PID CONTROLLER TUNING OF 3-PHASE SEPARATOR IN OIL & GAS INDUSTRY USING BACTERIA FORAGING OPTIMIZATION ALGORITHM

By

HO JOON HENG

FINAL PROJECT REPORT

Submitted to the Department of Electrical & Electronic Engineering
in Partial Fulfillment of the Requirements
for the Degree
Bachelor of Engineering (Hons)
(Electrical & Electronic Engineering)

Universiti Teknologi PETRONAS

Bandar Seri Iskandar

31750 Tronoh

Perak Darul Ridzuan

CERTIFICATION OF APPROVAL

PID CONTROLLER TUNING OF 3-PHASE SEPARATOR IN OIL & GAS INDUSTRY USING BACTERIA FORAGING OPTIMