\partial t} \omega_l(t) + B_{eq} \omega_l(t) = A_m V_m(t) \qquad (3.2.24)$$

where the equivalent damping term is given by:

$$B_{eq,v} = \frac{k_m \eta_m \eta_g K_g^2 k_t + B_{eq} R_m}{R_m} \qquad (3.2.24a)$$

and the actuator gain equals

$$A_m = \frac{\eta_m \eta_g K_G^2 k_t}{R_m} \qquad (3.2.24b)$$

Taking the Laplace transform of the equation and assuming $\omega_l(0\text{-}) = 0$ gives

$$J_{eq} s \omega_l(s) + B_{eq,v} \omega_l(s) = A_m V_m(s)$$

Solving for $\frac{\omega_l(s)}{V_m(s)}$ gives the plant transfer function of the load shaft speed as a function of the motor input voltage:

$$\frac{\omega_l(s)}{V_m(s)} = \frac{A_m}{J_{eq} s + B_{eq,v}} \qquad (3.2.25)$$

The time constant parameter is $\tau = \frac{J_{eq}}{B_{eq,v}}$ (3.2.26)

And the steady state gain is $K = \frac{A_m}{B_{eq,v}}$ (3.2.27)

The equivalent viscous damping parameter $B_{eq}$= 0.015Nms/rad (in the high-gear configuration). Substituting all the specifications into the above equation gives

$$B_{eq,v} = 0.0844\text{N m s / rad}$$

Evaluating the actuator gain expression with the SRV02 parameters gives

$$A_m = 0.129 \text{ N m/V}$$

The moment of inertia about the motor shaft equals $J_m = J_{tach} + J_{m,rotor}$
Evaluating the above expression with the parameters outlined in gives

$$J_m = 4.606251061 \times 10^{-7} \text{ kg m}^2$$

The formula to calculate the moment of inertia of a disc is

$J_{disc} = m_r^2$ /2 where $m$ is the mass and $r$ is the radius.

the external load moment of inertia equals

$$J_{l,ext} = 5.00 \times 10^{-5} \text{ kg m}^2$$

Assuming the gears are discs and using the parameters given in Table 3.1, the moment of inertia of the 24-tooth, 72-tooth, and 120-tooth gears are

$$J_{24} = 1.01 \times 10^{-7} \text{ kg m}^2$$

$$J_{72} = 5.44 \times 10^{-6} \text{ kg m}^2$$

and

$$J_{120} = 4.18 \times 10^{-5} \text{ kg m}^2$$

The total moment of inertia from the gears is

$$J_g = J_{24}(120/24)^2 + 2J_{72} + J_{120}$$

which equals $J_g = 5.52 \times 10^{-5}$ kg m$^2$
Using $J_l = J_g + J_{l,ext}$, the total load moment of inertia is

$$J_l = 1.05\times 10^{-4} \text{ kg m}^2$$

Using Equations found above with the gear train and motor specifications listed in Table 3.1 and the load inertia, the equivalent moment of inertia acting on the SRV02 motor shaft is

$$J_{eq} = 0.00214 \text{ kg m}^2$$

The steady-state gain using the above equation is $K$ = 1.53 rad/(V s)
and the model time constant is $T$ = 0.0253 s

Hence, the voltage-to-position transfer function is

$$P(s) = \frac{\theta_m(s)}{V_m(s)} = \frac{1.53}{s(0.0253s+1)} \qquad (3.2.28)$$

## 3.3 Bump test Modelling

The bump test is a simple test based on the step response of a stable system. A step input is given to the system and its response is recorded. Considering a first order system given by the following transfer function:

$$\frac{Y(s)}{U(s)} = \frac{K}{\tau s+1}$$

(3.3.1)

The step input begins at time $t_0$. The input signal has a minimum value of $u_{min}$ and a maximum value of $u_{max}$. The resulting output signal is initially at $y_0$. Once the step is applied, the output tries to follow it and eventually settles at its steady-state value $y_{ss}$. From the output and input signals, the steady-state gain is

$$K = \frac{\Delta y}{\Delta u} \qquad (3.3.2)$$

where $\Delta y = y_{ss} - y_0$ and $\Delta u = u_{max} - u_{min}$. The time constant of a system $T$ is defined as the time it takes the system to respond to the application of a step input to reach 63.2% of its steady-state value, i.e.

$$t_1 = t_0 + \tau \qquad (3.3.3)$$

where $y(t_1) = 0.632\Delta y + y_0$

Then, we can read the time $t_1$ that corresponds to $y(t_1)$ from the response data . From this, the model time constant can be found as:

$$\tau = t_1 - t_0$$

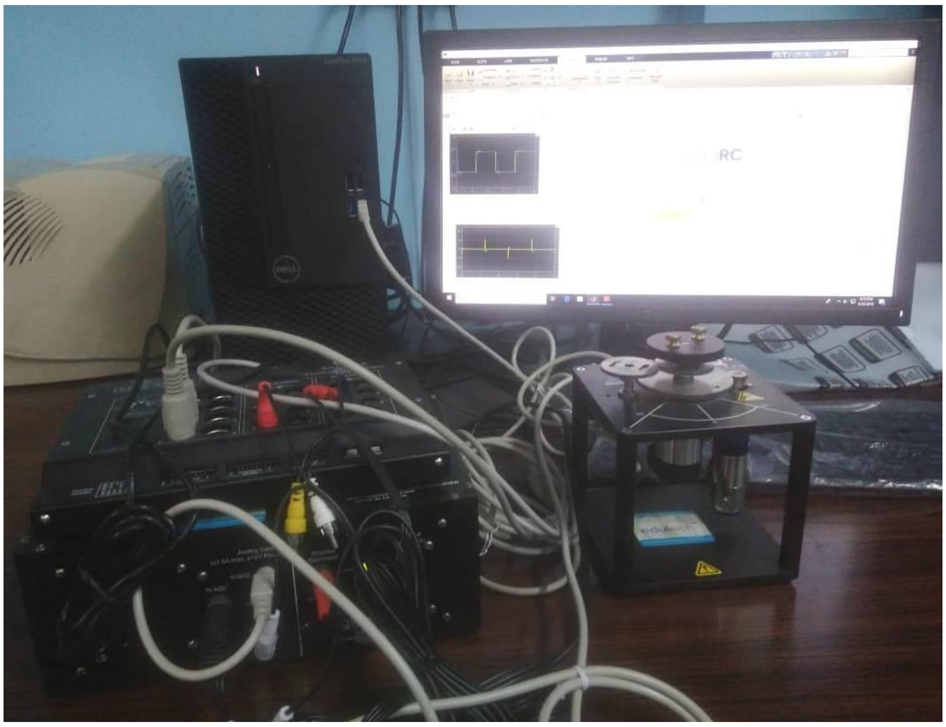

**Fig 3.3:** Experimental set up for Quanser SRV02 system

Going back to the Quanser SRV02 system, the s-domain representation of a step input voltage with a time delay $t_0$ is given by

$$V_m(s) = \frac{A_v}{s} e^{-st_0} \qquad (3.3.4)$$

where $A_v$ is the amplitude of the step and $t_0$ is the step time (i.e. the delay).

The voltage-to-speed transfer function is

$$\frac{\Omega_m(s)}{V_m(s)} = \frac{K}{\tau s+1} \qquad (3.3.5)$$

where K is the model steady-state gain, τ is the model time constant, $\Omega_m(s) = \mathcal{L}[wm(t)]$ is the load gear rate, and

$V_m(s) = \mathcal{L}[vm(t)]$ is the applied motor voltage.
If we substitute input in Eq. 3.3.4 into the system transfer function in Eq. 3.3.5, we get:

$$\Omega_m(s) = \frac{K}{(\tau s+1)}\frac{Av}{s}e^{-st0} \qquad (3.3.6)$$

We can then find the SRV02 motor speed step response in the time domain $\omega_m(t)$ by taking inverse Laplace of this equation

$$\omega_m(t) = K A_v (1 - e^{-(t-t_0)/\tau}) + \omega_m(t_0) \qquad (3.3.7)$$

noting the initial conditions $\omega_m(0-) = \omega_m(t_0)$.
Based on the models designed in SRV02, we desire to design a model that applies a step of 2 V to the motor for 2.5 seconds and reads the servo velocity using the encoder.In this method, a step input is given to the SRV02 and the corresponding load shaft response is recorded. Using the saved response, the model parameters can then be found. We have found the steady state value as $K$=2.61 rad/(V s) and the model time constant $\tau$ = 0.039s.

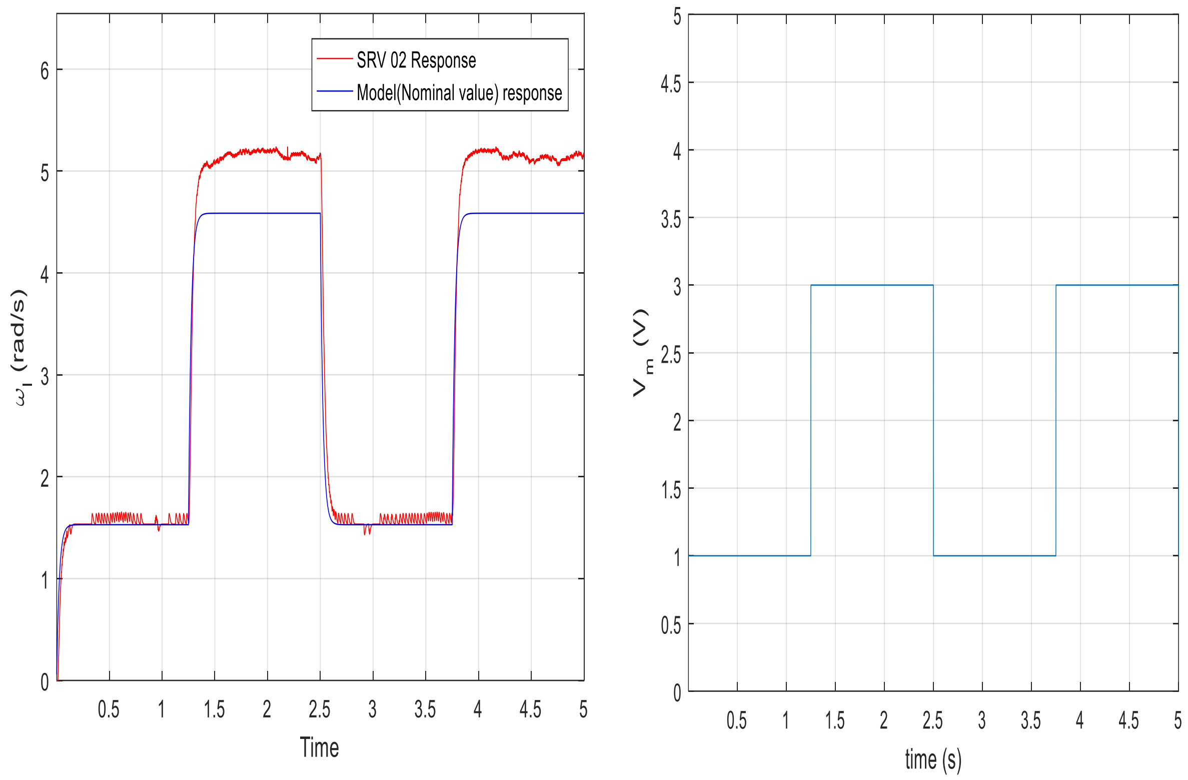

**Fig 3.4**: Motor Speed and Motor voltage of the SRV02 step response

From the Fig. 3.4, the measured initial and steady-state load shaft speeds are $\omega_l(t_0)$ = 0 rad/s and
$\omega_{lss}$ = 5.2366 rad/s and the input voltage amplitude is $A_v$ = 2V
Using the equation K = $\frac{\omega_{lss}-\omega_l(t_0)}{A_v}$ with the collected data from the Fig 3.4, the resulting steady-state gain is: $K$ = 2.6183rad/(V.s)

To find time of the first decay $t_1 = t_0 + \tau$ , the corresponding speed measurement is found. From Figure the time at the shaft speed

$$\omega_m(t_0 + \tau) = 3.9012 \text{ rad/s}$$

is $t_1 = 1.2890$ s

The step start time is $t_0 = 1.2500$ s

Given the step start time $t_0$ and decay time $t_1$ the time constant is

$$\tau = 0.0390 \text{ s}$$

Therefore, voltage-to-speed transfer function of the identified model becomes

$$\frac{\Omega_m(s)}{V_m(s)} = \frac{2.61}{0.039s+1} \tag{3.3.8}$$

And the voltage-to-position transfer function is

$$P(s) = \frac{\theta_m(s)}{V_m(s)} = \frac{2.61}{s(0.039s+1)} \tag{3.3.9}$$

## 3.4 Model Validation

To check if the model parameters *K* and τ derived from bumptest modelling and nominal value calculation are correct, the Simulink diagram has been modified to include a Transfer Function block with the first-order model .

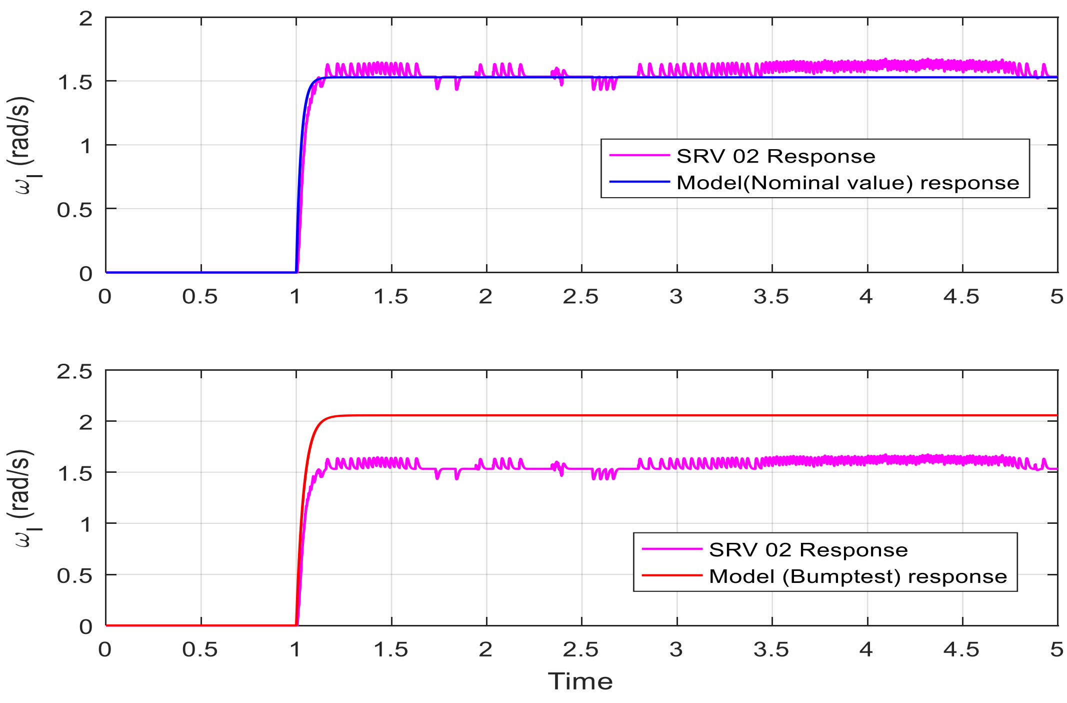

**Fig 3.5:** Nominal value and Bump test Model comparison with SRV02 response

Both the nominal response model parameters represent the SRV02 well. The transient is represented more accurately with the nominal method. The parameters derived using the bump test method do not represent the SRV02. As shown in the plot of Fig 3.5, the simulated steady-state value is higher than the measured speed.

## 3.5 SRV02 Position Control

The SRV 02 voltage-to-position transfer function as considered is

$$P(s) = \frac{\theta_m(s)}{V_m(s)} = \frac{1.53}{s(0.0253s+1)} \qquad (3.5.1)$$

The desired time-domain specifications for controlling the position of the SRV02 load shaft are:

$$e_{ss} = 0$$
$$t_p = 0.20 \text{ s and}$$
$$PO = 5.0\ \%$$

Thus, when tracking the load shaft reference, the transient response should have a peak time less than or equal to 0.20 seconds, an overshoot less than or equal to 5 %, and the steady-state response should have no error.

The proportional-velocity (PV) control has the following structure

$$u = (k_p\,(r(t) - y(t)) - k_d\,\dot{y}(t)) \qquad (3.5.2)$$

where $k_p$ is the proportional gain, $k_d$ is the derivative (velocity) gain, $r = \theta_d(t)$ is the set point or reference motor or load angle, $y = \theta_m(t)$ is the measured load shaft angle, and $u = V_m(t)$ is the control input (applied motor voltage).

The closed-loop transfer function of the QUBE-Servo 2 is denoted $\frac{Y(s)}{R(s)} = \frac{\theta_m(s)}{\theta_d(s)}$

Assume all initial conditions are zero, i.e. $\theta_m(0\text{-}) = 0$ and $\dot{\theta}_m(0\text{-}) = 0$, taking the Laplace transform of Eq.(3.5.2) yields

$$U(s) = (k_p\,(R(s) - Y(s)) - k_d s\,Y(s)) \qquad (3.5.3)$$

which can be substituted into Eq. (3.3.1) to result in

$$Y(s) = \frac{K(k_p\,(R(s) - Y(s)) - k_d s\,Y(s))}{s(\tau s+1)}$$

Solving for $Y(s)/R(s)$, we obtain the closed-loop expression

$$\frac{Y(s)}{R(s)} = \frac{Kk_p}{\tau s^2 + (1+K\,k_d)s + Kk_p} \qquad (3.5.4)$$

This is a second-order transfer function. By comparing the standard second-order transfer function $\frac{Y(s)}{R(s)} = \frac{\omega_n^2}{s^2+2\xi\omega_n+\omega_n^2}$ (3.5.5)

The characteristic equation of the SRV 02 closed-loop transfer function in Eq. (3.5.4) is: $\tau s^2 + (1 + K\,k_d)s + Kk_p = 0$ and can be re-structured into the form $s^2 + \frac{(1+K\,k_d)s}{\tau} + \frac{Kk_p}{\tau} = 0$. Equating this with the standard second order system Eq. (3.5.5) gives the expressions $\frac{Kk_p}{\tau} = \omega_n^2$ and $\frac{(1+K\,k_d)s}{\tau} = 2\xi\omega_n$

Solving for $k_p$ and $k_d$ to obtain the control gain equations,

$$\text{we get } k_p = \frac{\omega_n^2}{\tau} \qquad (3.5.6)$$

$$\text{and } k_d = \frac{2\xi\omega_n - 1}{K} \qquad (3.5.7)$$

For the response to have a peak time of 0.15 s and a percentage overshoot of 5 %, the natural frequency and damping ratio needed are $\omega_n$ = 21.7 rad/s and $\xi$ = 0.6903.
Using the model parameters given above and the desired natural frequency with Eq. 3.5.6, generates the proportional control gain

$$k_p = 7.83 \text{ V/rad}$$

Similarly, the derivative control gain is obtained by substituting the model parameters given above with the damping ratio specification into Eq. 3.5.7

$$k_d = 0.156 \text{ V/(rad/s)}$$

When these gains are used with the PD controller, the position response of the load gear on SRV02 with a disk load should satisfy the specifications. The motor voltage and the position response of both the SRV02 and the designed model has been plotted in the following Fig 3.6

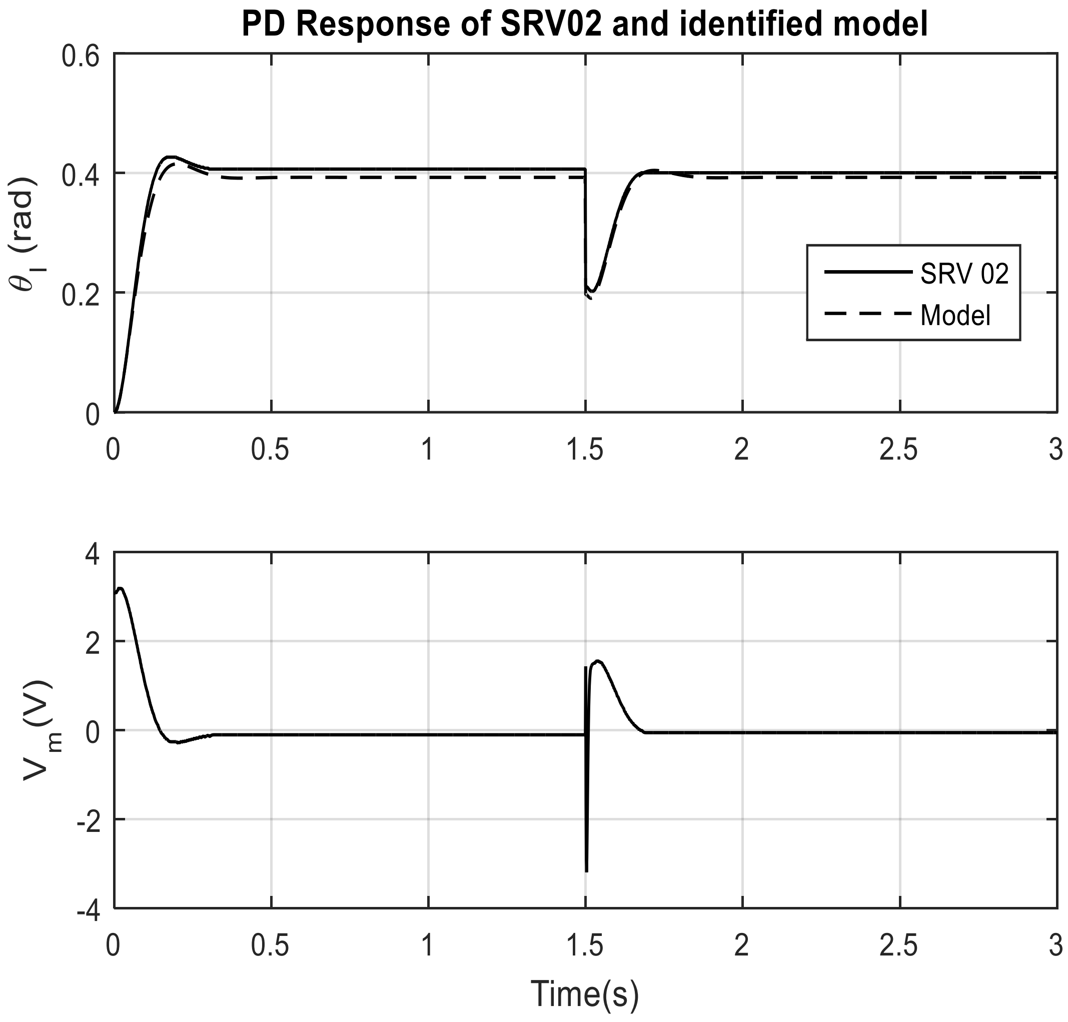

**Fig 3.6**: Motor Speed and Motor voltage of the SRV02 and the identified model according to PD Control

| | **SRV02** | **Model** |
|---|---|---|
| Rise Time (ms) | 72.083 | 78.132 |
| %Overshoot | 7.484 | 8.717 |
| Peak Time(sec) | 0.166 | 0.202 |
| ITAE | 0.06585 | 0.03135 |
| IAE | 0.07305 | 0.04871 |
| ISE | 0.01021 | 0.01033 |

**Table 3.2:** Performance table for PD Controller of SRV02 system and the designed model

The magnitude of the measured percent overshoot is slightly more than 5 %. However, the measured peak time in Table 3.2 surpasses 0.15 s in the case of identified model of SRV02; the response is slower, while that of SRV02 is nearly 0.15s. The SRV02 response from the transfer function is ideal, and thus, has no steady state error. To increase the response time (i.e. decrease the peak time), we can increase the proportional gain. Thus in order to get tuned result we have implemented various optimization technique. For the optimization purpose we have chosen ITAE as objective functions because they provide the overall improved performance, ITAE indicates improved set point response and good load rejection respectively.

**Reference:**

[1] *Documentation for the* USER MANUAL QUBE-Servo 2. Quanser, Ontario, Canada, 2016.

# Chapter 4

Optimal PD Controller using Particle Swarm Optimization (PSO)

## 4.1 Introduction

Particle swarm optimization (PSO) algorithm is a stochastic optimization technique based on swarm, which was proposed by Eberhart and Kennedy (1995) and Kennedy and Eberhart (1995). PSO algorithm simulates animal's social behaviour, including insects, herds, birds and fishes. These swarms conform a cooperative way to find food, and each member in the swarms keeps changing the search pattern according to the learning experiences of its own and other members. Main design idea of the PSO algorithm is closely related to two researches: One is evolutionary algorithm, just like evolutionary algorithm; PSO also uses a swarm mode which makes it to simultaneously search large region in the solution space of the optimized objective function.[1] The other is artificial life, namely it studies the artificial systems with life characteristics. In studying the behaviour of social animals with the artificial life theory, for how to construct the swarm artificial life systems with cooperative behaviour by computer, Millonas proposed five basic principles (van den Bergh 2001):[3]

(1) Proximity: the swarm should be able to carry out simple space and time computations.
(2) Quality: the swarm should be able to sense the quality change in the environment and response it.
(3) Diverse response: the swarm should not limit its way to get the resources in a narrow scope.
(4) Stability: the swarm should not change its behaviour mode with every environmental change.
(5) Adaptability: the swarm should change its behaviour mode when this change is worthy.

In PSO, particles can update their positions and velocities according to the environment change, namely it meets the requirements of proximity and quality. In addition, the swarm in PSO does not limit its movement but continuously search the optimal solution in the possible solution space. Particles in PSO can keep their stable movement in the search space, while change their movement mode to adapt the change in the environment. So particle swarm systems meet the above five principles.

Proposed in 1995 by J. Kennedy an R.Eberhart, the article “Particle Swarm Optimization” [1] became very popular due to its continuous optimization process allowing variations to multi targets and more. Consisting in the constant search of best solution, the method moves the particles (in this case represented as a (x,y) position) with a certain velocity calculated in every iteration. [5] Each particle’s movement has the influence of his own the best known position and also the best known position in the space-search. The final result expected is that the particle swarm converge to the best solution. It’s important to mention that PSO doesn’t use Gradient Descent, so it can be used to non linear problems once it doesn’t require that the problem have to be differentiable.

## 4.2 Particle Swarm Algorithm Flowchart [8]

The following flowchart gives a relatively complete presentation of the PSO algorithm. In the continuous space coordinate system, mathematically, the PSO can be described as follows.
Assume that swarm size is N, each particle’s position vector in D-dimensional space is $Xi = (x_{i1}, x_{i2}, \cdots , x_{id} , \cdots , x_{i D})$,
Velocity vector is $Vi = (v_{i1}, v_{i2}, \cdots , v_{id} , \cdots , v_{i D})$,
Individual’s optimal position (i.e., the optimal position that the particle has experienced) is $Pi = (p_{i1}, p_{i2}, \cdots , p_{id} , \cdots , p_{i D})$,
Swarm’s optimal position (i.e., the optimal position that any individual in this swarm has experienced) is represented as $Pg = (p_{g1}, p_{g2}, \cdots , p_{gd} , \cdots , p_{gD})$.
Without loss of generality, taking the minimizing problem as the example, in the initial version of the PSO algorithm, update formula of the individual’s optimal position is:

$$p^d_{i,t+1} = \begin{cases} x^d_{i,t+1}, & \text{if } f(X_{i,t+1}) < f(P_{i,t}) \\ p^d_{i,t}, & \text{otherwise} \end{cases} \tag{4.1}$$

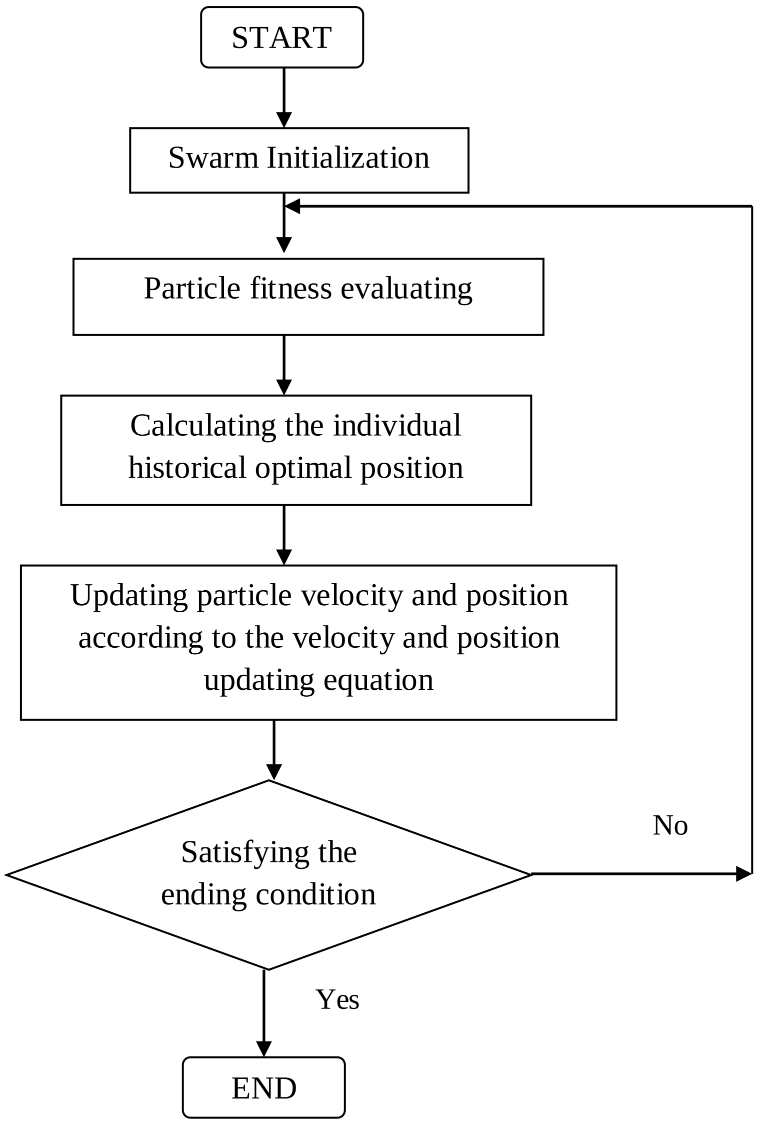
START
Swarm Initialization
Particle fitness evaluating
Calculating the individual historical optimal position
Updating particle velocity and position according to the velocity and position updating equation
Satisfying the ending condition
No
Yes
END

**Fig4.1:** Flow diagram illustrating the particle swarm optimization algorithm.

The swarm's optimal position is that of all the individual's optimal positions. Update formula of velocity and position is denoted as follows, respectively:

$$v^{d}_{i,t+1} = v^{d}_{i,t} + c1 * rand * (p^{d}_{i,t} - x^{d}_{i,t}) + c2 * rand * (p^{d}_{g,t} - x^{d}_{i,t}) \quad (4.2)$$

$$x^{d}_{i,t+1} = x^{d}_{i,t} + v^{d}_{i,t+1} \quad (4.3)$$

Since the initial version of PSO was not very effective in optimization problem, a modified PSO algorithm (Shi and Eberhart 1998) appeared soon after the initial algorithm was proposed. Inertia weight was introduced to the velocity update formula, and the new velocity update formula became:

$$v^{d}_{i,t+1} = w * v^{d}_{i,t} + c_1 * rand * (p^{d}_{i,t} - x^{d}_{i,t}) + c_2 * rand * (p^{d}_{g,t} - x^{d}_{i,t}) \quad (4.4)$$

Although this modified algorithm has almost the same complexity as the initial version, it has greatly improved the algorithm performance; therefore, it has

achieved extensive applications. Generally, the modified algorithm is called canonical PSO algorithm, and the initial version is called original PSO algorithm.
PSO algorithm has two versions, called global version and local version, respectively. In the global version, two extremes that the particles track are the optimal position *pbest* of its own and the optimal position *gbest* of the swarm. Accordingly, in local version, aside from tracking its own optimal position *pbest*, the particle does not track the swarm optimal position *gbest*, instead it tracks all particles' optimal position *nbest* in its topology neighbourhood. For the local version, the velocity update became Eq. (4.5), where $p_i$ was the optimal position in the local neighbourhood.
Analyzing the velocity update formula from a sociological perspective, we can see that in this update formula, the first part is the influence of the particle's previous velocity. It means that the particle has confidence on its current moving state and conducts inertial moving according to its own velocity, so parameter ω is called inertia weight. The second part depends on the distance between the particle's current position and its own optimal position, called the "cognitive" item. It means particle's own thinking, i.e., particle's move resulting from its own experience. Therefore, parameter $c_1$ is called cognitive learning factor (also called cognitive acceleration factor). The third part relies on the distance between the particle's current position and the global (or local) optimal position in the swarm, called "social" factor. It means the information share and cooperation among the particles, namely particle's moving coming from other particles' experience in the swarm. It simulates the move of good particle through the cognition, so the parameter $c_2$ is called social learning factor (also called social acceleration factor).

## 4.3 Objective function of the Particle Swarm Algorithm

Here, minimization of integral-time-absolute-error (ITAE) is defined as the objective function (performance index or fitness function). The ITAE is calculated as:

$$\text{ITAE} = \int_0^t t * |e(t)| dt$$

## 4.4 Particle Swarm Algorithm Parameters

| | |
|---|---|
| Population Size | 50 |
| No. Of iterations | 100 |
| Inertia coefficient | 0.9 |
| Personal acceleration coefficient | 0.12 |
| Global/social acceleration coefficient | 1.2 |
| Damping ration of inertia coefficient | 0.99 |
| Range of Variables | 0-200% of initial parameters |

**Table 4.1:** Particle Swarm Algorithm Parameters

## 4.5 Steps of PSO Algorithm

The pseudo code of the PSO algorithm is stated as follows:

```
FOR each particle i
  FOR each dimension d
      Initialize position $x_{id}$ randomly within permissible range
      Initialize velocity $v_{id}$ randomly within permissible range
   End FOR
End FOR
Iteration k=1
DO
   FOR each particle i
         Calculate fitness value
         IF the fitness value is better than $p\_best_{id}$ in history
           Set the current fitness value as the $p\_best_{id}$
         End IF
   End FOR
Choose the particle having the best fitness value as the $g\_best_d$
FOR each particle i
  FOR each dimension d
         Calculate velocity according to the equation
         $V_{id}(k+1)=w*v_{id}(k)+c1*rand1(p_{id}-x_{id})+c2*rand2(p_{gd}-x_{id})$
         Update particle position according to the equation
         $X_{id}(K+1)=x_{id}(k)+v_{id}(k+1)$
   End FOR
End FOR
K=k+1
WHILE maximum iterations or minimum error criteria are not attained
```

At first, in the for loops, we have initialized the particles' positions with a random uniform distribution within a permissible range for all its dimensions. After that, for each particle, it calculates its fitness value and compared with his own best position (The *p_best* value is the best position of that specific particle has ever been) and then it chooses the best position of all particles in *g_best*. Let us take a closer look to the equation that defines the velocity of the next iteration of a particle dimension [7]:

- $V_i$ (k+1) is the next iteration velocity, w is an inertial parameter. This parameter affects the movement propagation given by the last velocity value.
- $c_1$ and $c_2$ are acceleration coefficients. $c_1$ value gives the importance of personal best value and $c_2$ is the importance of social best value.
- $p_i$ is the best individual position and $p_g$ is the best position of all particles. In the equation, is measured the distance of each of these parameters to the particle's actual position.
- $rand_1$ and $rand_2$ are random numbers where $0 \leq rand \leq 1$ and they control the influence of each value: Social and individual as shown in the next Fig 4.2.

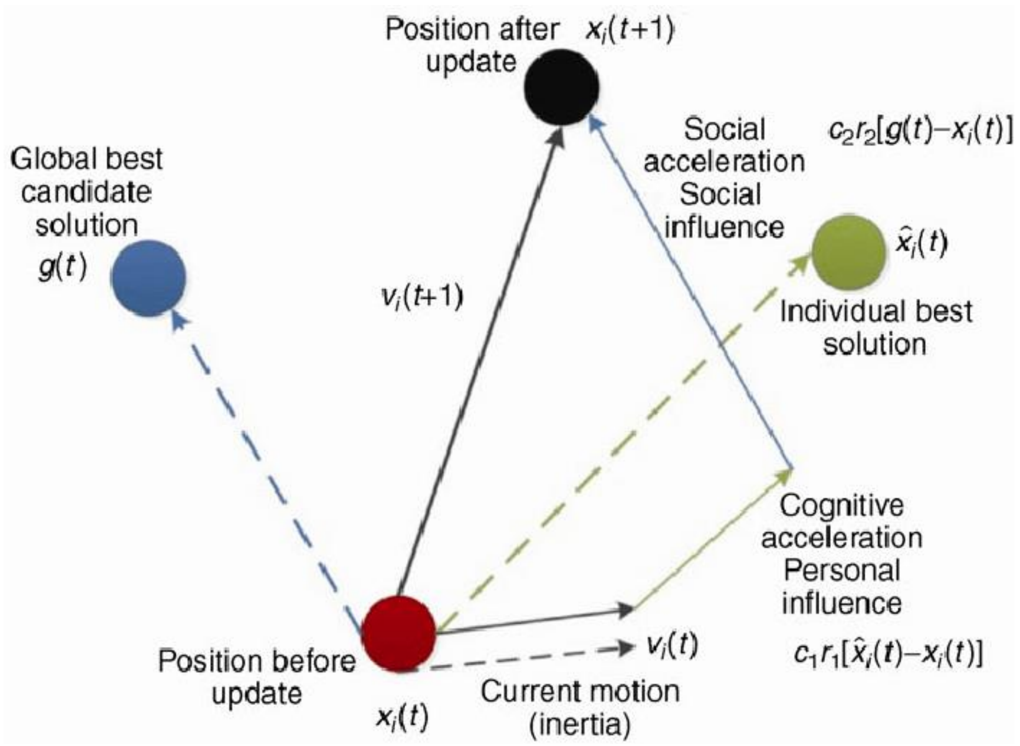
Position after update $x_i(t+1)$
Global best candidate solution $g(t)$
Social acceleration Social influence
$c_2r_2[g(t)-x_i(t)]$
$\hat{x}_i(t)$
Individual best solution
$v_i(t+1)$
Cognitive acceleration Personal influence
$v_i(t)$
$c_1r_1[\hat{x}_i(t)-x_i(t)]$
Position before update
$x_i(t)$
Current motion (inertia)

**Fig 4.2:** Illustration of velocity and position updates in PSO Algorithm

After that is calculated the new particle's position until the number of iterations specified or an error criteria be reached.

A typical Convergence plot of the objective function vs. Iteration is shown below:

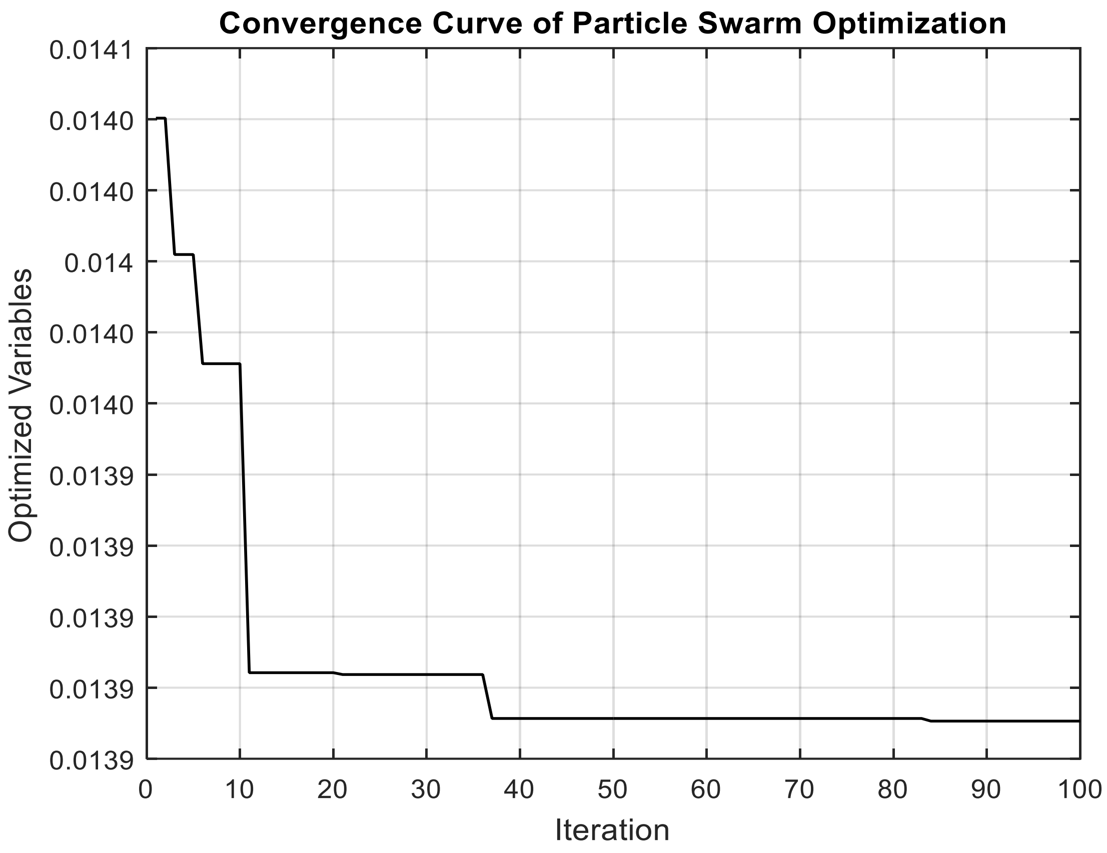

**Fig 4.3:** Convergence Curve for QUBE-Servo2

## 4.6 Results

For simulation and performance study the process transfer function of the identified model and the process itself of both QUBE-Servo2and SRV02 are being considered.

For QUBE-Servo2, the voltage-to-position transfer function is

$$P(s) = \frac{\theta_m(s)}{V_m(s)} = \frac{22.7}{s(0.158s+1)}$$

For Rotary Servo Base unit SRV02, the voltage-to-position transfer function is

$$P(s) = \frac{\theta_m(s)}{V_m(s)} = \frac{1.53}{s(0.0253s+1)}$$

We have derived the close loop response characteristics for the identified process model by using different controllers. For detailed comparison, in addition to the response characteristics, several performance indices, such as percentage overshoot (%OS),rise time ($T_r$), settling time($T_s$), integral absolute error (IAE), integral time absolute error (ITAE), integral square error (ISE) are calculated for each controller. Performance of PSO-PD is compared with the corresponding PD controller. Simpson's1/3rd rule is used for numerical integration. The detailed performance analysis is discussed next.

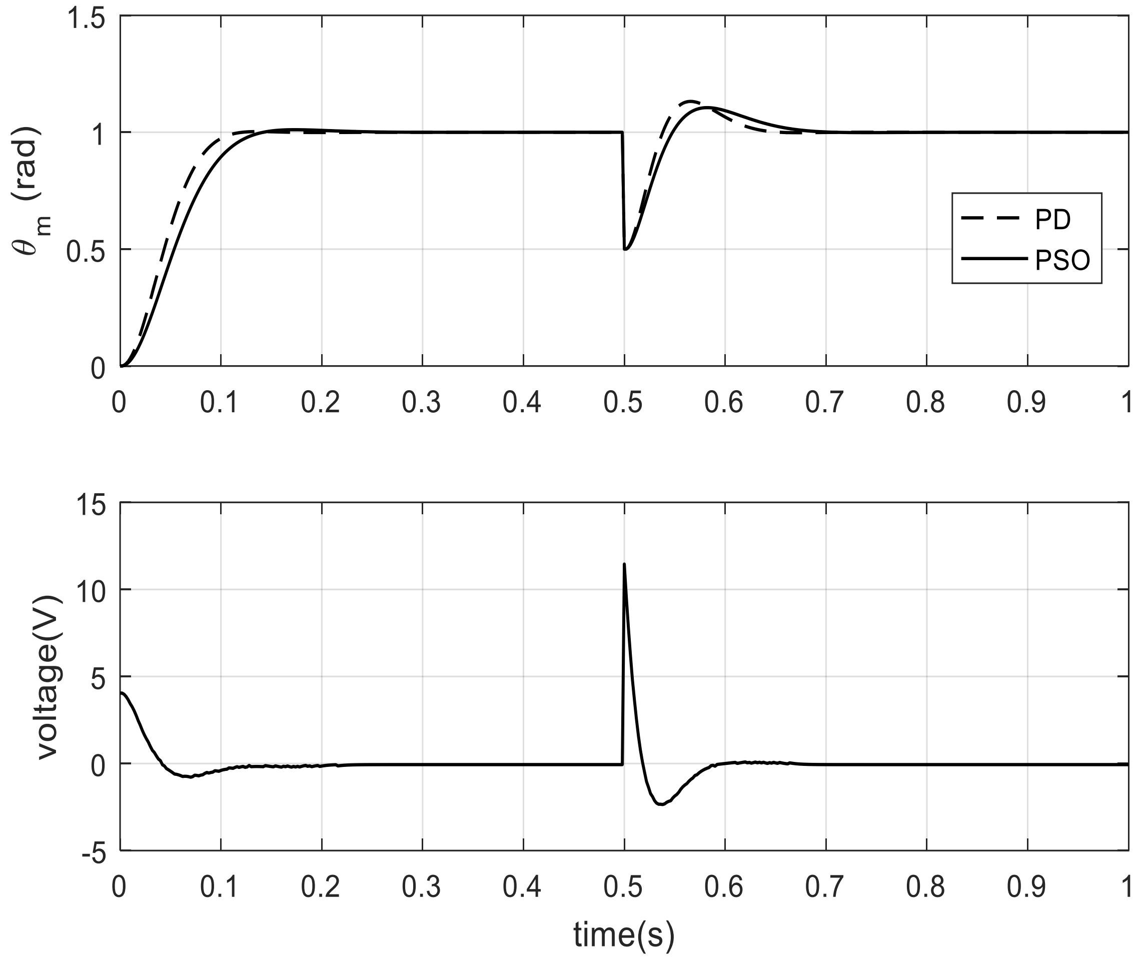

**Fig 4.4:** Responses of Motor Speed of PD and PSO based controllers of the QUBE-Servo2identified model and its corresponding Motor voltage

| **Objective Function** | **QUBE Servo 2 Model Characteristics** | **PSO** | **PD** |
|---|---|---|---|
| ITAE | Rise Time (ms) | 80.448 | 64.010 |
| | %Overshoot | 1.065 | 0.117 |
| | Peak Time(sec) | 0.174 | 0.136 |
| | ITAE | 0.01374 | 0.01135 |
| | IAE | 0.08029 | 0.06633 |
| | ISE | 0.04684 | 0.03891 |

**Table 4.2:** Performance Table of Controllers of QUBE-Servo2Model

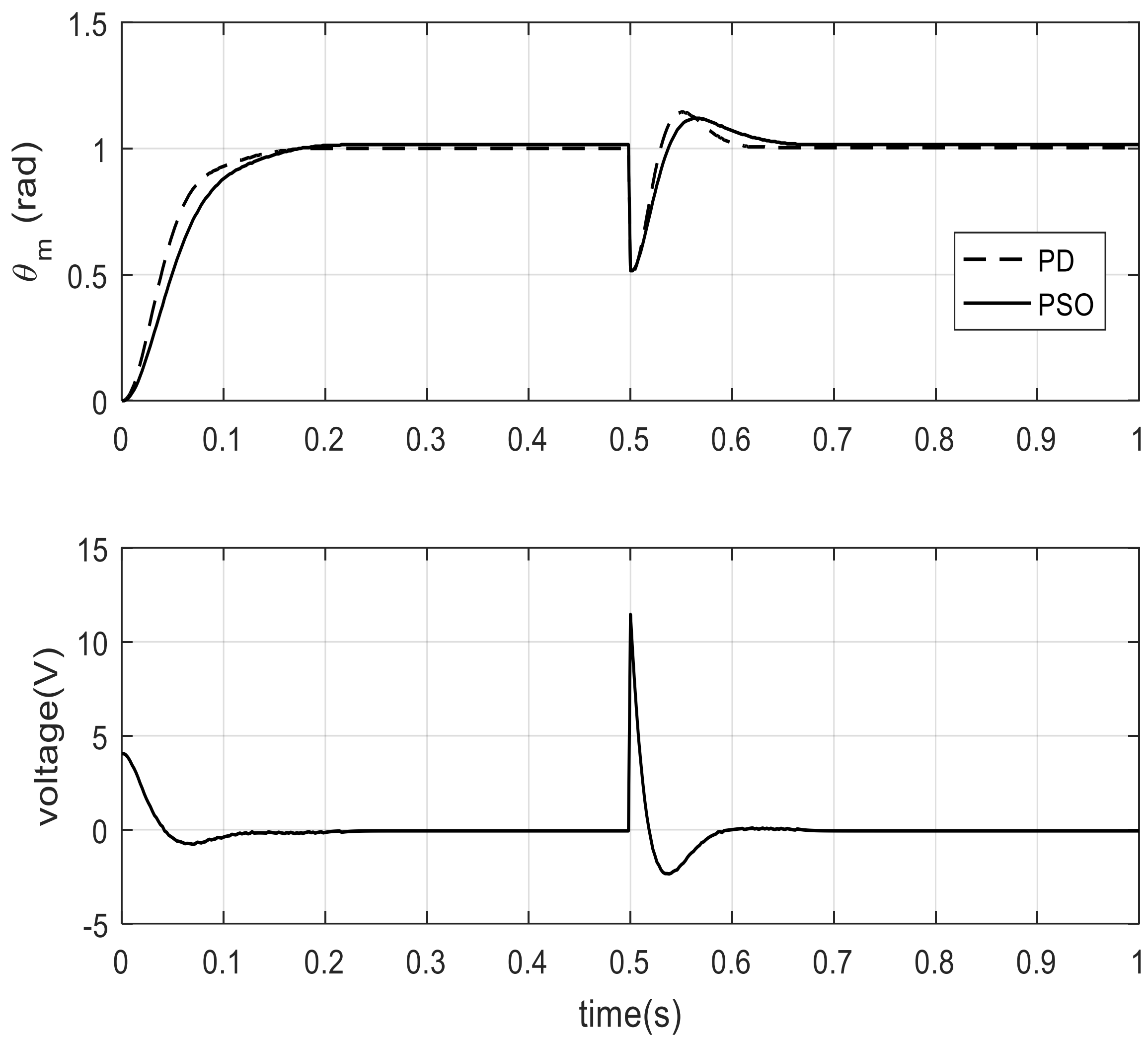

**Fig 4.5:** Responses of Motor Speed of PD and PSO based controllers of the QUBE-Servo2 and its corresponding Motor voltage

| Objective Function | QUBE Servo 2 Characteristics | PSO | PD |
|---|---|---|---|
| ITAE | Rise Time (ms) | 92.731 | 70.762 |
| | %Overshoot | 0.198 | -0.122 |
| | Peak Time(sec) | 0.216 | 0.178 |
| | ITAE | 0.01827 | 0.01074 |
| | IAE | 0.08607 | 0.06353 |
| | ISE | 0.043 | 0.03474 |

**Table 4.3:** Performance Table of Controllers of QUBE-Servo2

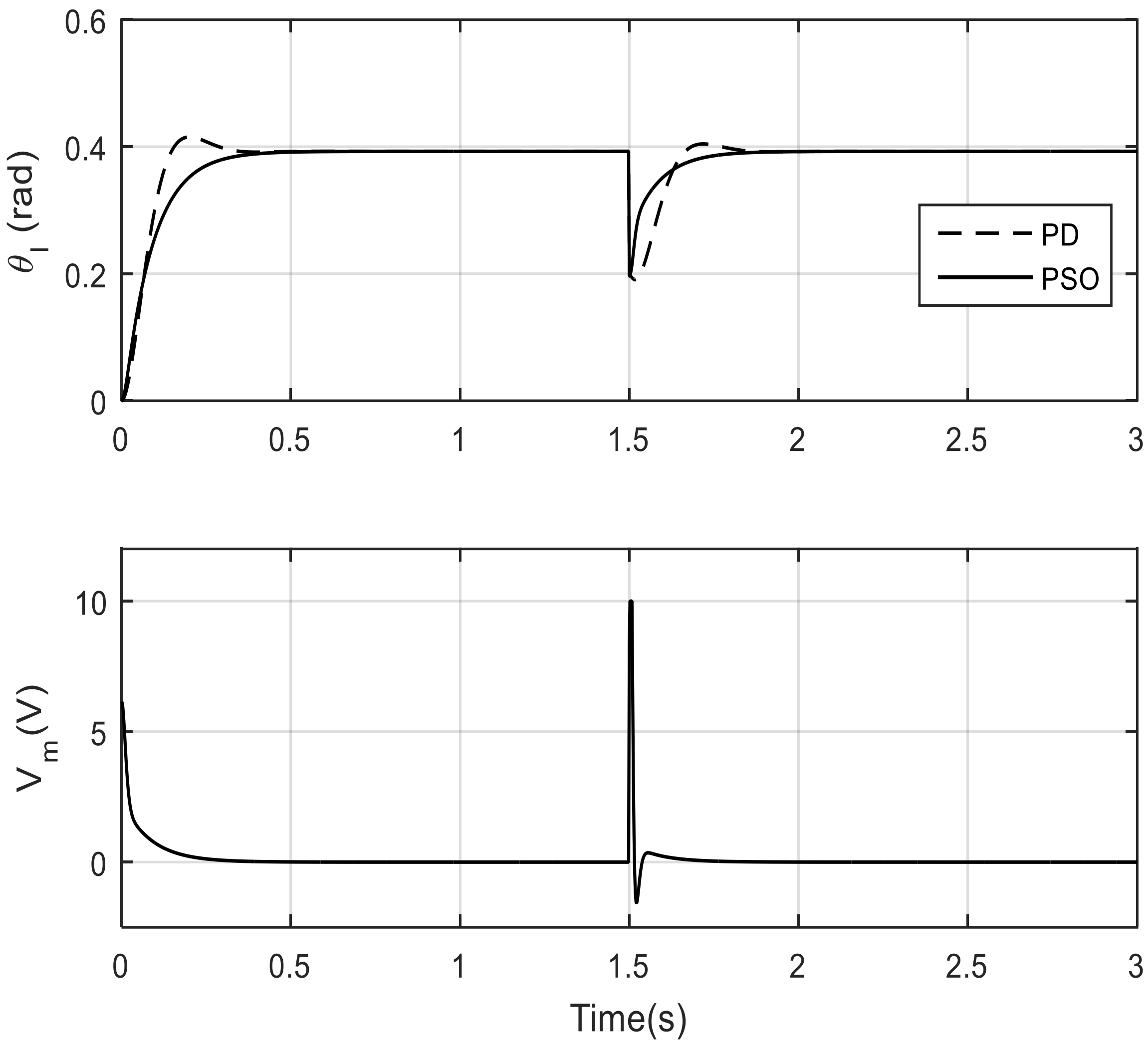

**Fig 4.6:** Responses of Motor Speed of PD and PSO based controllers of the Rotary servo base SRV02 identified model and its corresponding Motor voltage

| **Objective Function** | **Rotary servo base SRV02 model Characteristics** | **PSO** | **PD** |
|---|---|---|---|
| ITAE | Rise Time (ms) | 182.415 | 78.132 |
| | %Overshoot | 0.501 | 8.717 |
| | Peak Time(sec) | 1.498 | 0.202 |
| | ITAE | 0.02266 | 0.03135 |
| | IAE | 0.04901 | 0.04871 |
| | ISE | 0.008904 | 0.01033 |

**Table 4.4:** Performance Table of Controllers of Rotary servo base SRV02 model

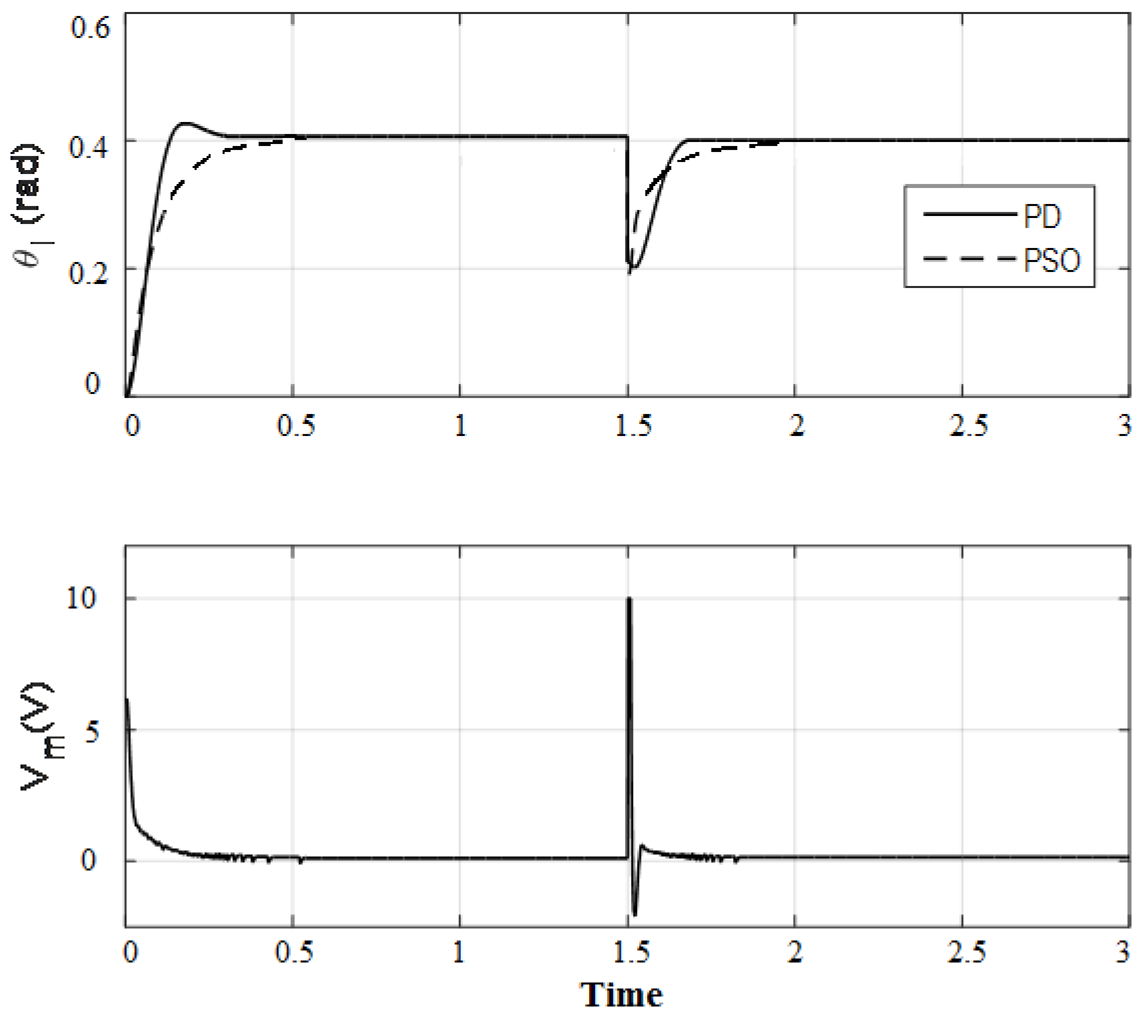

**Fig 4.7:** Responses of Motor Speed of PD and PSO based controllers of the Rotary servo base SRV02 and its corresponding Motor voltage

| Objective Function | Rotary servo base SRV02 Characteristics | PSO | PD |
|---|---|---|---|
| ITAE | Rise Time (ms) | 182.415 | 72.083 |
| | %Overshoot | 0.501 | 7.484 |
| | Peak Time(sec) | 1.498 | 0.166 |
| | ITAE | 0.02266 | 0.06585 |
| | IAE | 0.04901 | 0.07305 |
| | ISE | 0.008904 | 0.01021 |

**Table 4.5:** Performance Table of Controllers of Rotary servo base SRV02

## 4.7 Conclusion

Here, we have explored the possibility of performance enhancement of position control by Particle Swarm Optimization technique. From the simulation results we observed that the actual process model of QUBE-Servo2 and rotary servo base unit SRV02 produces good result due to set point changes as well as load disturbances.

## References:

[1] R. C. Eberhart and J. Kennedy, "A new optimizer using particle swarm theory", in Proceedings of the sixth international symposium on micro machine and human science, 1995.

[2] J. Kennedy and R. C. Eberhart, "Particle swarm optimization", in Proceedings of the IEEE International Conference on Neural Networks, 1995.

[3] J. Kennedy, "Small Worlds and Mega-Minds: Effects of Neighborhood Topology on Particle Swarm Performance", in Proceedings of the 1999 Congress on Evolutionary Computation, 1999.

[4] J. Kennedy and R. C. Eberhart and Y. Shi, "Swarm Intelligence", Morgan Kaufmann, 2001.

[5] R. Poli and J. Kennedy and T. Blackwell, "Particle swarm optimization an overview", Swarm Intelligence, 2007.

[6] R. Poli, "Analysis of the publications on the applications of particle swarm optimisation", Journal of Artificial Evolution and Applications, 2008.

[7] C. W. Reynolds, "Flocks, herds and schools: A distributed behavioural model", in Proceedings of the 14th annual conference on Computer graphics and interactive techniques, 1987.

[8] Y. Shi and R. C. Eberhart, "A Modified Particle Swarm Optimizers", in Proceedings of the IEEE International Conference on Evolutionary Computation, 1998.

# Chapter 5

Optimal PD Controller using Moth Flame Optimization (MFO)

## 5.1 Introduction

Optimization refers to the process of finding the best possible solution for a particular problem. As the complexity of problems increases, over the last few decades, the need for new optimization techniques becomes evident more than before. Mathematical optimization techniques used to be the only tools for optimizing problems before the proposal of heuristic optimization techniques. Mathematical optimization methods are mostly deterministic that suffer from one major problem: local optima entrapment. Some of them such as gradient-based algorithms require derivation of the search space as well. This makes them highly inefficient in solving real problems. Moth-Flame Optimization (MFO) algorithm was proposed in 2016 [1] by Seyedali Mirjalili, as one of the seminal attempt to simulate the navigation of moths in computer and propose an optimization algorithm. This algorithm has been widely used in science and industry.

## 5.2. Inspiration

Moths are fancy insects, which are highly similar to the family of butterflies. Basically, there are over 160,000 various species of this insect in nature. They have two main milestones in their lifetime: larvae and adult. The larva is converted to moth by cocoons. The most interesting fact about moths is their special navigation methods in night. They have been evolved to fly in night using the moon light. They utilized a mechanism called transverse orientation for navigation. In this method, a moth flies by maintaining a fixed angle with respect to the moon, a very effective mechanism for travelling long distances in a straight path [2]. Since the moon is far away from the moth, this mechanism guarantees flying in straight line. The same navigation method can be done by humans. Suppose that the moon is in the south side of the sky and a human wants to go the east. If he keeps moon of his left side when walking, he would be able to move toward the east on a straight line.

Despite the effectiveness of transverse orientation, we usually observe that moths fly spirally around the lights. In fact, moths are tricked by artificial lights and show such behaviours. This is due to the inefficiency of the transverse orientation, in which it is only helpful for moving in straight line when the light source is very far. When moths see a human-made artificial light, they try to maintain a similar angle with the light to fly in straight line. Since such a light is extremely close compared to the moon, however, maintaining a similar angle to the light source causes a useless or deadly spiral fly path for moths [3]. It may be observed in Fig. 5.1 that the moth eventually converges towards the light. This behavior is modeled mathematically to propose an optimizer called Moth-Flame Optimization (MFO) algorithm in the following subsection.

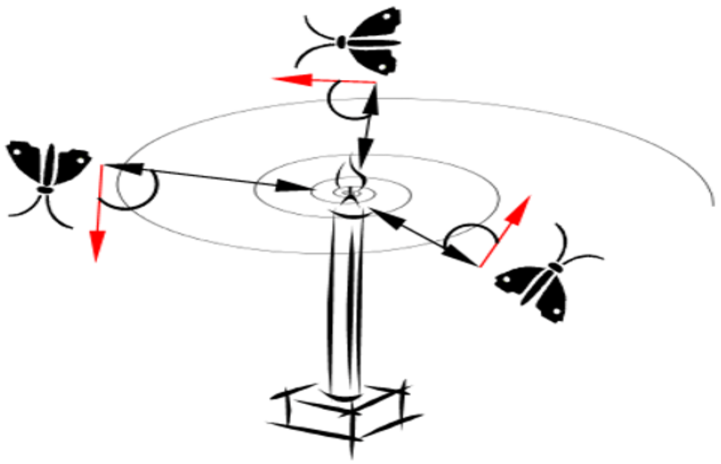

**Fig 5.1:** Spiral flying path around close light source [1]

## 5.3 MFO algorithm [1]

In the MFO algorithm, it is assumed that the candidate solutions are moths and the problem's variables are the position of moths in the space. Therefore, the moths can fly in 1-D, 2-D, 3-D, or hyper dimensional space with changing their position vectors. Since the MFO algorithm is a population-based algorithm, the set of moths is represented in a matrix as follows:

$$M = \begin{bmatrix} m_{1,1} & \cdots & m_{1,d} \\ \vdots & \ddots & \vdots \\ m_{n,1} & \cdots & m_{n,d} \end{bmatrix}$$

Where $n$ is the number of moths and $d$ is the number of variables (dimension).For all the moths, we also assume that there is an array for storing the corresponding fitness values as follows:

OM = $\begin{bmatrix} OM_1 \\ OM_2 \\ \vdots \\ OM_n \end{bmatrix}$ ; where *n* is the number of moths.

The fitness value is the return value of the fitness (objective) function for each moth. The position vector (first row in the matrix M for instance) of each moth is passed to the fitness function and the output of the fitness function is assigned to the corresponding moth as its fitness value ($OM_1$ in the matrix OM for instance). Another key component in the proposed algorithm are flames. A matrix similar to the moth matrix is considered as follows:

$$F = \begin{bmatrix} F_{1,1} & F_{1,2} & \dots & F_{1,d} \\ F_{2,1} & F_{2,2} & \cdots & F_{2,d} \\ \vdots & \dots & \ddots & \vdots \\ F_{n,1} & F_{n,2} & \cdots & F_{n,d} \end{bmatrix}$$

where *n* is the number of moths and *d* is the number of variables (dimension).
It may be seen in the above equation that the dimensions of M and F arrays are equal. For the flames, it is also assumed that there is an array for storing the corresponding fitness values as follows:

OF = $\begin{bmatrix} OF_1 \\ OF_2 \\ \vdots \\ OF_n \end{bmatrix}$ ; where *n* is the number of moths.

The moths are actual search agents that move around the search space, whereas flames are the best position of moths that obtains so far. Therefore, each moth searches around a flag (flame) and updates it in case of finding a better solution. With this mechanism, a moth never loses its best solution.
The MFO algorithm is a three-tuple that approximates the global optimal of the optimization problems and defined as follows:
MFO = (I, P, T)
I is a function that generates a random population of moths and corresponding fitness values. The methodical model of this function is as follows:
$I : \varphi = \{M, OM\}$
The P function, which is the main function, moves the moths around the search space. This function received the matrix of M and returns its updated one eventually.
$P : M = M$
The T function returns true if the termination criterion is satisfied and false if the termination criterion is not satisfied:
$T : M = \{true; false\}$

With I,P, and T, the general framework of the MFO algorithm is defined as follows:

```
M = I();
while T(M) is equal to false
M = P(M);
End
```

The function I have to generate initial solutions and calculate the objective function values. Any random distribution can be used in this function. The following method is utilized as the default:

```
for i = 1: n
for j = 1: d
M(i, j) = (ub(i) _ lb(i))∕rand() + lb(i);
end
end
OM = Fitness Function (M);
```

As can be seen, there are two other arrays called ub and lb. These matrixes define the upper and lower bounds of the variables as follows:

$ub =[ub_1, ub_2, ub_3, . . ., ub_n]$

where ub(i) indicates the upper bound of the i-th variable.

$lb =[lb_1, lb_2,lb_3, . . . , lb_n]$

where lb(i) indicates the lower bound of the i-th variable.

After the initialization, the P function is iteratively run until the T function returns true. The P function is the main function that moves the moths around the search space. As mentioned above the inspiration of this algorithm is the transverse orientation. In order to mathematically model this behaviour, the position of each moth is updated with respect to a flame using the following equation:

$M_i = S(M_i,F_j)$

where Mi indicate the i-th moth, $F_j$ indicates the j-th flame, and S is the spiral function. The exploration of the search space around the best locations obtained so far is guaranteed with this method due to the following reasons:

- Moths update their positions in hyper spheres around the best solutions obtained so far.
- The sequence of flames is changed based on the best solutions in each iteration, and the moths are required to update their positions with respect to the updated flames. Therefore, the position updating of moths may occur around different flames, a mechanism that causes sudden movement of moths in the search space and promotes exploration.

Another concern here is that the position updating of moths with respect to n different locations in the search space may degrade the exploitation of the best promising solutions. To resolve this concern, an adaptive mechanism is proposed for the number of flames. Fig. 5.2 shows that how the number of

flames is decreased adaptively over the course of iterations. The following formula is utilized in this regard:
flame no = round (N-*l**(N-1)/T) .....(a)

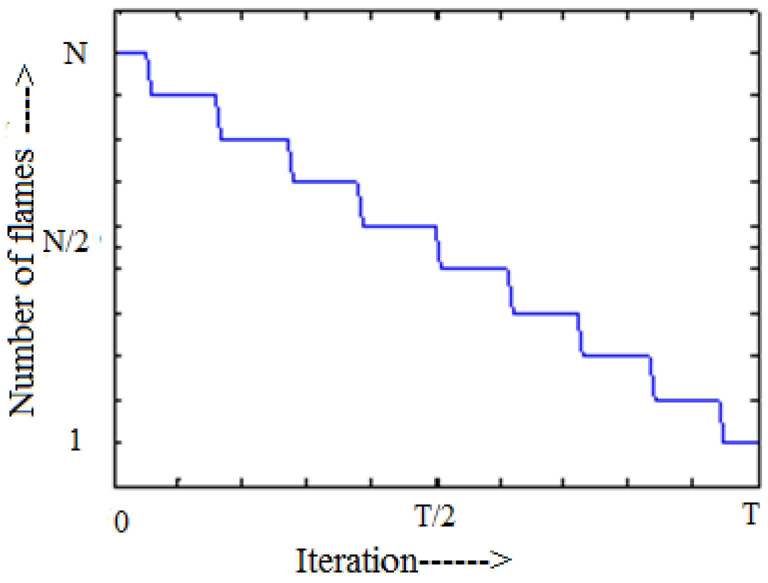

**Fig 5.2:** Number of flame is decreased adaptively over the course of iterations

where l is the current number of iteration, N is the maximum number of flames, and T indicates the maximum number of iterations. Fig. 5.2 shows that there is N number of flames in the initial steps of iterations. However, the moths update their positions only with respect to the best flame in the final steps of iterations. The gradual decrement in number of flames balances exploration and exploitation of the search space.

## 5.4 Objective function of the Moth Flame Algorithm

Here, minimization of integral-time-absolute-error (ITAE) is defined as the objective function (performance index or fitness function). The ITAE is calculated as:

ITAE = $\int_0^t t * |e(t)| dt$

## 5.5 Moth Flame Algorithm Parameters

| | |
|---|---|
| Search agent No. | 30 |
| No. Of iterations | 100 |
| Dimension (No. Of Variables) | 2 |
| Range of Variables | 0-200% of initial parameters |

**Table 5.1:** Moth Flame Algorithm Parameters

## 5.6 Steps of MFO Algorithm [1]

After all, the general steps of the P function are as follows.

```
Update flame no using Eq. (a)
OM = Fitness Function (M);
if iteration == 1
F = sort(M);
OF = sort(OM);
else
F = sort(Mt_1, Mt);
OF = sort(Mt_1, Mt);
end
for i = 1: n
for j = 1: d
Update r and t
Calculate D with respect to the corresponding moth
Update M(i,j) using with respect to the corresponding moth
end
end
```

As discussed above, the P function is executed until the function returns true. After termination of the P function, the best moth is returned as the best obtained approximation of the optimum.

## 5.7 Results:

For simulation and performance study the process transfer function of the identified model and the process itself of both QUBE-Servo2and SRV02 are being considered.

For QUBE-Servo2, the voltage-to-position transfer function is

$$P(s) = \frac{\theta_m(s)}{V_m(s)} = \frac{22.7}{s(0.158s+1)}$$

For Rotary Servo Base unit SRV02, the voltage-to-position transfer function is

$$P(s) = \frac{\theta_m(s)}{V_m(s)} = \frac{1.53}{s(0.0253s+1)}$$

We have observed the close loop response characteristics for the identified process model by using different controllers. For detailed comparison, in addition to the response characteristics, several performance indices, such as percentage overshoot (%OS),rise time ($T_r$), settling time ($T_s$), integral absolute error (IAE), integral time absolute error (ITAE), integral square error (ISE) are calculated for each controller. Performance of MFO-PD is compared with the corresponding PD controller. Simpson's1/3rd rule is used for numerical integration. The detailed performance analysis is discussed next.

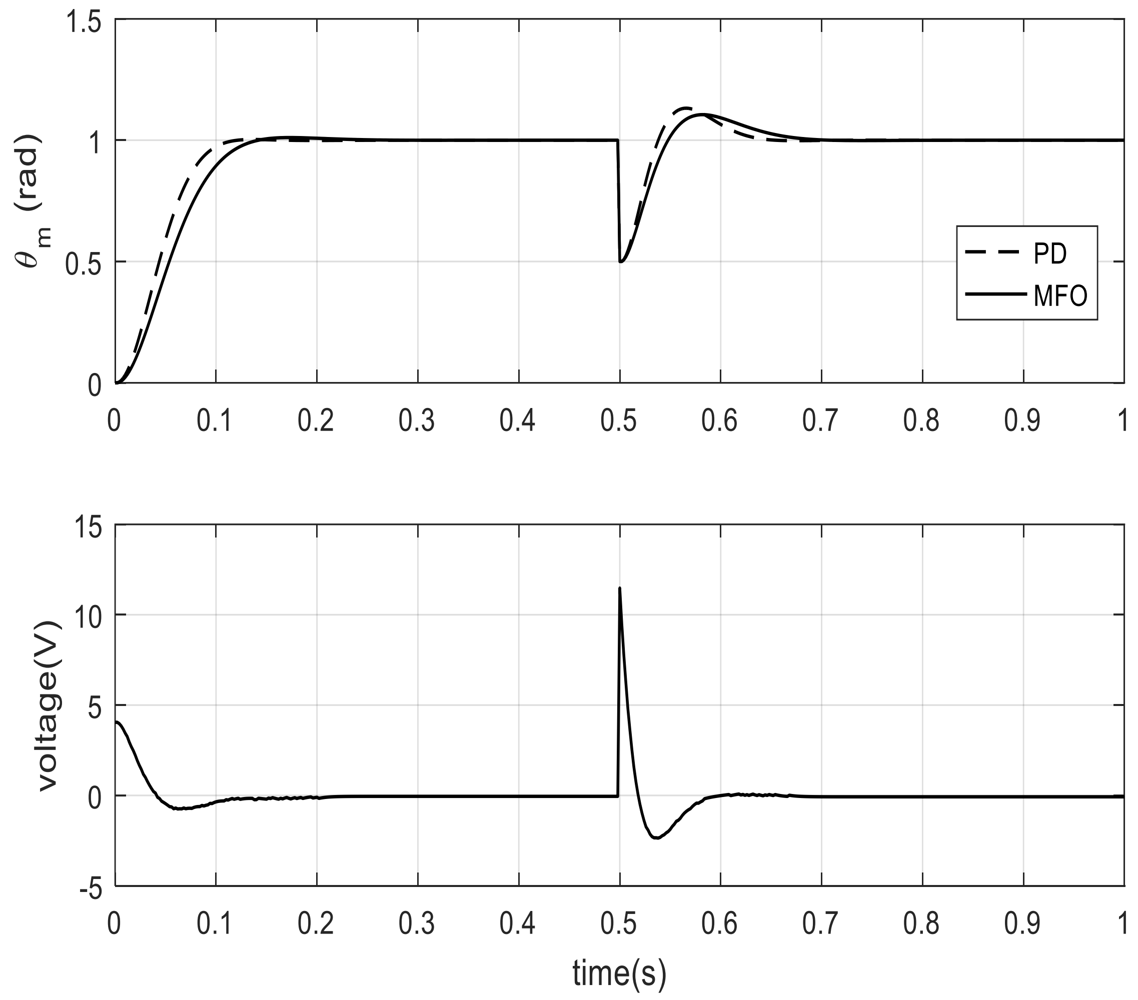

**Fig 5.3:** Responses of Motor Speed of PD and MFO based controllers of the QUBE-Servo2 identified model and its corresponding Motor voltage

| **Objective Function** | **QUBE-Servo2 Model Characteristics** | **MFO** | **PD** |
|---|---|---|---|
| ITAE | Rise Time (ms) | 80.423 | 64.010 |
| | %Overshoot | 1.058 | 0.117 |
| | Peak Time(sec) | 0.174 | 0.136 |
| | ITAE | 0.01373 | 0.01135 |
| | IAE | 0.08026 | 0.06633 |
| | ISE | 0.04682 | 0.03891 |

**Table 5.2:** Performance Table of Controllers of QUBE-Servo2 Model

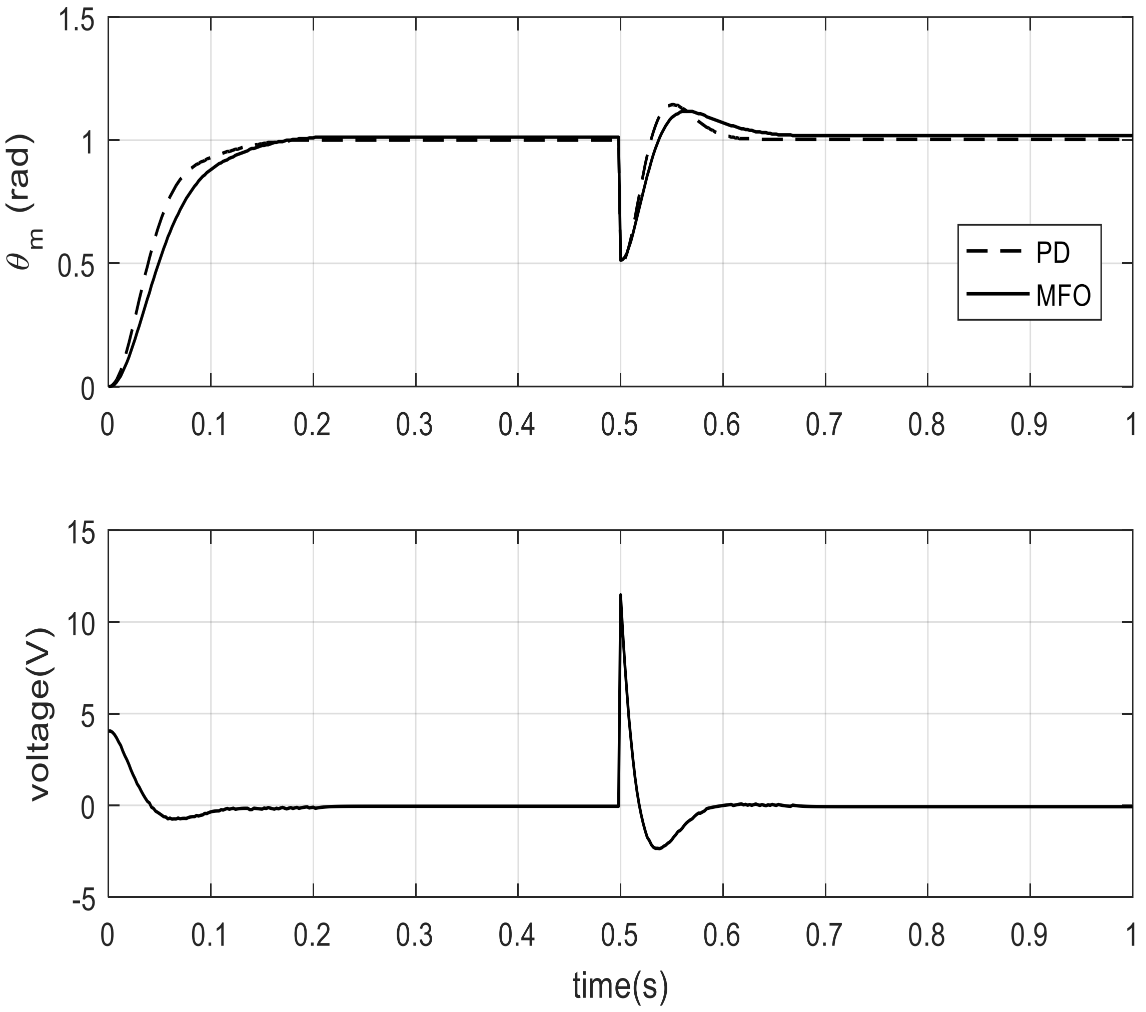

**Fig 5.4:** Responses of Motor Speed of PD and MFO based controllers of the QUBE-Servo2and its corresponding Motor voltage

| **Objective Function** | **QUBE-Servo2 Characteristics** | **MFO** | **PD** |
|---|---|---|---|
| ITAE | Rise Time (ms) | 97.072 | 70.762 |
| | %Overshoot | -0.931 | -0.122 |
| | Peak Time(sec) | 0.202 | 0.178 |
| | ITAE | 0.01883 | 0.01074 |
| | IAE | 0.08601 | 0.06353 |
| | ISE | 0.04284 | 0.03474 |

**Table 5.3:** Performance Table of Controllers of QUBE-Servo2

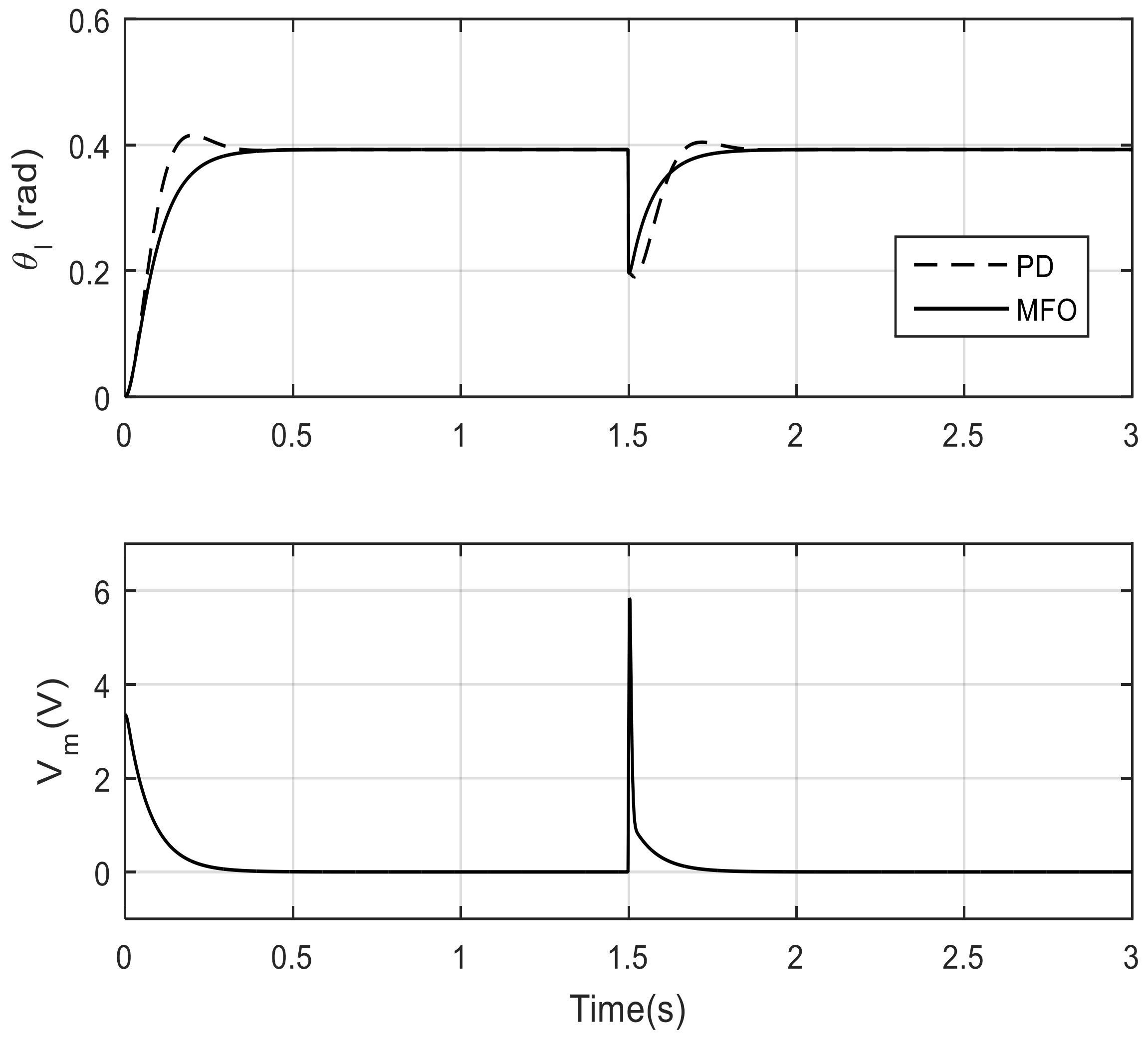

**Fig 5.5:** Responses of Motor Speed of PD and MFO based controllers of the Rotary servo base SRV02 identified model and its corresponding Motor voltage

| **Objective Function** | **Rotary servo base SRV02 model Characteristics** | **MFO** | **PD** |
|---|---|---|---|
| ITAE | Rise Time (ms) | 170.96 | 78.132 |
| | %Overshoot | 0.503 | 8.717 |
| | Peak Time(sec) | 0.12 | 0.202 |
| | ITAE | 0.02668 | 0.03135 |
| | IAE | 0.05342 | 0.04871 |
| | ISE | 0.01055 | 0.01033 |

**Table 5.4:** Performance Table of Controllers of Rotary servo base SRV02 model

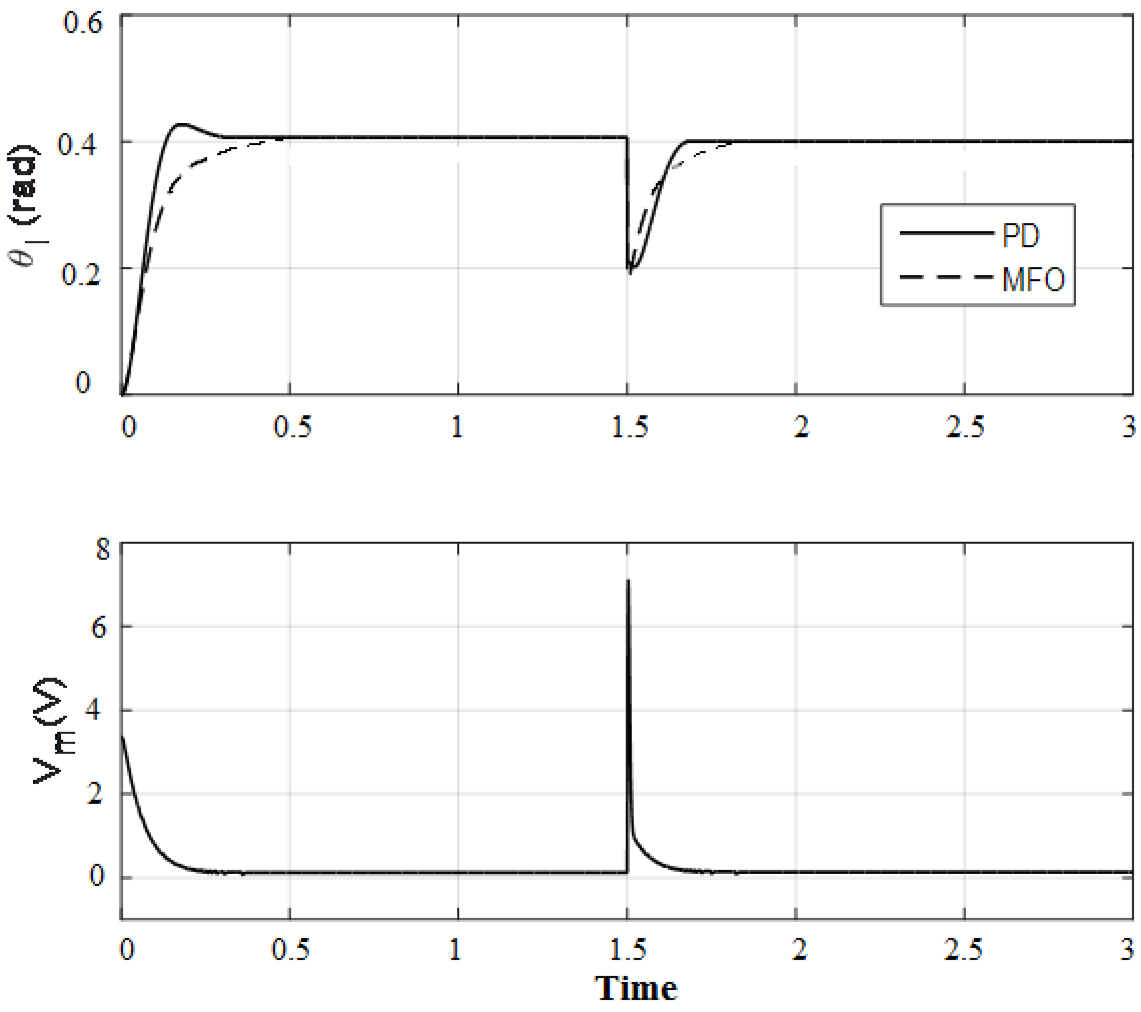

**Fig 5.6:** Responses of Motor Speed of PD and MFO based controllers of the Rotary servo base SRV02 and its corresponding Motor voltage

| **Objective Function** | **Rotary servo base SRV02 Characteristics** | **MFO** | **PD** |
|---|---|---|---|
| ITAE | Rise Time (ms) | 122.722 | 72.083 |
| | %Overshoot | -0.631 | 7.484 |
| | Peak Time(sec) | 0.357 | 0.166 |
| | ITAE | 0.08878 | 0.06585 |
| | IAE | 0.08879 | 0.07305 |
| | ISE | 0.01103 | 0.01021 |

**Table 5.5:** Performance Table of Controllers of Rotary servo base SRV02

## 5.8 Conclusion

Here, we have explored the possibility of performance enhancement of servo position control by Moth Flame Optimization technique. From the simulation results we observed that the actual process model of QUBE Servo and rotary servo base unit produces good result due to set point changes as well as load disturbances. In tables, results for the best optimized variables $K_p$ and $K_d$ are shown. A typical Convergence plot of the objective function vs. Iteration is shown below:

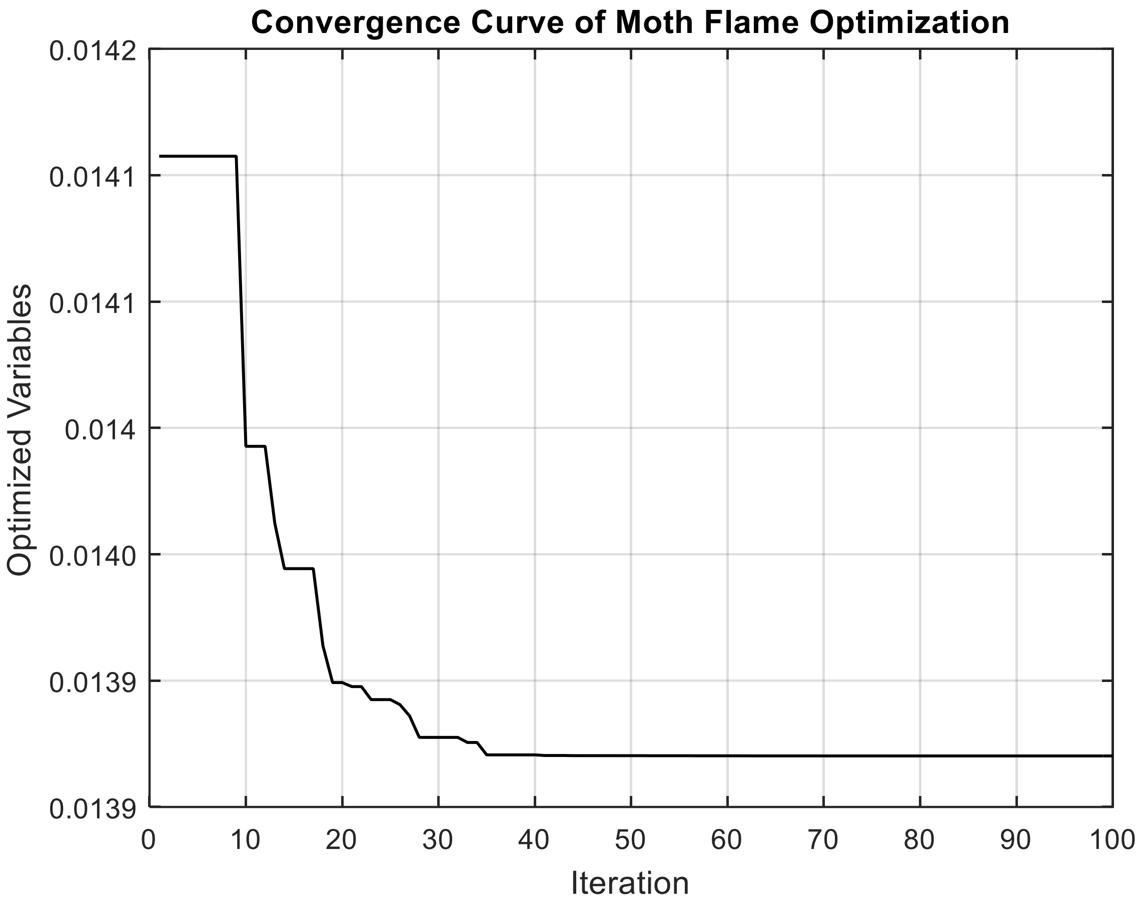

**Fig 5.7:** Convergence curve for QUBE-Servo2

## References

[1] Mirjalili, Seyedali. "Moth-flame optimization algorithm: A novel nature-inspired heuristic paradigm." Knowledge-Based Systems 89 (2015): 228-249.

[2] Gaston, Kevin J., et al. "The ecological impacts of night time light pollution: a mechanistic appraisal." Biological reviews 88.4 (2013): 912-927

[3] Frank, Kenneth D. "Effects of artificial night lighting on moths." Ecological consequences of artificial night lighting (2006): 305-344.

[4] M. Mahdavi, M. Fesanghary, E. Damangir, An improved harmony search algorithm for solving optimization problems, Appl. Math. Comput. 188 (2007)1567–1579

[5] A.Y. Lam, V.O. Li, Chemical-reaction-inspired metaheuristic for optimization,IEEE Trans. Evol. Comput. 14 (2010) 381–399.

[6] B. Alatas, A novel chemistry based metaheuristic optimization method for mining of classification rules, Expert Syst. Appl. 39 (2012) 11080–11088.

[7] Gutjahr WJ (2009) Convergence analysis of metaheuristics. In: Matheuristics. Springer, Boston, pp 159–187

[8] Bonabeau E, Dorigo M, Theraulaz G (1999) Swarm intelligence: from natural to artificial systems (No. 1). Oxford University Press, New York

[9] Storn R, Price K (1995) Differential evolution—a simple and efficient adaptive scheme for global optimization over continuous spaces, vol 3. ICSI, Berkeley4.

[10] Simon D (2008) Biogeography-based optimization. IEEE Trans Evol Comput 12(6):702–71

[11] Rechenberg I (1978) Evolutions strategien. In: Schneider B, Ranft U (eds) Simulationsm ethoden in der Medizin und Biologie. Springer, Berlin, pp 83–114

[12] Muangkote N, Sunat K, Chiewchanwattana S (2016) Multilevel thresholding for satellite image segmentation with moth-flame based optimization. In: 2016 13th international joint conference on computer science and software engineering (JCSSE). IEEE, pp 1–6

[13] Bentouati Bachir, Chaib Lakhdar, Chettih Saliha (2016) Optimal power flow using the moth flam optimizer: a case study of the algerian power system. Indones J Electr Eng Comput Sci 1(3):431–445

[14] Yamany W, Fawzy M, Tharwat A, Hassanien AE (2015) Moth-flame optimization for training multi-layer perceptrons. In: 2015 11th international computer engineering conference (ICENCO). IEEE, pp 267–272

[15] Li Z, Zhou Y, Zhang S, Song J (2016) Lévy-flight moth-flame algorithm for function optimization and engineering design problems. Math Probl Eng. .

[16] Garg P, Gupta A (2016) Optimized open shortest path first algorithm based on moth flame optimization. Indian J Sci Technol. 29.

[17] Derrac J, García S, Molina D, Herrera F (2011) A practical tutorial on the use of nonparametric statistical tests as a methodology for comparing Evolutionary and swarm intelligence algorithms.

# Chapter 6

Optimal PD Controller using Grey Wolf optimization (GWO)

## 6.1 Introduction

The GWO algorithm mimics the leadership hierarchy and hunting mechanism of gray wolves in nature proposed by Mirjalili et al. in 2014.[1] Four types of grey wolves such as alpha, beta, delta, and omega are employed for simulating the leadership hierarchy. In addition, three main steps of hunting, searching for prey, encircling prey, and attacking prey, are implemented to perform optimization. Meta-heuristic optimization techniques have become very popular over the last two decades. Surprisingly, some of them such as Genetic Algorithm (GA) [1], Ant Colony Optimization (ACO) [2], and Particle Swarm Optimization (PSO) [3] are fairly well-known among not only computer scientists but also scientists from different fields. In addition to the huge number of theoretical works, such optimization techniques have been applied in various fields of study. There are four main reasons behind this. They are: simplicity, flexibility, derivation- free mechanism, and local optima avoidance. The inspirations are typically related to physical phenomena, animals' behaviours, or evolutionary concepts. The simplicity allows the scientists to simulate different natural concepts, propose new meta-heuristics, hybridize two or more meta-heuristics, or improve the current meta-heuristics. Secondly, flexibility refers to the applicability of meta-heuristics to different problems without any special changes in the structure of the algorithm. The optimization process starts with random solution(s), and there is no need to calculate the derivative of search spaces to find the optimum. This makes meta-heuristics highly suitable for real problems with expensive or unknown derivative information. The search space of real problems is usually unknown and very complex with a massive number of local optima, so meta-heuristics are good options for optimizing these challenging real problems.

## 6.2 Inspiration [2]

Grey wolf (Canis lupus) belongs to Canidae family. Grey wolves are considered as apex predators, meaning that they are at the top of the food chain. Grey wolves mostly prefer to live in a pack. The group size is 5-12 on average. Of particular interest is that they have a very strict social dominant hierarchy.

The leaders are a male and female, called alphas. The alpha is mostly responsible for making decisions about hunting, sleeping place, time to wake, and so on. The alpha's decisions are dictated to the pack. However, some kind of democratic behaviour has also been observed, in which an alpha follows the other wolves in the pack. In gatherings, the entire pack acknowledges the alpha by holding their tails down. The alpha wolf is also called the dominant wolf since his/her orders should be followed by the pack [5]. The alpha wolves are only allowed to mate in the pack. Interestingly, the alpha is not necessarily the strongest member of the pack but the best in terms of managing the pack. This shows that the organization and discipline of a pack is much more important than its strength.

The second level in the hierarchy of grey wolves is beta. The betas are subordinate wolves that help the alpha in decision-making or other pack activities. The beta wolf can be either male or female, and he/she is probably the best candidate to be the alpha in case one of the alpha wolves passes away or becomes very old. The beta wolf should respect the alpha, but commands the other lower-level wolves as well. It plays the role of an adviser to the alpha and discipliner for the pack. The beta reinforces the alpha's commands throughout the pack and gives feedback to the alpha.

The lowest ranking grey wolf is omega. The omega plays the role of scapegoat. Omega wolves always have to submit to all the other dominant wolves. They are the last wolves that are allowed to eat. It may seem the omega is not an important individual in the pack, but it has been observed that the whole pack face internal fighting and problems in case of losing the omega. This is due to the venting of violence and frustration of all wolves by the omega(s). This assists satisfying the entire pack and maintaining the dominance structure. In some cases the omega is also the babysitters in the pack.

If a wolf is not an alpha, beta, or omega, he/she is called subordinate (or delta in some references). Delta wolves have to submit to alphas and betas, but they dominate the omega. Scouts, sentinels, elders, hunters, and caretakers belong to this category. Scouts are responsible for watching the boundaries of the territory and warning the pack in case of any danger. Sentinels protect and guarantee the safety of the pack. Elders are the experienced wolves who used to be alpha or beta. Hunters help the alphas and betas when hunting prey and providing food for the pack. Finally, the caretakers are responsible for caring for the weak, ill, and wounded wolves in the pack.

In addition to the social hierarchy of wolves, group hunting is another interesting social behaviour of grey wolves. According to Muro et al [6] the main phases of gray wolf hunting are as follows:

- Tracking, chasing, and approaching the prey
- Pursuing, encircling, and harassing the prey until it stops moving
- Attack towards the prey

In this work this hunting technique and the social hierarchy of grey wolves are mathematically modelled in order to design GWO and perform optimization.

## 6.3 Mathematical Model [7]

The hunting technique and the social hierarchy of grey wolves are mathematically modeled in order to design GWO and perform optimization. The proposed mathematical models of the social hierarchy, tracking, encircling, and attacking prey are as follows:

### 6.3.1 Social hierarchy

In order to mathematically model the social hierarchy of wolves when designing GWO, we consider the fittest solution as the alpha (α). Consequently, the second and third best solutions are named beta (β) and delta (δ) respectively. The rest of the candidate solutions are assumed to be omega (ω). In the GWO algorithm the hunting (optimization) is guided by α, β, and δ. The ω wolves follow these three wolves as shown in Fig 6.1.

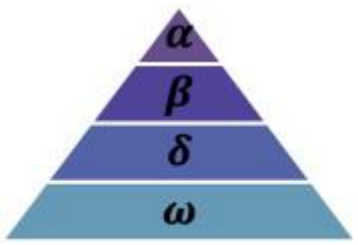

**Fig.6.1**: Hierarchy of grey wolf (dominance decreases from top down)

### 6.3.2 Encircling prey

As mentioned above, grey wolves encircle prey during the hunt. In order to mathematically model encircling behaviour the following equations are proposed:

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

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

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:

$\vec{A} = 2\vec{a}\,\vec{r_1} - \vec{a}$

$\vec{C} = 2.\vec{r_2}$

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

With the above equations, a grey wolf in the position of (X,Y) can update its position according to the position of the prey (X*,Y*). Different places around the best agent can be reached with respect to the current position by adjusting the value of $\vec{A}$ and $\vec{C}$ vectors.

### 6.3.3 Hunting

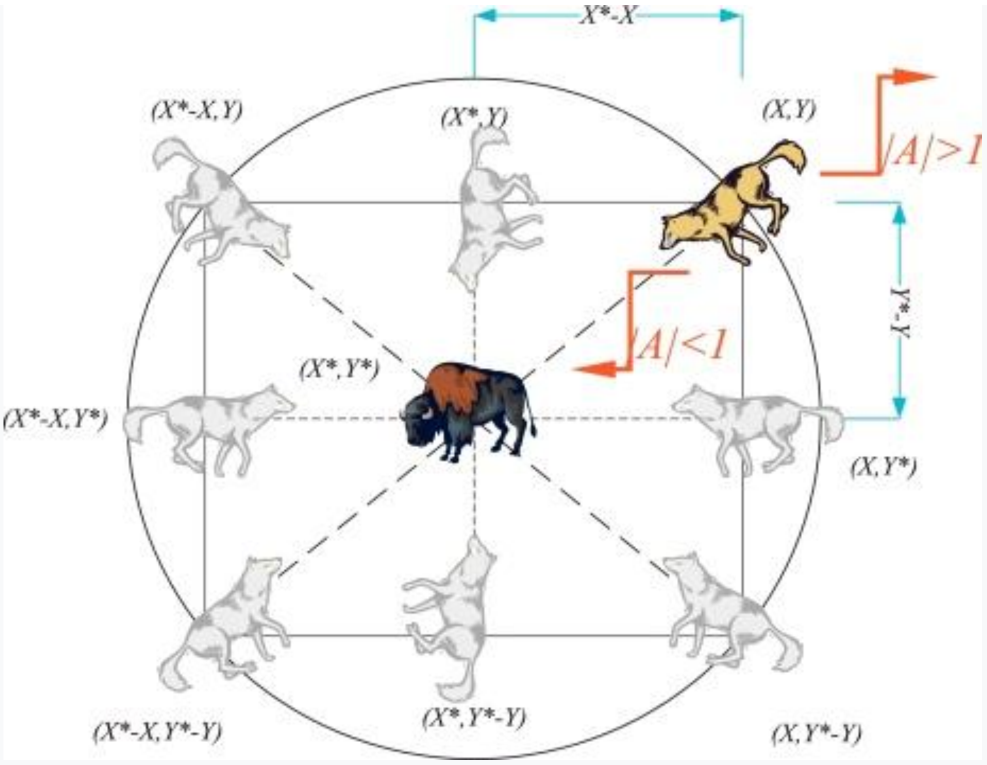

**Fig 6.2:** Updating position of gray wolves in GWO

Grey wolves have the ability to recognize the location of prey and encircle them. The hunt is usually guided by the alpha. The beta and delta might also participate in hunting occasionally. However, in an abstract search space we have no idea about the location of the optimum (prey). In order to mathematically simulate the hunting behaviour of grey wolves, we suppose that the alpha (best candidate solution) beta and delta have better knowledge about the potential location of prey. Therefore, we save the first three best solutions obtained so far and oblige the other search agents (including the omegas) to update their positions according to the position of the best search agent. The following formulas are proposed in this regard.

$\vec{D}_{\alpha} = |\vec{C}_1 . \vec{X}_{\alpha}(t) - \vec{X}|$

$\vec{D}_{\beta} = |\vec{C}_2 . \vec{X}_{\beta}(t) - \vec{X}|$

$\vec{D}_{\gamma} = |\vec{C}_3 . \vec{X}_{\gamma}(t) - \vec{X}|$

$\vec{X}_1 = \vec{X}_{\alpha} - \vec{A}_1 .(\vec{D}_{\alpha})$

$\vec{X}_2 = \vec{X}_{\beta} - \vec{A}_2 .(\vec{D}_{\beta})$

$\vec{X}_3 = \vec{X}_{\gamma} - \vec{A}_3 .(\vec{D}_{\gamma})$

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

With these equations, a search agent updates its position according to alpha, beta, and delta in a n dimensional search space. In addition, the final position would be in a random place within a circle which is defined by the positions of alpha, beta, and delta in the search space. In other words alpha, beta, and delta estimate the position of the prey, and other wolves updates their positions randomly around the prey.

### 6.3.4 Attacking prey (exploitation)

As mentioned above the grey wolves finish the hunt by attacking the prey when it stops moving. In order to mathematically model approaching the prey we decrease the value of $\vec{a}$. Note that the fluctuation range of $\vec{A}$ is also decreased by $\overline{a}$. In other words $\vec{A}$ is a random value in the interval [-2a, 2a] where $a$ is decreased from 2 to 0 over the course of iterations. When random values of $\vec{A}$ are in [-1, 1], the next position of a search agent can be in any position between its current position and the position of the prey.

With the operators proposed so far, the GWO algorithm allows its search agents to update their position based on the location of the alpha, beta, and delta; and attack towards the prey. However, the GWO algorithm is prone to stagnation in local solutions with these operators. It is true that the encircling mechanism proposed shows exploration to some extent, but GWO needs more operators to emphasize exploration.

### 6.3.5 Search for prey (exploration)

Grey wolves mostly search according to the position of the alpha, beta, and delta. They diverge from each other to search for prey and converge to attack prey. In order to mathematically model divergence, we utilize $\vec{A}$ with random values greater than 1 or less than -1 to oblige the search agent to diverge from the prey. This emphasizes exploration and allows the GWO algorithm to search globally. $|A|>1$ forces the grey wolves to diverge from the prey to hopefully find a fitter prey. Another component of GWO that favors exploration is $\vec{C}$, which contains random values in [0, 2]. This component provides random weights for prey in order to stochastically emphasize ($C>1$) or de-emphasize ($C<1$) the effect of prey in defining the distance in Equation (3.1). This assists GWO to show a more random behaviour throughout optimization, favoring exploration and local optima avoidance. It is worth mentioning here that C is not linearly decreased in contrast to A. We deliberately require C to provide random values at all times in order to emphasize exploration not only during initial iterations but also final iterations. This component is very helpful in case of local optima stagnation, especially in the final iterations.

The C vector can be also considered as the effect of obstacles to approaching prey in nature. Generally speaking, the obstacles in nature appear in the hunting paths of wolves and in fact prevent them from quickly and conveniently approaching prey. This is exactly what the vector C does. Depending on the position of a wolf, it can randomly give the prey a weight and make it harder and farther to reach for wolves, or vice versa.

To sum up, the search process starts with creating a random population of grey wolves (candidate solutions) in the GWO algorithm. Over the course of iterations, alpha, beta, and delta wolves estimate the probable position of the prey. Each candidate solution updates its distance from the prey. The parameter $a$ is decreased from 2 to 0 in order to emphasize exploration and exploitation, respectively. Candidate solutions tend to diverge from the prey when $|\vec{A}| > 1$ and converge towards the prey when $|\vec{A}| < 1$. Finally, the GWO algorithm is terminated by the satisfaction of an end criterion.

## 6.4 Objective function of the Grey Wolf Algorithm

Here, minimization of integral-time-absolute-error (ITAE) is defined as the objective function (performance index or fitness function). The ITAE is calculated as:

ITAE = $\int_0^t t * |e(t)| dt$

## 6.5 Grey Wolf Algorithm Parameters

| | |
|---|---|
| Search agent No. | 30 |
| No. Of iterations | 100 |
| Dimension (No. Of Variables) | 2 |
| Range of Variables | 0-200% of initial parameters |

**Table 6.1:** Grey Wolf Algorithm Parameters

## 6.6 Steps of GWO Algorithm [5]

The GWO Algorithm:

```
Initialize the grey wolf population Xi (i = 1, 2, ..., n)
Initialize $\vec{a}$ , $\vec{A}$ , and $\vec{C}$
Calculate the fitness of each search agent
$\vec{X}_\alpha$ =the best search agent
$\vec{X}_\beta$ =the second best search agent
$\vec{X}_\gamma$ =the third best search agent
while (t < Max number of iterations)
for each search agent
            Update the position of the current search agent by above equations
end for
Update $\overrightarrow{a}$ , $\overrightarrow{A}$ and $\overrightarrow{C}$
Calculate the fitness of all search agents
Update $\vec{X}_\alpha$ , $\vec{X}_\beta$ , and $\vec{X}_\gamma$
t=t+1
end while
return $\vec{X}_\alpha$
```

To see how GWO is theoretically able to solve optimization problems, some points may be noted:

- The proposed social hierarchy assists GWO to save the best solutions obtained so far over the course of iteration
- The proposed encircling mechanism defines a circle-shaped neighbourhood around the solutions which can be extended to higher dimensions as a hyper-sphere
- The random parameters *A* and *C* assist candidate solutions to have hyper-spheres with different random radii
- The proposed hunting method allows candidate solutions to locate the probable position of the prey

- Exploration and exploitation are guaranteed by the adaptive values of $a$ and $A$
- The adaptive values of parameters $a$ and $A$ allow GWO to smoothly transition between exploration and exploitation
- With decreasing A, half of the iterations are devoted to exploration ($|A|\geq 1$) and the other half are dedicated to exploitation ($|A|<1$)
- The GWO has only two main parameters to be adjusted ($a$ and $C$)

## 6.7 Results:

For simulation and performance study the process transfer function of the identified model and the process itself of both QUBE-Servo2 and SRV02 are being considered.

For QUBE Servo 2, the voltage-to-position transfer function is

$$P(s) = \frac{\theta_m(s)}{V_m(s)} = \frac{22.7}{s(0.158s+1)}$$

For Rotary Servo Base unit SRV02, the voltage-to-position transfer function is

$$P(s) = \frac{\theta_m(s)}{V_m(s)} = \frac{1.53}{s(0.0253s+1)}$$

We have studied the close loop response characteristics for the identified process model by using different controllers. For detailed comparison, in addition to the response characteristics, several performance indices, such as percentage overshoot (%OS),rise time($T_r$), settling time($T_s$), integral absolute error (IAE), integral time absolute error (ITAE), integral square error (ISE) are calculated for each controller. Performance of GWO-PD is compared with the corresponding PD controller. Simpson's1/3rd rule is used for numerical integration. The detailed performance analysis for various types of process is discussed next.

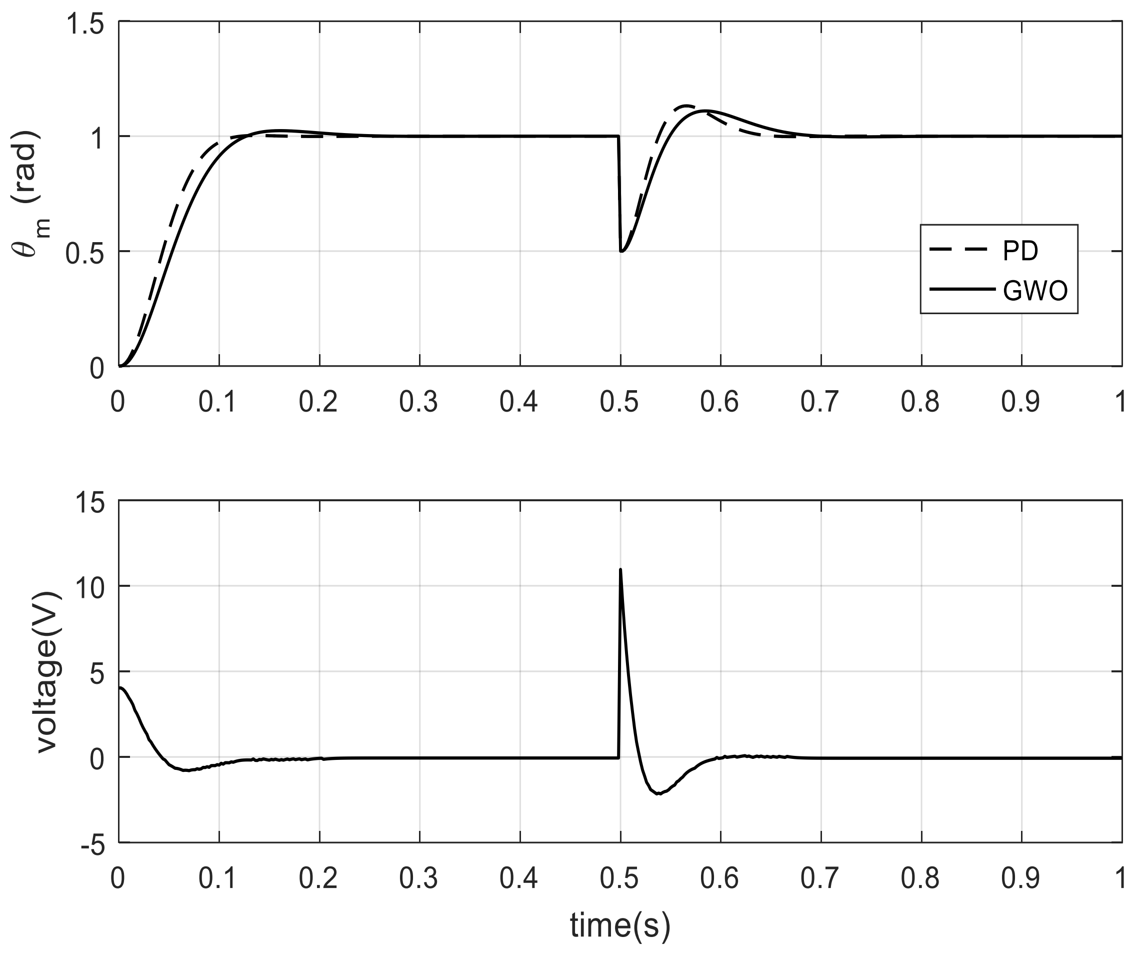

**Fig6.3:** Responses of Motor Speed of PD and GWO based controllers of the QUBE-Servo2 identified model and its corresponding Motor voltage

| Objective Function | QUBE-Servo2 Model Characteristics | GWO | PD |
|---|---|---|---|
| ITAE | Rise Time (ms) | 77.313 | 64.010 |
| | %Overshoot | 1.950 | 0.117 |
| | Peak Time(sec) | 0.160 | 0.136 |
| | ITAE | 0.0143 | 0.01135 |
| | IAE | 0.08085 | 0.06633 |
| | ISE | 0.04667 | 0.03891 |

**Table 6.2:** Performance Table of Controllers of QUBE-Servo2 Model

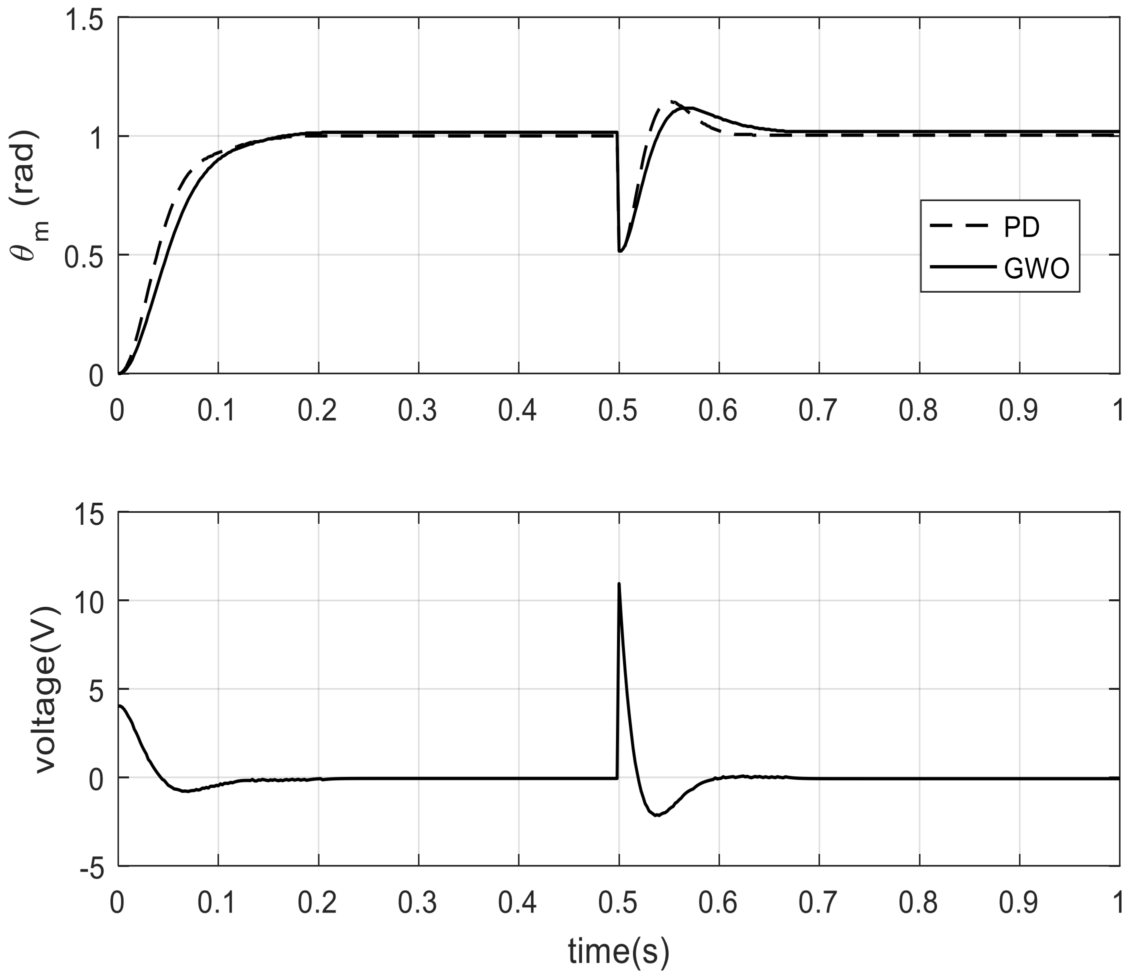

**Fig6.4:** Responses of Motor Speed of PD and GWO based controllers of the QUBE-Servo2 and its corresponding Motor Voltage

| Objective Function | QUBE Servo 2 Characteristics | GWO | PD |
|---|---|---|---|
| ITAE | Rise Time (ms) | 88.115 | 70.762 |
| | %Overshoot | -0.629 | -0.122 |
| | Peak Time(sec) | 0.204 | 0.178 |
| | ITAE | 0.01924 | 0.01074 |
| | IAE | 0.08581 | 0.06353 |
| | ISE | 0.04255 | 0.03474 |

**Table 6.3:** Performance Table of Controllers of QUBE-Servo2

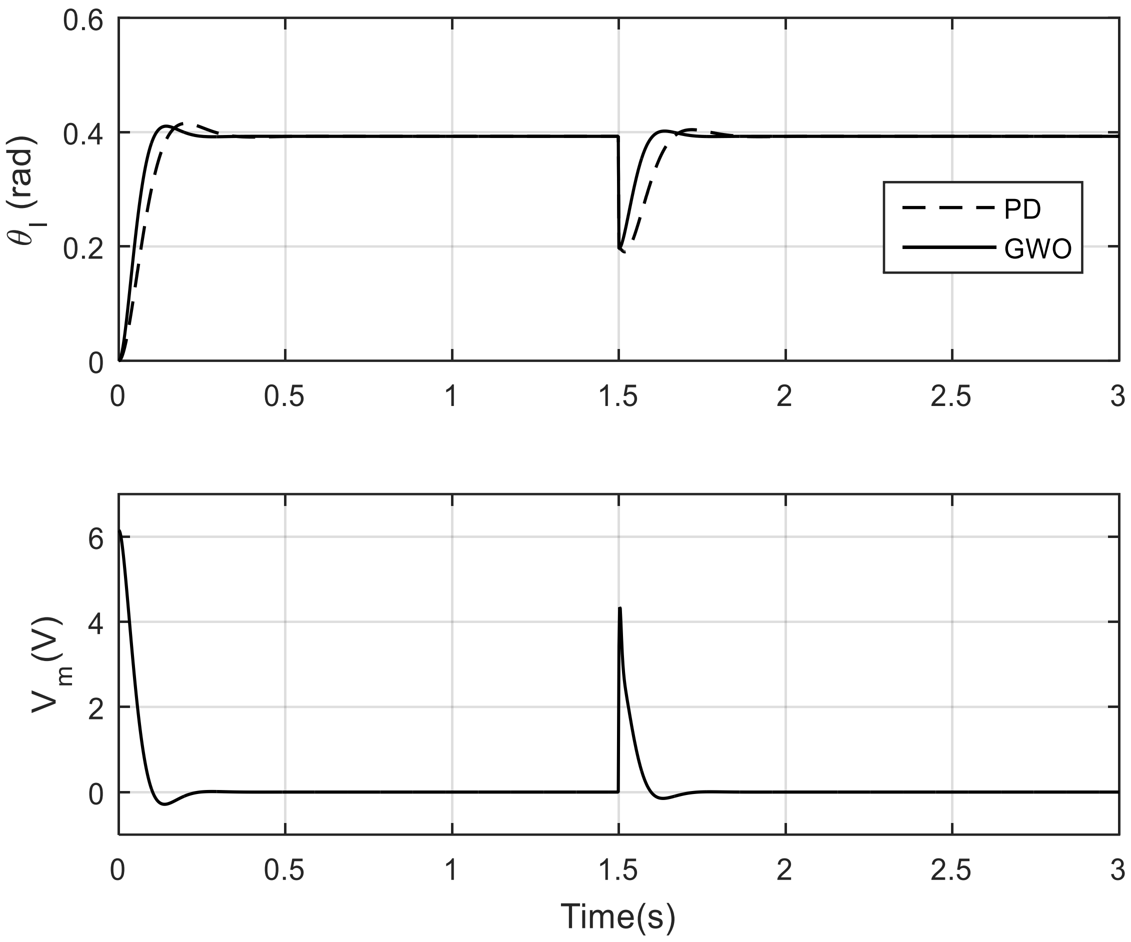

**Fig 6.5:**Responses of Motor Speed of PD and GWO based controllers of the Rotary servo base SRV02 identified model and its corresponding Motor voltage

| **Objective Function** | **Rotary servo base SRV02 model Characteristics** | **GWO** | **PD** |
|---|---|---|---|
| ITAE | Rise Time (ms) | 68.126 | 78.132 |
| | %Overshoot | 4.737 | 8.717 |
| | Peak Time(sec) | 0.142 | 0.202 |
| | ITAE | 0.01551 | 0.03135 |
| | IAE | 0.02996 | 0.04871 |
| | ISE | 0.0006471 | 0.01033 |

**Table 6.4:** Performance Table of Controllers of Rotary servo base SRV02 model

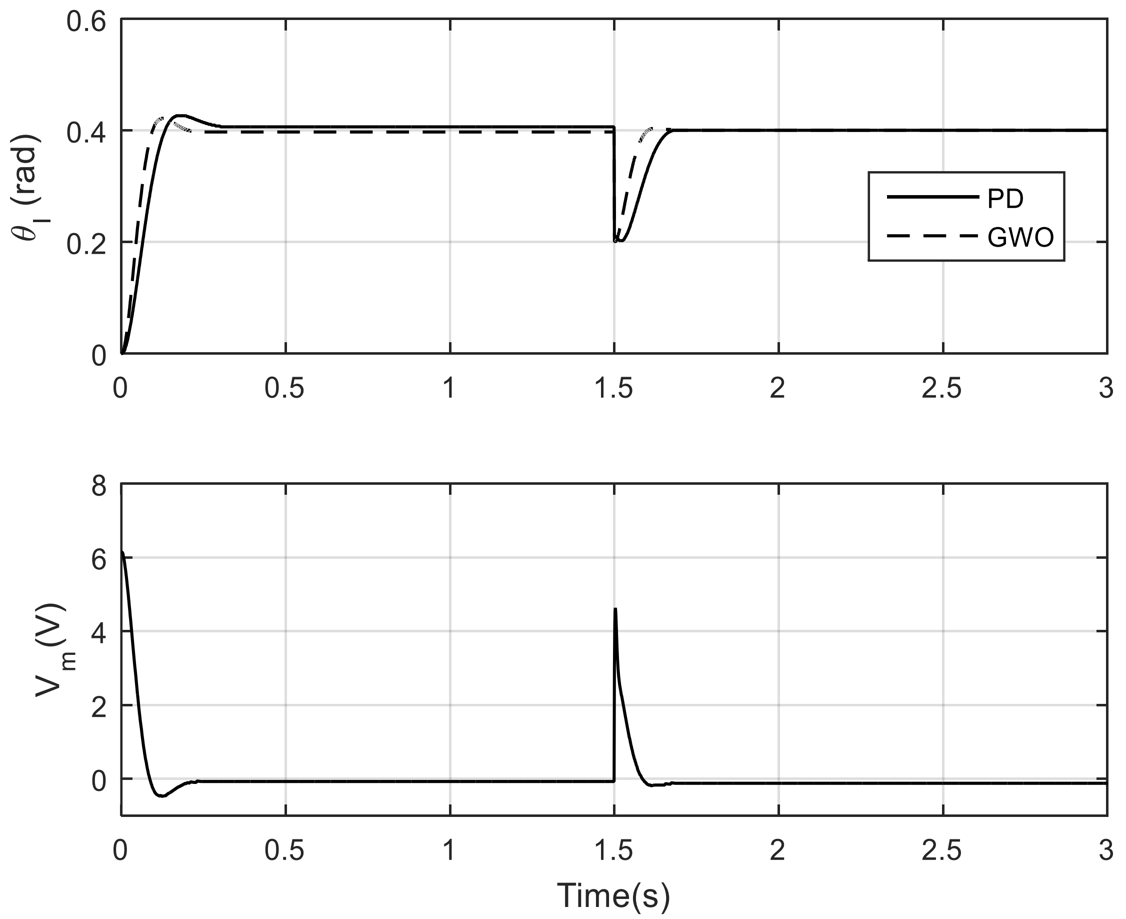

**Fig 6.6:** Response of Motor Speed of PD and GWO based controllers of the Rotary servo base SRV02 and its corresponding motor voltage

| **Objective Function** | **Rotary servo base SRV02 Characteristics** | **GWO** | **PD** |
|---|---|---|---|
| ITAE | Rise Time (ms) | 49.506 | 72.083 |
|  | %Overshoot | 7.060 | 7.484 |
|  | Peak Time(sec) | 0.122 | 0.166 |
|  | ITAE | 0.04351 | 0.06585 |
|  | IAE | 0.004536 | 0.07305 |
|  | ISE | 0.006419 | 0.01021 |

**Table 6.5:** Performance Table of Controllers of Rotary servo base SRV02

## 6.8 Conclusion

Here, we have explored the possibility of performance enhancement of position control by Grey Wolf Optimization technique. From the simulation results we observed that the actual process model of QUBE Servo and rotary servo base unit produces good result due to set point changes as well as load disturbances. In tables, results for the best optimized variables $K_p$ and $K_d$ are shown. A typical Convergence plot of the objective function vs. Iteration is shown below:

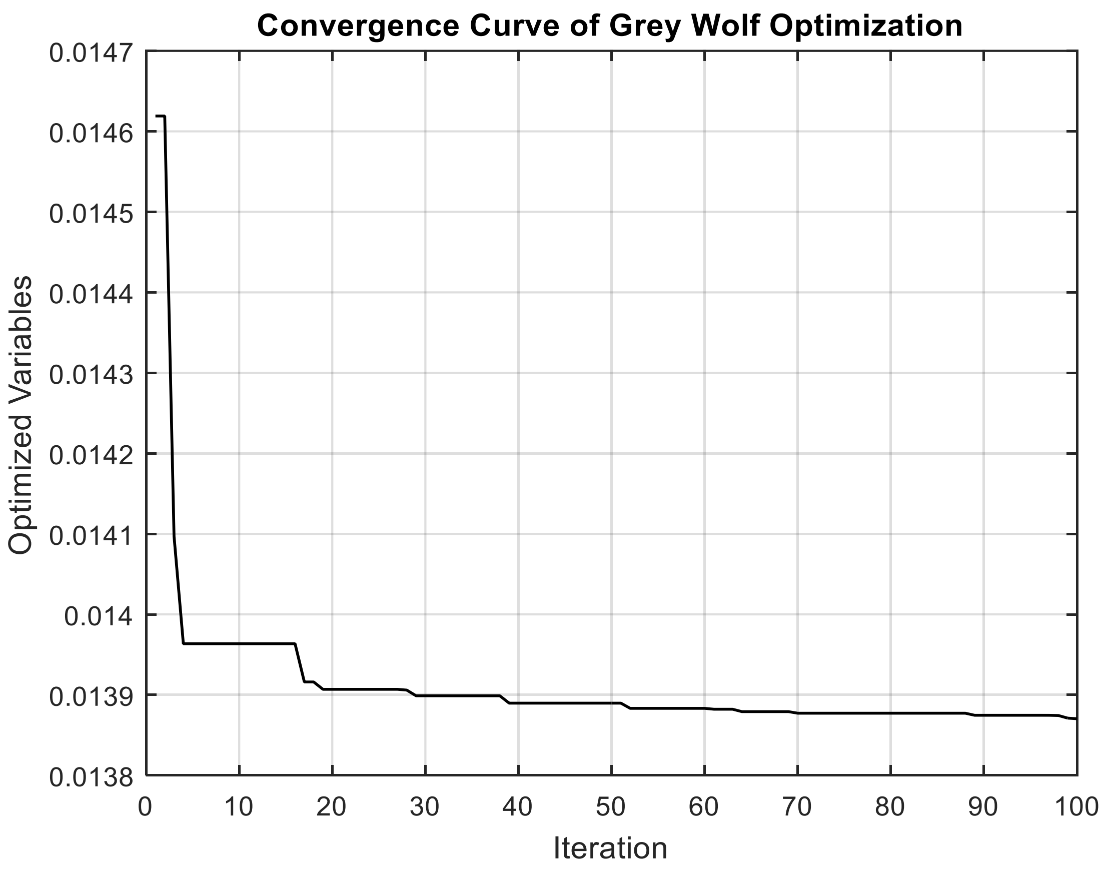

**Fig6.7:** Convergence curve for QUBE-Servo2

## 6.9 References

[1] S.Mirjalili, S. M. Mirjalili, and A. Lewis, "Grey Wolf Optimizer," Advances in Engineering Software, vol. 69, pp. 46-61, 2014.

[2] Bonabeau E, Dorigo M, Theraulaz G. Swarm intelligence: from natural to artificial systems: OUP USA; 1999.

[3] Dorigo M, Birattari M, Stutzle T. Ant colony optimization. Computing Intelligence Magazine, IEEE 2006;1:28–39.

[4] Kennedy J, Eberhart R. Particle swarm optimization, in Neural Networks, 1995.In: Proceedings, IEEE international conference on; 1995. p. 1942–1948.

[5] Mech LD. Alpha status, dominance, and division of labour in wolf packs. Can J Zool 1999;77:1196–203.

[6] Muro C, Escobedo R, Spector L, Coppinger R. Wolf-pack (Canis lupus) hunting strategies emerge from simple rules in computational simulations. Behav Process 2011;88:192–7.

[7] Grey Wolf Optimizer ,Seyedali Mirjalili ,Seyed Mohammad Mirjalili ,Andrew Lewis School of Information and Communication Technology, Griffith University, Nathan Campus, Brisbane QLD 4111, Australia Department of Electrical Engineering, Faculty of Electrical and Computer Engineering, Shahid Beheshti University, G.C. 1983963113, Tehran, Iran

[8] Lee KS, Geem ZW. A new meta-heuristic algorithm for continuous engineering optimization: harmony search theory and practice. Comput Methods ApplMech Eng 2005;194:3902–33.

# Chapter 7

Optimal PD Controller using Ant Lion Optimization (ALO)

## 7.1 Introduction

In recent years metaheuristic algorithms have been used as primary techniques for obtaining the optimal solutions of real engineering design optimization problems [1–3]. Such algorithms mostly benefit from stochastic operators [4] that make them distinct from deterministic approaches. A deterministic algorithm [5–7] reliably determines the same answer for a given problem with a similar initial starting point. However, this behaviour results in local optima entrapment, which can be considered as a disadvantage for deterministic optimization techniques [8]. Stochastic optimization (metaheuristic) algorithms [9] refer to the family of algorithms with stochastic operators including evolutionary algorithms [10]. Randomness is the main characteristic of stochastic algorithms [11]. This means that they utilize random operators when seeking for global optima in search spaces. Evolutionary algorithms search for the global optimum in a search space by creating one or more random solutions for a given problem [13]. This set is called the set of candidate solutions. The set of candidates is then improved iteratively until the satisfaction of a terminating condition. The improvement can be considered as finding a more accurate approximation of the global optimum than the initial random guesses. This mechanism brings evolutionary algorithms several intrinsic advantages: problem independency, derivation independency, local optima avoidance, and simplicity. Problem and derivation independencies originate from the consideration of problems as a black box. Evolutionary algorithms only utilize the problem formulation for evaluating the set of candidate solutions. The main process of optimization is done completely independent from the problem and based on the provided inputs and received outputs. Therefore, the nature of the problem is not a concern, yet the representation is the key step when utilizing evolutionary algorithms. This is the same reason why evolutionary algorithms do not need to derivate the problem for obtaining its global optimum.

## 7.2 Inspiration [15]

Antlions (doodlebugs) belong to the Myrmeleontidae family and Neuroptera order (net-winged insects). The lifecycle of antlions includes two main phases: larvae and adult. A natural total lifespan can take up to 3 years, which mostly occurs in larvae (only 3–5 weeks for adulthood). Antlions undergo metamorphosis in a cocoon to become adult. They mostly hunt in larvae and the adulthood period is for reproduction.

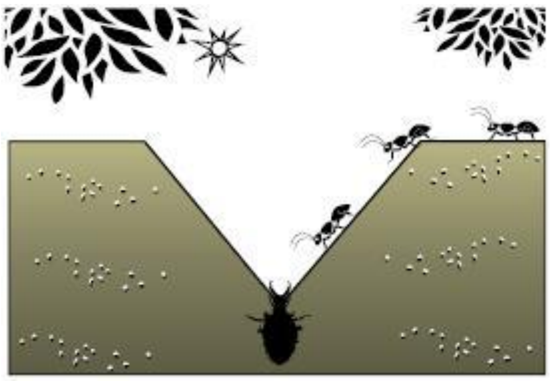

**Fig 7.1:** Antlion trap

Their names originate from their unique hunting behaviour and their favourite prey. An antlion larvae digs a cone-shaped pit in sand by moving along a circular path and throwing out sands with its massive jaw. Fig. 7.1 shows several cone-shaped pits with different sizes. After digging the trap, the larvae hides underneath the bottom of the cone and waits for insects (preferably ant) to be trapped in the pit as illustrated in Fig 7.1. The edge of the cone is sharp enough for insects to fall to the bottom of the trap easily. Once the antlion realizes that a prey is in the trap, it tries to catch it. However, insects usually are not caught immediately and try to escape from the trap. In this case, antlions intelligently throw sands towards to edge of the pit to slide the prey into the bottom of the pit. When a prey is caught into the jaw, it is pulled under the soil and consumed. After consuming the prey, antlions throw the leftovers outside the pit and amend the pit for the next hunt.

The main inspiration of the ALO algorithm comes from the foraging behaviour of antlion’s larvae. In the next subsection the behaviour of antlions and their prey in nature is first modelled mathematically. An optimization algorithm is then proposed based on the mathematical model.

## 7.3 Operations used in antlion algorithm: [15]

### 7.3.1. Random walks of ants

The ALO algorithm mimics interaction between antlions and ants in the trap. To model such interactions, ants are required to move over the search space, and antlions are allowed to hunt them and become fitter using traps. Since ants move stochastically in nature when searching for food, a random walk is chosen for modelling ants' movement. Ants update their positions with random walk at every step of optimization. Since every search space has a boundary (range of variable), however, the equation cannot be directly used for updating position of ants. In order to keep the random walks inside the search space, they are normalized using the following equation (min–max normalization):

$$X_i^t = (X_i^t - a_i) - (d_i - c_i^t) / (d_i^t - a_i) \quad (7.1)$$

where $a_i$ is the minimum of random walk of i-th variable, $d_i$ is the maximum of random walk in i-th variable, $c_i^t$ is the minimum of i-th variable at t-th iteration, and $d_i^t$ indicates the maximum of i-th variable at t-th iteration

### 7.3.2. Trapping in antlion's pits

As discussed above, random walks of ants are affected by antlions' traps. In order to mathematically model this assumption, the following equations are proposed:

$$c_i^t = Antlion_t^j + c^t \quad (7.2)$$

$$d_i^t = Antlion_t^j + d^t \quad (7.3)$$

where $c^t$ is the minimum of all variables at t-th iteration, $d^t$ indicates the vector including the maximum of all variables at t-th iteration, $c_i^t$ is the minimum of all variables for i-th ant, $d_i^t$ is the maximum of all variables for i-th ant, and $Antlion_t^j$ shows the position of the selected j-th antlion at t-th iteration.

### 7.3.3. Building trap

In order to model the antlions's hunting capability, a roulette wheel is employed. The ants are assumed to be trapped in only one selected antlion. The ALO algorithm is required to utilize a roulette wheel operator for selecting antlions based of their fitness during optimization. This mechanism gives high chances to the fitter antlions for catching ants.

### 7.3.4. Sliding ants towards antlion

With the mechanisms proposed so far, antlions are able to build traps proportional to their fitness and ants are required to move randomly. However, antlions shoot sands outwards the centre of the pit once they realize that an ant is in the trap. This behaviour slides down the trapped ant that is trying to escape. For mathematically modelling this behaviour, the radius of ants's random walks hyper-sphere is decreased adaptively. The following equations are proposed in this regard:

$c^t = c^t / I$ (7.4)

$d^t = d^t / I$ (7.5)

where I is a ratio, $c^t$ is the minimum of all variables at t-th iteration, and $d^t$ indicates the vector including the maximum of all variables at t-th iteration. In Eq. (7.4) and (7.5), $I = 10^w\ t/T$ where t is the current iteration, T is the maximum number of iterations, and w is a constant defined based on the current iteration (w = 2 when t > 0.1T, w = 3 when t > 0.5T, w = 4 when t > 0.75T, w = 5 when t > 0.9T, and w = 6 when t > 0.95T). Basically, the constant w can adjust the accuracy level of exploitation.

### 7.3.5. Catching prey and re-building the pit

The final stage of hunt is when an ant reaches the bottom of the pit and is caught in the antlion's jaw. After this stage, the antlion pulls the ant inside the sand and consumes its body. For mimicking this process, it is assumed that catching prey occur when ants becomes fitter (goes inside sand) than its corresponding antlion. An antlion is then required to update its position to the latest position of the hunted ant to enhance its chance of catching new prey. The following equation is proposed in this regard:

$Antlion_j^t = Ant_j^t$ ; if $f(Ant_j^t) > f(Antlion_j^t)$ (7.6)

where t shows the current iteration, $Antlion_j^t$ shows the position of selected j-th antlion at t-th iteration, and $Ant_j^t$ indicates the position of i-th ant at t-th iteration.

### 7.3.6. Elitism

Elitism is an important characteristic of evolutionary algorithms that allows them to maintain the best solution(s) obtained at any stage of optimization process. In this study the best antlion obtained so far in each iteration is saved and considered as elite. Since the elite is the fittest antlion, it should be able to affect the movements of all the ants during iterations. Therefore, it is assumed that every ant randomly walks around a selected antlion by the roulette wheel and the elite simultaneously as follows:

$Ant_j^t = \frac{R_A^t + R_E^t}{2}$ (7.7) where $R_A^t$ is the random walk around the antlion selected by the roulette wheel at t-th iteration, $R_E^t$ is the random walk around the elite at t-th iteration, and $Ant_j^t$ indicates the position of i-th ant at t-th iteration.

## 7.4 Objective function of the Ant Lion Algorithm

Here, minimization of integral-time-absolute-error (ITAE) is defined as the objective function (performance index or fitness function). The ITAE is calculated as:

ITAE = $\int_0^t t * |e(t)| dt$

## 7.5 Ant Lion Algorithm Parameters

| | |
|---|---|
| Search agent No. | 30 |
| No. Of iterations | 100 |
| Dimension (No. Of Variables) | 2 |
| Range of Variables | 0-200% of initial parameters |

**Table 7.1:** Ant Lion Algorithm Parameters

## 7.6 Steps of ALO Algorithm [15]

The pseudo codes the ALO algorithm is defined as follows:

```
Initialize the first population of ants and antlions randomly
Calculate the fitness of ants and antlions
Find the best antlions and assume it as the elite (determined optimum)
while the end criterion is not satisfied
for every ant
Select an antlion using Roulette wheel
Update c and d using equations Eqs. (7.4) and (7.5)
Create a random walk and normalize it using Eqs. (7.1) and
Update the position of ant using (7.7)
end for
Calculate the fitness of all ants
Replace an antlion with its corresponding ant it if becomes
```

```
fitter (Eq. (7.6))
Update elite if an antlion becomes fitter than the elite
end while
Return elite
```

## 7.7 Results:

For simulation and performance study the process transfer function of the identified model and the process itself of both QUBE-Servo2 and SRV02 are being considered.

For QUBE Servo 2, the voltage-to-position transfer function is

$$P(s) = \frac{\theta_m(s)}{V_m(s)} = \frac{22.7}{s(0.158s+1)}$$

For Rotary Servo Base unit SRV02, the voltage-to-position transfer function is

$$P(s) = \frac{\theta_m(s)}{V_m(s)} = \frac{1.53}{s(0.0253s+1)}$$

We have derived the close loop response characteristics for the identified process model by using different controllers. For detailed comparison, in addition to the response characteristics, several performance indices, such as percentage overshoot (%OS), rise time ($T_r$), settling time ($T_s$), integral absolute error (IAE), integral time absolute error (ITAE), integral square error (ISE) are calculated for each controller. Performance of ALO-PD is being compared with the corresponding PD controller. Simpson's1/3rd rule is used for numerical integration. The detailed performance analysis is discussed next.

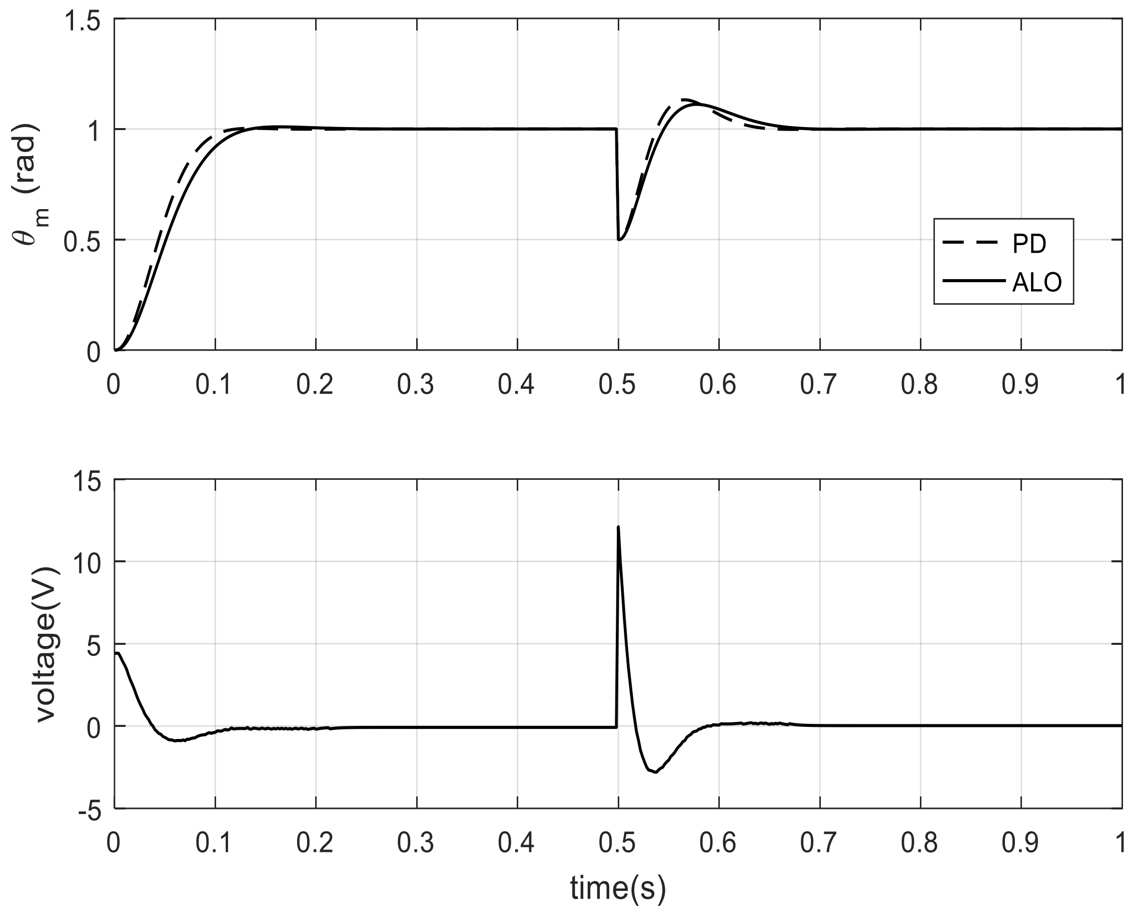

**Fig7.2:** Responses of Motor Speed of PD and ALO based controllers of the QUBE-Servo2 identified model and its corresponding Motor voltage

| **Objective Function** | **QUBE Servo 2 Model Characteristics** | **ALO** | **PD** |
|---|---|---|---|
| ITAE | Rise Time (ms) | 75.016 | 64.010 |
| | %Overshoot | 1.515 | 0.117 |
| | Peak Time(sec) | 0.164 | 0.136 |
| | ITAE | 0.0131 | 0.01135 |
| | IAE | 0.07664 | 0.06633 |
| | ISE | 0.4475 | 0.03891 |

**Table 7.2:** Performance Table of Controllers of QUBE-Servo2 Model

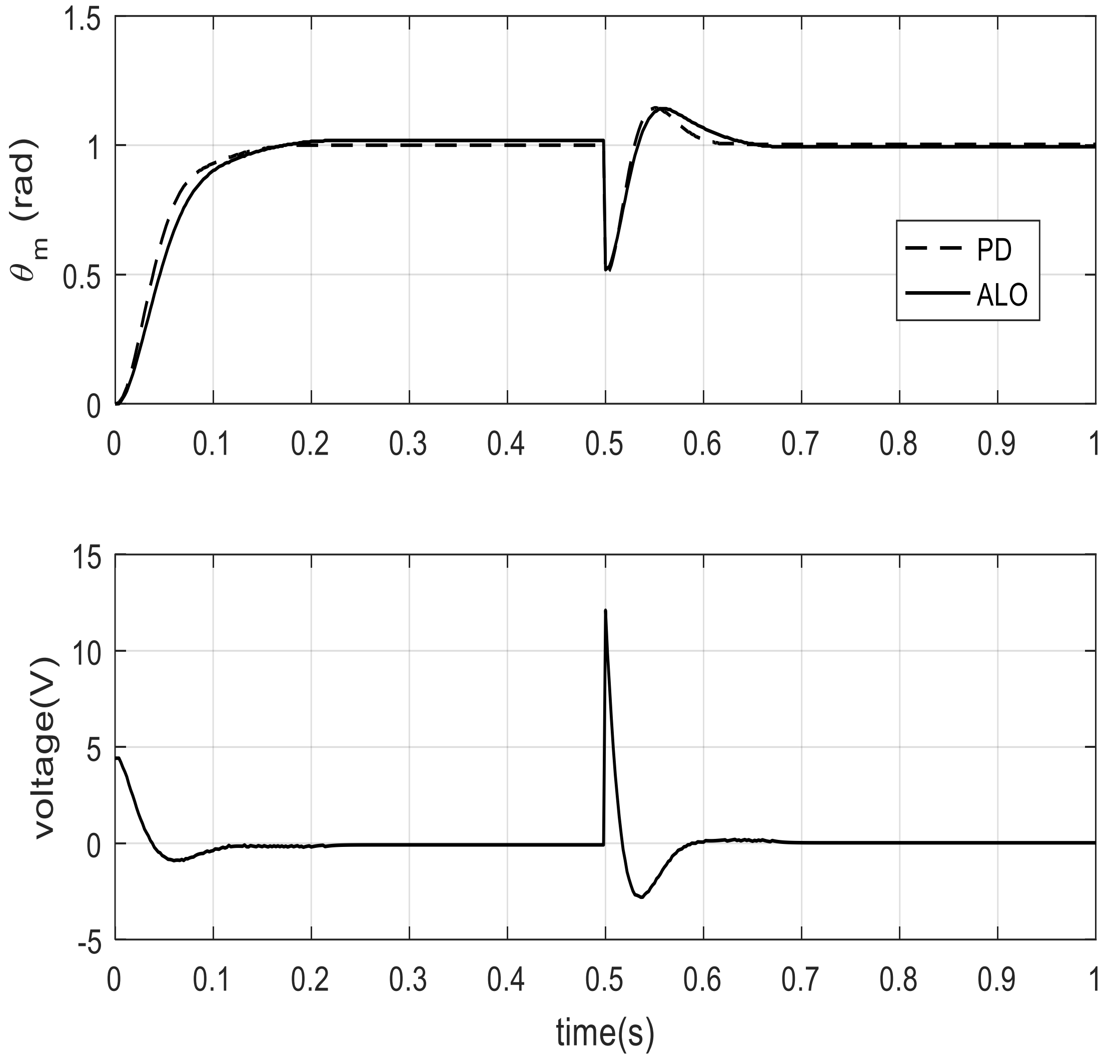

**Fig7.3:** Responses of Motor Speed of PD and ALO based controllers of the QUBE-Servo2 and its corresponding Motor voltage

| **Objective Function** | **QUBE Servo 2 Characteristics** | **ALO** | **PD** |
|---|---|---|---|
| ITAE | Rise Time (ms) | 81.948 | 70.762 |
| | %Overshoot | 0.575 | -0.122 |
| | Peak Time(sec) | 0.214 | 0.178 |
| | ITAE | 0.01524 | 0.01074 |
| | IAE | 0.07984 | 0.06353 |
| | ISE | 0.04007 | 0.03474 |

**Table 7.3:** Performance Table of Controllers of QUBE-Servo2

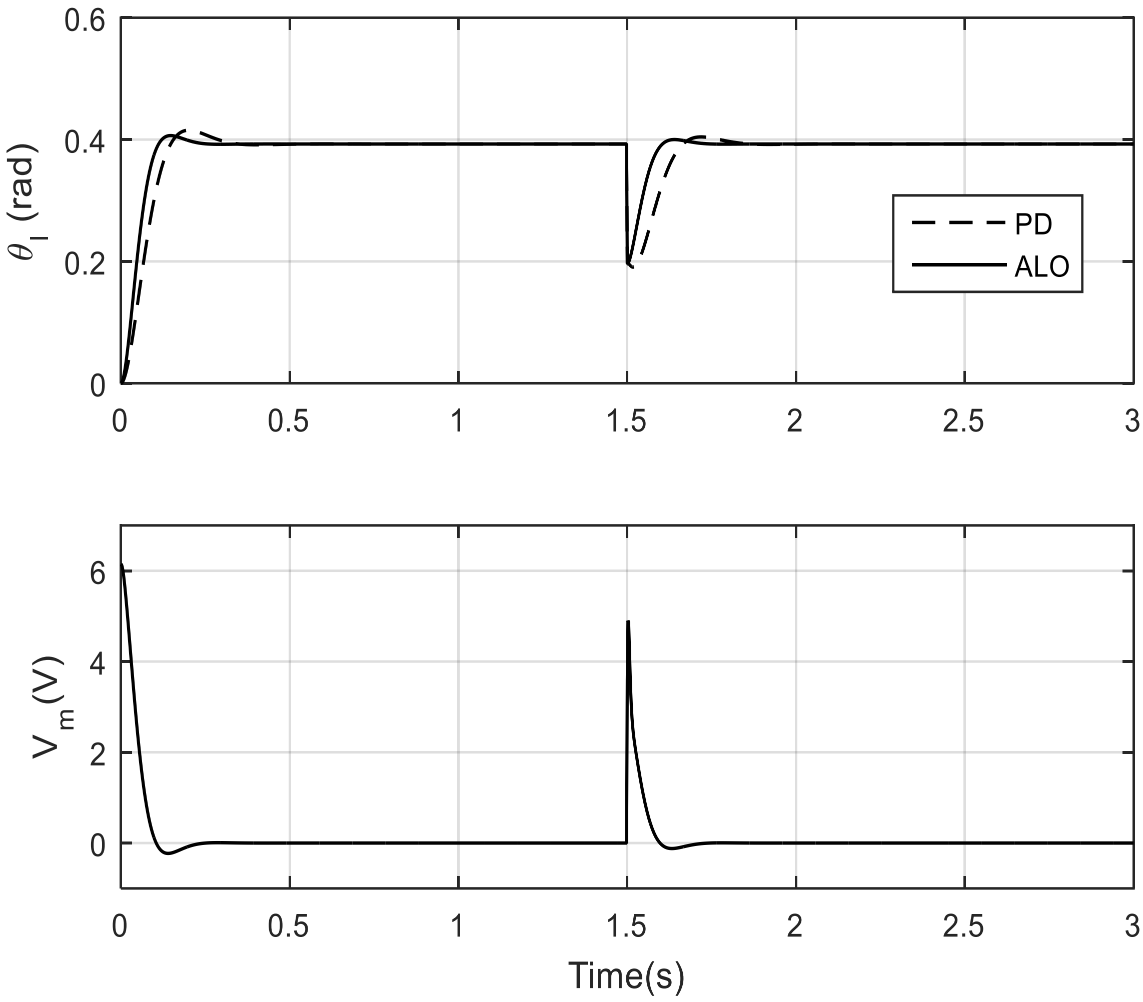

**Fig 7.4:** Responses of Motor Speed of PD and ALO based controllers of the Rotary servo base SRV02 identified model and its corresponding Motor voltage

| **Objective Function** | **Rotary servo base SRV02 model Characteristics** | **ALO** | **PD** |
|---|---|---|---|
| ITAE | Rise Time (ms) | 70.635 | 78.132 |
| | %Overshoot | 3.644 | 8.717 |
| | Peak Time(sec) | 0.148 | 0.202 |
| | ITAE | 0.01507 | 0.03135 |
| | IAE | 0.02973 | 0.04871 |
| | ISE | 0.006483 | 0.01033 |

**Table 7.4:** Performance Table of Controllers of Rotary servo base SRV02 Model

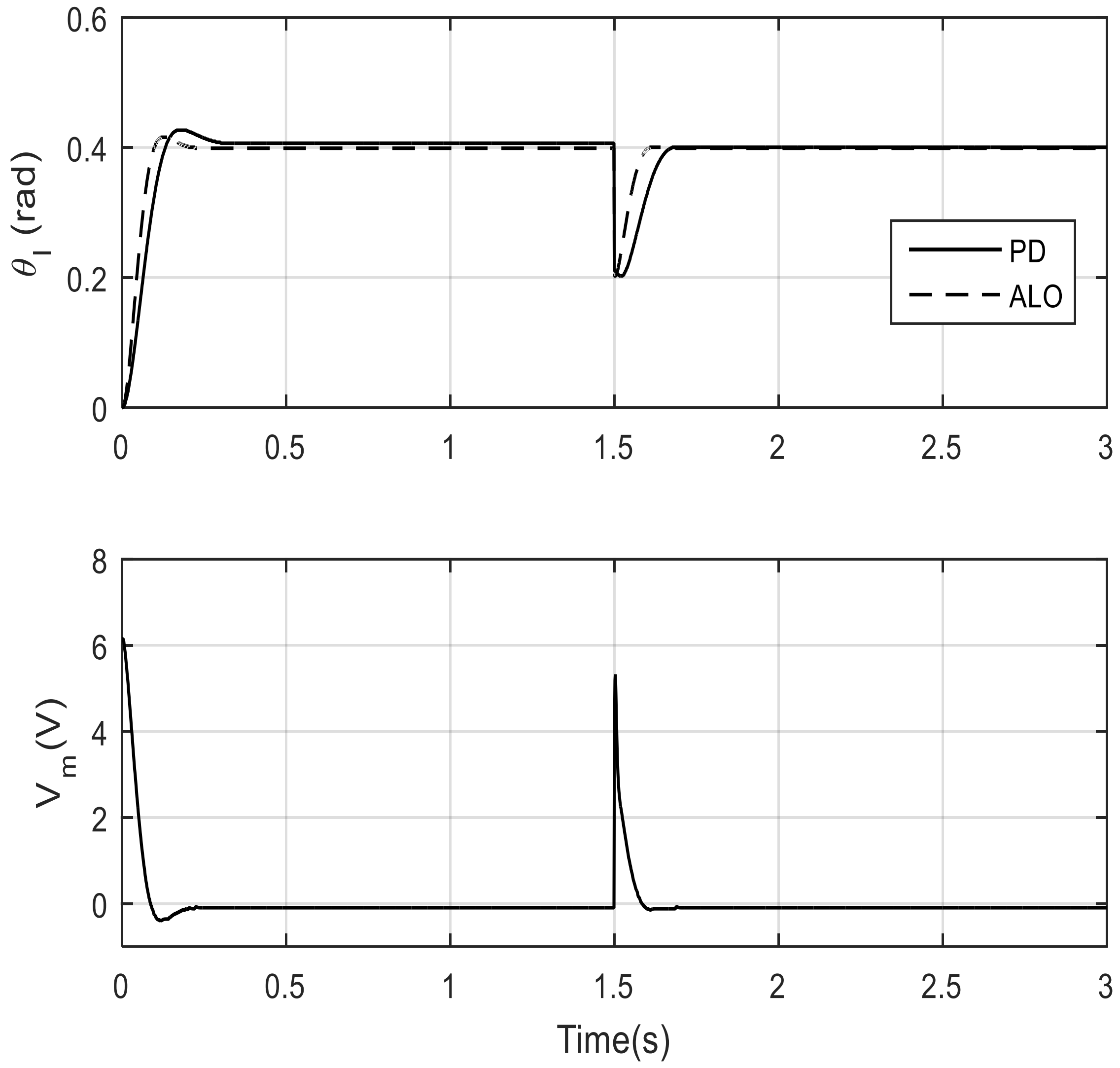

**Fig 7.5:** Responses of Motor Speed of PD and ALO based controllers of the Rotary servo base SRV02 and its corresponding Motor voltage

| Objective Function | Rotary servo base SRV02 Characteristics | ALO | PD |
|---|---|---|---|
| ITAE | Rise Time (ms) | 61.830 | 72.083 |
| | %Overshoot | 4.737 | 7.484 |
| | Peak Time(sec) | 0.119 | 0.166 |
| | ITAE | 0.03994 | 0.06585 |
| | IAE | 0.04498 | 0.07305 |
| | ISE | 0.006462 | 0.01021 |

**Table 7.5:** Performance Table of Controllers of Rotary servo base SRV02

## 7.8 Conclusion

Here, we have explored the possibility of performance enhancement of position control by Grey Wolf Optimization technique. From the simulation results we observed that the actual process model of QUBE-Servo2 and rotary servo base unit SRV02 produces good result due to set point changes as well as load disturbances. In tables, results for the best optimized variables $K_p$ and $K_d$ are shown. A typical Convergence plot of the objective function vs. Iteration is shown below:

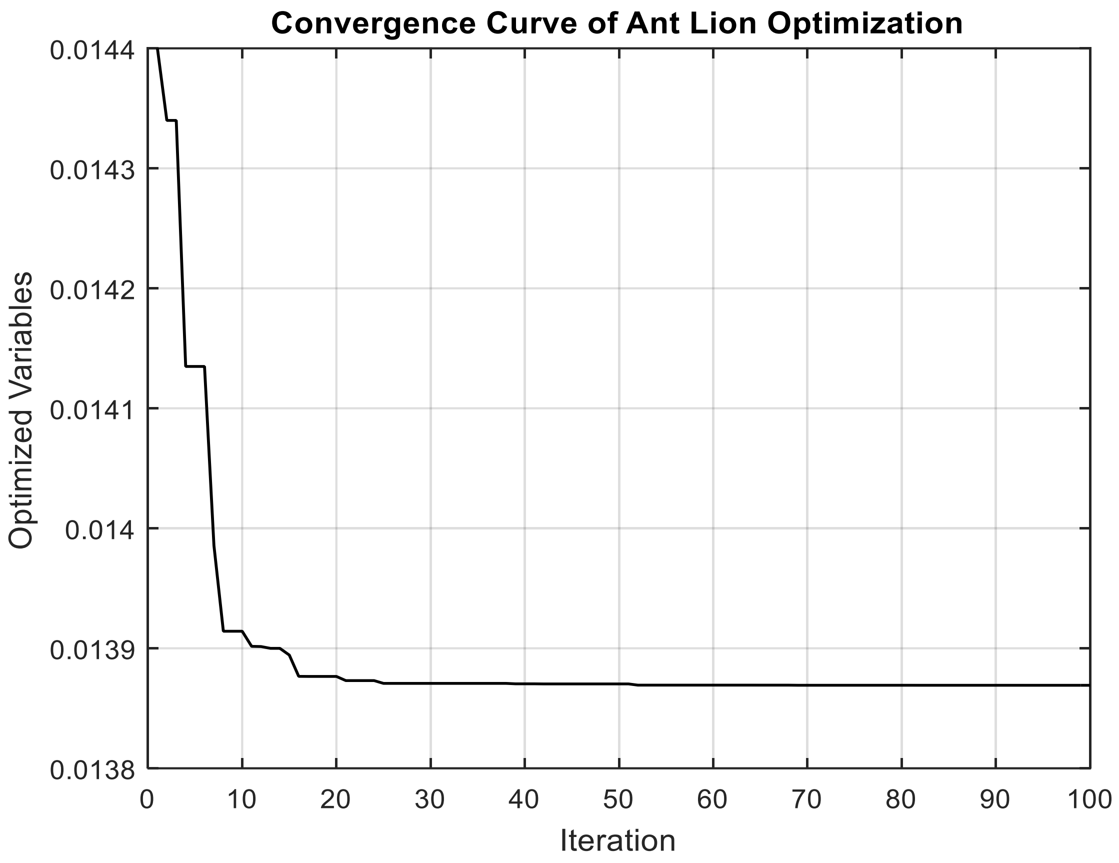

**Fig 7.6:** Convergence curve for QUBE-Servo2 motor

## References

[1] Blum C, Puchinger J, Raidl GR, Roli A. Hybrid metaheuristics in combinatorial optimization: a survey. Application for Soft Computing 2011;11:4135–51.

[2] Boussaïd I, Lepagnot J, Siarry P. A survey on optimization metaheuristics.
Inform Sci 2013;237:82–117.

[3] Gogna A, Tayal A. Metaheuristics: review and application. Experiment on Theory of Artificial Intelligence 2013;25:503–26.

[4] Bianchi L, Dorigo M, Gambardella LM, Gutjahr WJ. A survey on metaheuristics for stochastic combinatorial optimization. Nat Computing: Int J 2009;8:239–87.

[5] Cornuéjols G. Valid inequalities for mixed integer linear programs. Math Program 2008;112:3–44.

[6] Avriel M. Nonlinear programming: analysis and methods. Courier Dover Publications; 2003.

[7] Land AH, Doig AG. An automatic method for solving discrete programming
problems. In: 50 Years of integer programming 1958–2008. Springer; 2010. p. 105–32.

[8] Simpson AR, Dandy GC, Murphy LJ. Genetic algorithms compared to other techniques for pipe optimization. J Water Resour Plann Manage 1994;120:423–43.

[9] Spall JC. Introduction to stochastic search and optimization: estimation, simulation, and control, vol. 65. John Wiley & Sons; 2005.

[10] Back T. Evolutionary algorithms in theory and practice. Oxford Univ. Press;1996.

[11] Talbi E-G. Metaheuristics: from design to implementation, vol. 74. John Wiley & Sons; 2009.

[12] Scharf I, Ovadia O. Factors influencing site abandonment and site selection in a sit-and-wait predator: a review of pit-building antlion larvae. J Insect Behaviour 2006;19:197–218.

[13] Grzimek B, Schlager N, Olendorf D, McDade MC. Grzimek’s animal life encyclopedia. Michigan: Gale Farmington Hills; 2004.

[14] Goodenough J, McGuire B, Jakob E. Perspectives on animal behaviour. John Wiley & Sons; 2009.

[15] Seyedali Mirjalili, The Ant Lion Optimizer, Advances in Engineering, Software 83 (2015) 80-98

# Chapter 8

Comparative Result Analysis of Different Algorithms

## 8.1 Introduction

In this thesis, the tuning of the PD Controller is made by different Algorithms, namely a) Particle Swarm Optimization [Chapter 4], b) Moth Flame Optimization [Chapter 5], c) Grey Wolf Optimization [Chapter 6] and d) Ant Lion Optimization [Chapter 7]. Here the comparative study of these algorithms over the manually tuned controller has been observed. But it was also noticed that one out of these four algorithms outperforms the other algorithms. Here all the four algorithms are compared simultaneously for actual process and identified model of both the QUBE-Servo2 and Rotary Servo Base SRV02.

## 8.2 Parameters for different algorithm

### 8.2.1 Computer Environment

All these algorithms are executed on the same environment i.e. same computer with no other bulk process running in the background (other than necessary background Operating System & Matlab). Matlab R2016a has been used for all these simulation.

### 8.2.2 Algorithm Parameters

| | |
|---|---|
| Population Size: | 30 |
| Maximum iteration: | 100 |
| Objective Function: | ITAE |
| No. Decision variables: | 2 |

## 8.3 Overall Performance

Responses of the first order integrating identified model of QUBE-Servo2 and SRV02 for the position control have been observed under Particle Swarm optimization (PSO), Moth Flame Optimization (MFO), Grey Wolf optimization (GWO) and Ant Lion Optimization (ALO). Responses for QUBE-Servo2 under these four algorithms have been shown in Fig 8.1 and Fig 8.2.The responses for Rotary base servo SRV02 under these four algorithms have been shown in Fig 8.3 and Fig 8.4 .The performance indices of the processes for different controllers are given in Table 8.1 – Table 8.4 .Performance analysis reveals that all these four algorithms based controllers are capable of providing acceptable and improved performance during both set point change and load disturbance. Responses of the identified model and actual process of QUBE-Servo2 and SRV02 for four algorithms have been shown as follows.

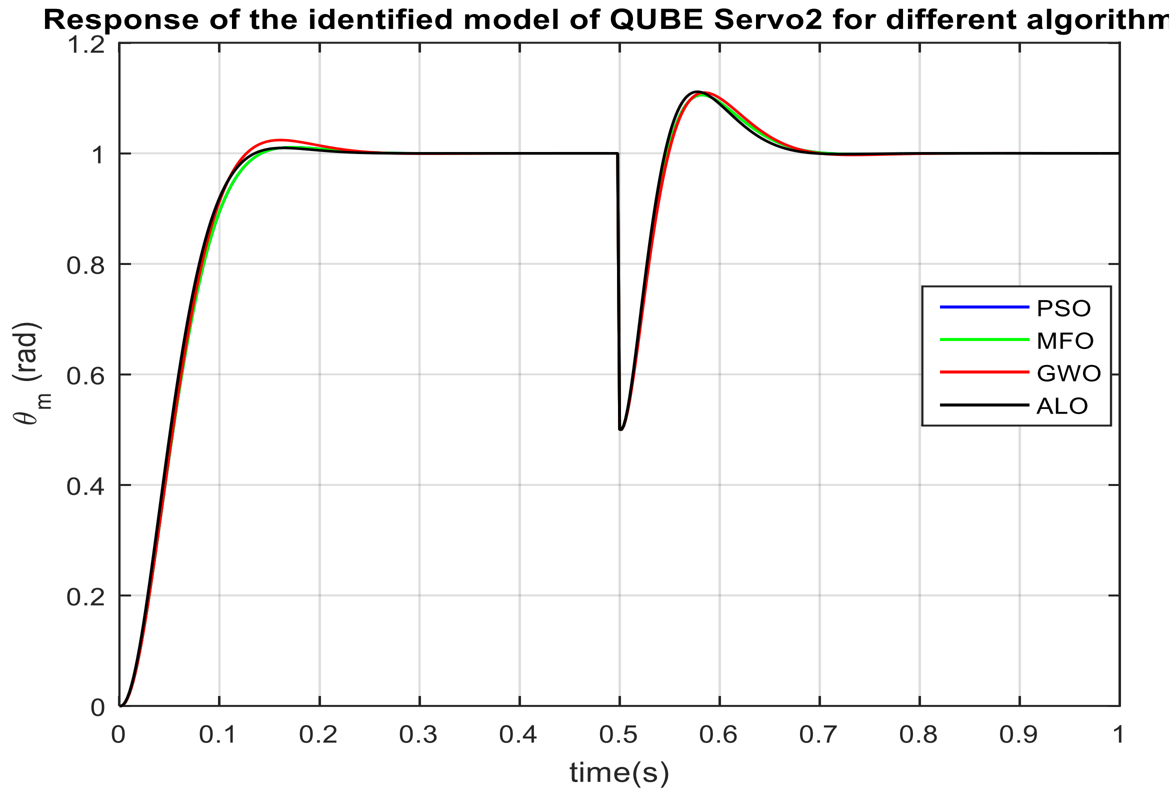

**Fig 8.1:** Responses of PSO-PD vs MFO-PD vs GWO-PD vs ALO-PD for the identified model of QUBE-Servo 2

| Objective Function | QUBE Servo2 identified model Characteristics | PSO | MFO | GWO | ALO |
|---|---|---|---|---|---|
| ITAE | Rise Time (ms) | 182.415 | 80.423 | 77.313 | 75.016 |
| | %Overshoot | 0.501 | 1.058 | 1.950 | 1.515 |
| | Peak Time(sec) | 1.498 | 0.174 | 0.160 | 0.164 |
| | ITAE | 0.02266 | 0.01373 | 0.0143 | 0.0131 |
| | IAE | 0.04901 | 0.08026 | 0.08085 | 0.07664 |
| | ISE | 0.008904 | 0.04682 | 0.04667 | 0.4475 |

**Table 8.1:** Performance Comparison of QUBE Servo2 identified model

Responses of the identified model of QUBE Servo2 under PSO-PD, MFO-PD, GWO-PD and ALO-PD has been shown in Fig 8.1.The performance indices of the process for different controllers are listed in Table 8.1. All optimized controllers perform in a better way. The detail analysis reveals that for the position control or PD control of the identified model of this servo motor, Ant lion and Particle Swarm optimized controller shows better performance as they produce lower integral-time-absolute error. The percentage overshoot of particle swarm optimized controller is the least among all the four other controllers but the response is much slower in this case.

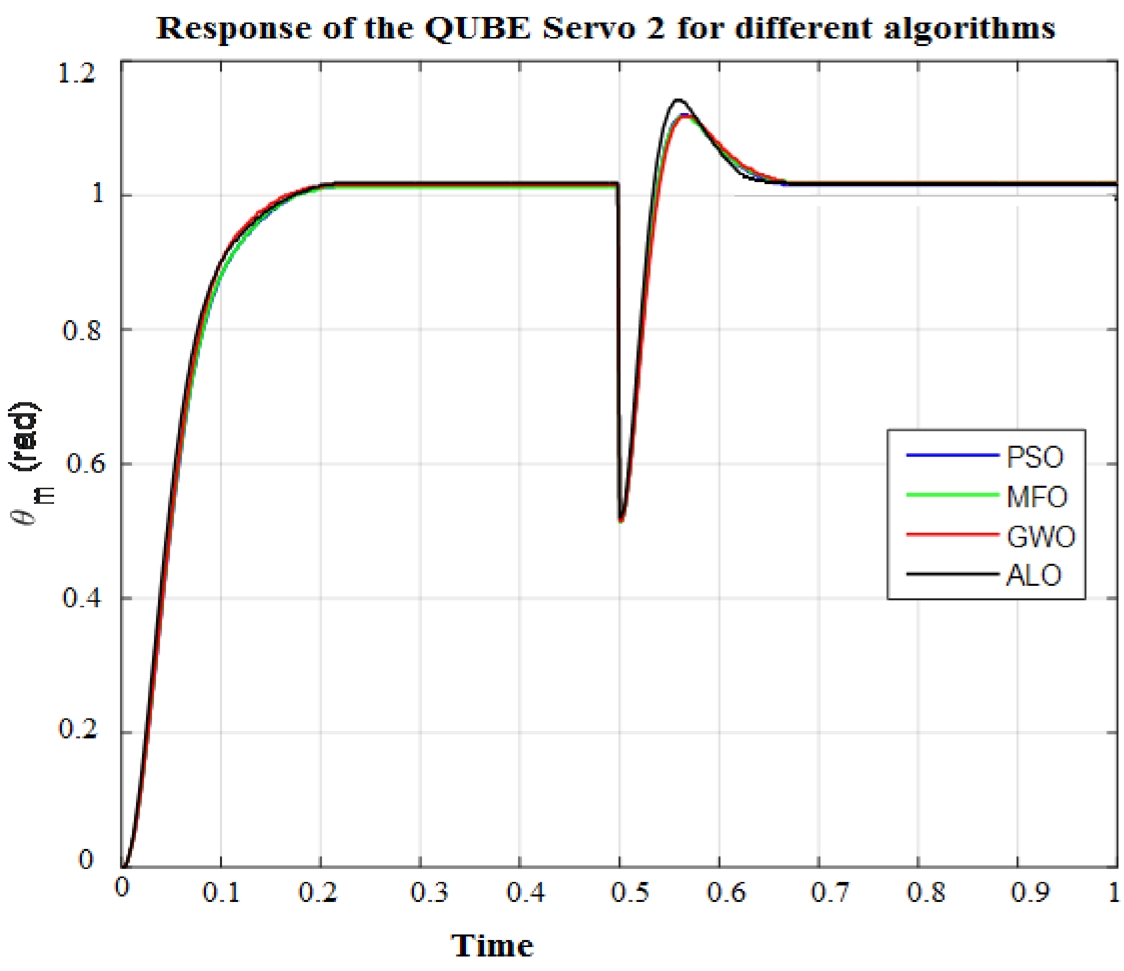

**Fig 8.2:** Responses of PSO-PD vs MFO-PD vs GWO-PD vs ALO-PD for the QUBE Servo 2

| **Objective Function** | **QUBE Servo2 Characteristics** | **PSO** | **MFO** | **GWO** | **ALO** |
|---|---|---|---|---|---|
| ITAE | Rise Time (ms) | 92.731 | 97.072 | 88.115 | 81.948 |
| | %Overshoot | 0.198 | -0.931 | -0.629 | 0.575 |
| | Peak Time(sec) | 0.216 | 0.202 | 0.204 | 0.214 |
| | ITAE | 0.01827 | 0.01883 | 0.01924 | 0.01524 |
| | IAE | 0.08607 | 0.08601 | 0.08581 | 0.07984 |
| | ISE | 0.043 | 0.04284 | 0.04255 | 0.04007 |

**Table 8.2:** Performance Comparison of QUBE Servo2

Responses of the identified model of QUBE Servo2 under PSO-PD, MFO-PD, GWO-PD and ALO-PD have been shown in Fig 8.2.The performance indices of the process for different controllers are provided in Table 8.2. All the optimized controllers perform in a better way. The detail study reveals that for the position control or PD control of the identified model of this servo motor, Ant lion has the best performance as it reduces the integral-time-absolute error to the least as well as  the percentage overshoot and also the response is the fastest with this type of controller. The proposed AL tuning of PD controller is designed towards achieving improved set point tracking along with better load rejection for servo motor based position control System. Thus, the performance of the position control under the Ant Lion algorithm outperforms the other.

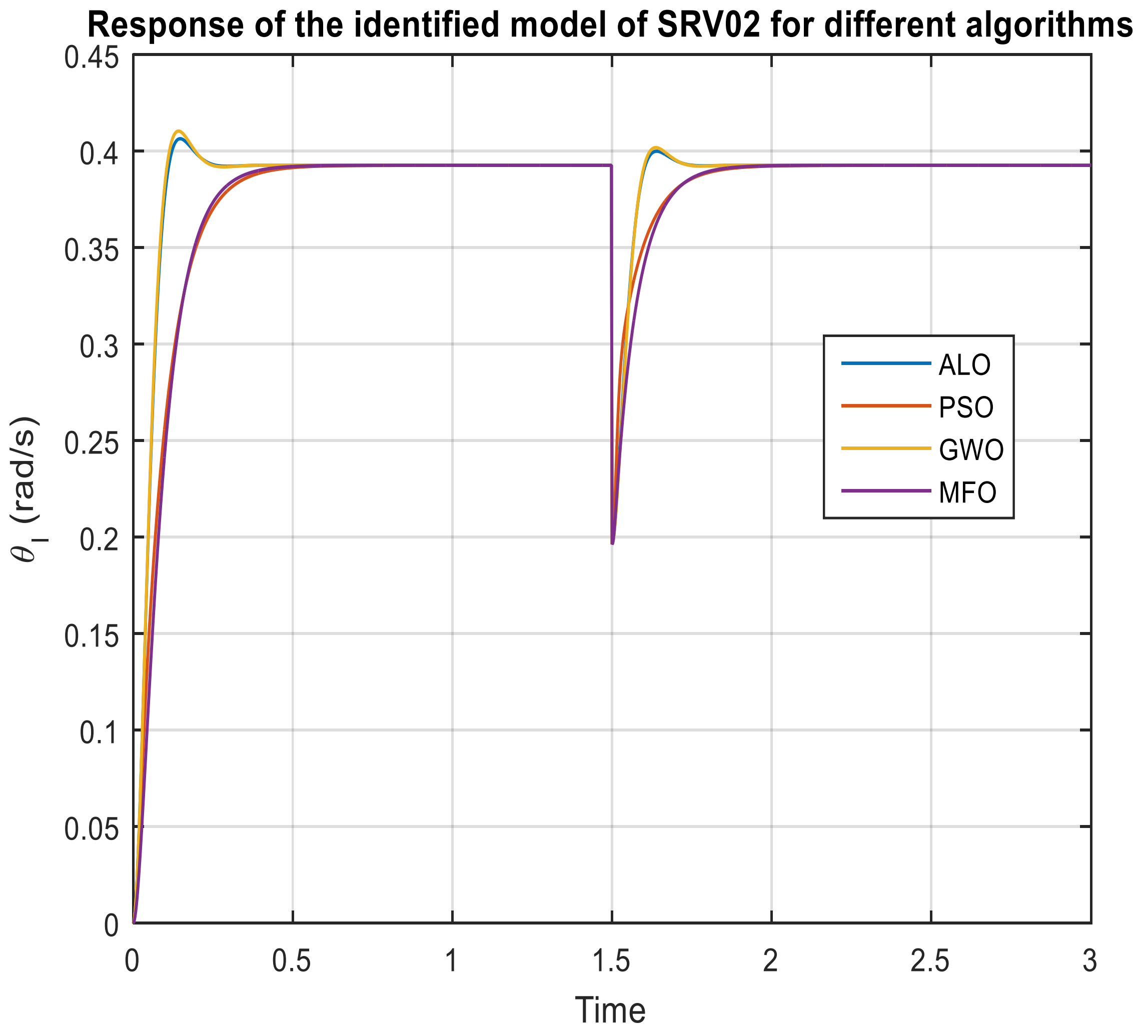

**Fig 8.3:** Responses of PSO-PD vs MFO-PD vs GWO-PD vs ALO-PD for the identified model of Rotary Base servo SRV02

| **Objective Function** | **Rotary servo SRV02 identified model Characteristics** | **PSO** | **MFO** | **GWO** | **ALO** |
|---|---|---|---|---|---|
| ITAE | Rise Time (ms) | 92.731 | 170.96 | 68.126 | 70.635 |
| | %Overshoot | 0.198 | 0.503 | 4.737 | 3.644 |
| | Peak Time(sec) | 0.216 | 0.12 | 0.142 | 0.148 |
| | ITAE | 0.01827 | 0.02668 | 0.01551 | 0.01507 |
| | IAE | 0.08607 | 0.05342 | 0.02996 | 0.02973 |
| | ISE | 0.043 | 0.01055 | 0.0006471 | 0.006483 |

**Table 8.3:** Performance Comparison of Rotary Servo SRV02 identified model

Responses of the identified model of SRV02 under PSO-PD, MFO-PD, GWO-PD and ALO-PD have been shown in Fig 8.3.The performance indices of the process for different controllers are listed in Table 8.3. All optimized controllers perform in a better way. The detail analysis reveals that for the position control of the identified model of this servo motor, Ant lion optimized controller has the best performance as it reduces the integral-time-absolute error to the least among all the other four controllers and also the response is much faster. The percentage overshoot of particle swarm optimized controller is the least among all the four other controllers but the response is much slower in this case.

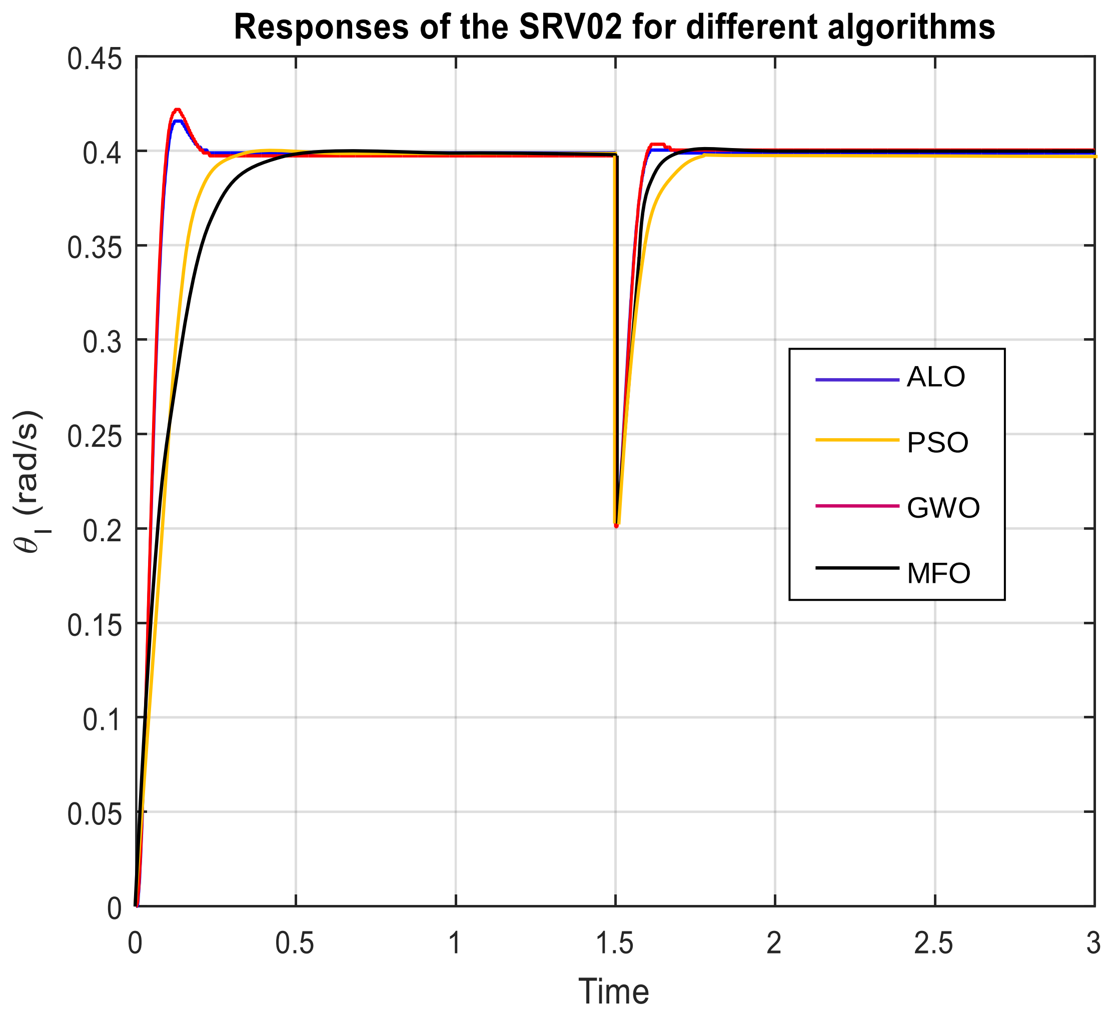

**Fig 8.4:** Responses of PSO-PD vs MFO-PD vs GWO-PD vs ALO-PD of Rotary Base servo SRV02

| **Objective Function** | **Rotary servo SRV02 Characteristics** | **PSO** | **MFO** | **GWO** | **ALO** |
|---|---|---|---|---|---|
| ITAE | Rise Time (ms) | 182.415 | 122.722 | 49.506 | 61.830 |
| | %Overshoot | 0.501 | -0.631 | 7.060 | 4.737 |
| | Peak Time(sec) | 1.498 | 0.357 | 0.122 | 0.119 |
| | ITAE | 0.02266 | 0.08878 | 0.04351 | 0.03994 |
| | IAE | 0.04901 | 0.08879 | 0.004536 | 0.04498 |
| | ISE | 0.008904 | 0.01103 | 0.006419 | 0.006462 |

**Table 8.4:** Performance Comparison of Rotary Servo SRV02

The comparative analysis reveals that Ant lion optimized controller has the best performance as it achieves improved set point tracking along with better load rejection and also its response is much faster. Note that the percentage overshoots of particle swarm optimized controllers and grey wolf optimized controllers are smaller than the others as depicted in the Fig 8.4.

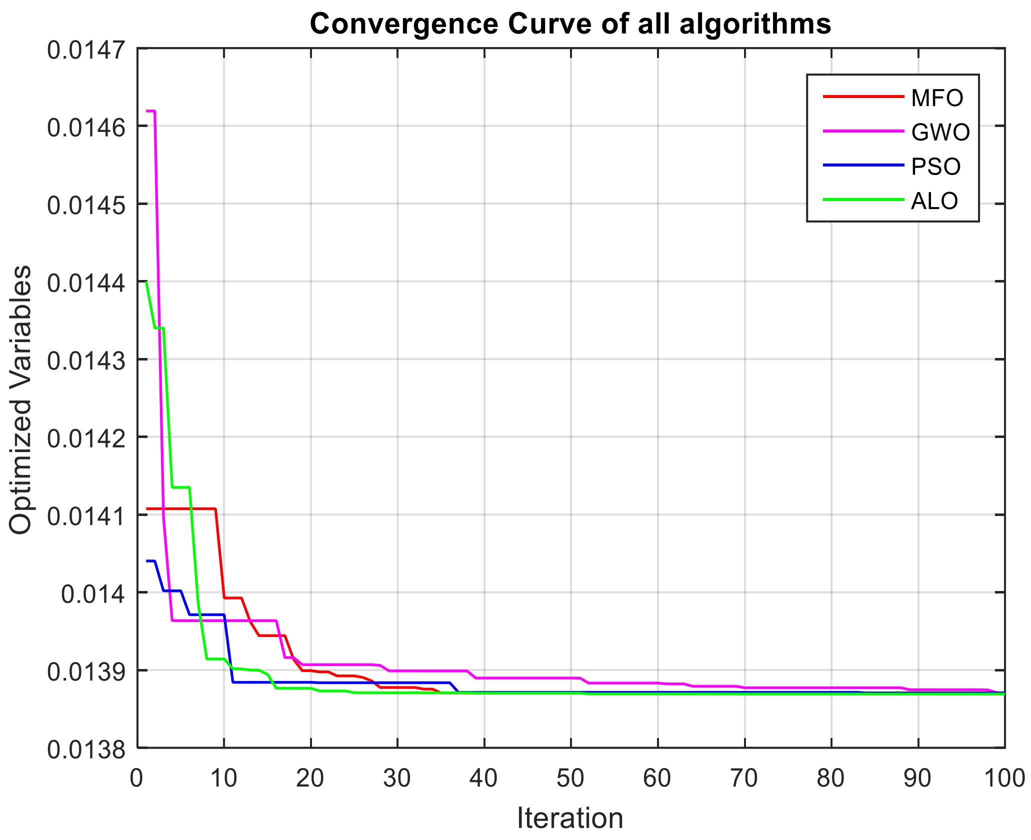

**Fig 8.5:** Comparison of Convergence Curve for different algorithms

In Fig 8.5, typical convergence curves for the same objective function i.e. first order integrating process is shown. A PD controller is optimized by all these four algorithms. From the Convergence curve, we can observe that Ant Lion algorithm converges very fast almost within 20 iterations. While Particle Swarm algorithm and Moth Flame algorithm converge around 35 iterations. And note that Grey Wolf Algorithm takes longer time to converge and also this algorithm converges at a higher value.

So, it can be concluded that ‘Ant Lion Algorithm’ is the best for tuning or optimizing controller for servo position control. Particle Swarm optimization also performed well.

# Chapter 9

## 9.1 Conclusion

We have incorporated four optimization techniques – Particle Swarm Optimization (PSO), Moth Flame optimization (MFO), Grey wolf optimization (GWO) and Ant Lion Optimization (ALO) on an already developed position controller of the Qube Servo and rotary base servo motor by using empirical relations with a view a) to overcome its empirical and percentage overshoot method of choosing appropriate tuneable parameters and b) achieving its optimal performance. Here the two tuneable parameters of the PD controller have been optimized by PSO, MFO, GWO and ALO for two given processes. The derived optimal controllers are tested through extensive simulation experiments, even with the application of load disturbance at a particular time, for checking the robustness. Performances of the optimal controllers PSO-PD, MFO-PD, GWO-PD, and ALO-PD have been compared. The detailed performance analysis revealed that all algorithms provide significantly improved performance in set point tracking along with better load rejection for servo motor based position control System as the initial PD Controller.

## 9.2 Future Scope

In this study we have used four different optimization algorithms to optimize the parameters of both the processes of QUBE Servo 2 motor and rotary based servo SRV02 motor. These algorithms are taken on the basis of their convergence rate and run time. Initially we used some empirical relations and expert knowledge to tune the parameters i.e. $K_p$ and $K_d$ of the PD controller for the position control. While developing PSO-PD, MFO-PD, GWO-PD, and ALO-PD, we have considered the range of variables i.e. $K_p$ & $K_d$ to be in the range of 0-200% of the initial variables. Therefore by increasing the range of the variables we may further improve the performance. Here we consider only ITAE as the objective function. The performance of the controller may be improved if we consider other performance criteria to be optimized simultaneously e.g. IAE, IAE+ITAE or total variation as the objective functions. For future aspect we may study the stability of the process when the controller is optimized by the algorithm used. We have studied the robustness of the controller by application of the load disturbance at a particular time for each process. Finally the performance can be improved by using different optimization algorithms or some other techniques like using machine learning, neural network etc.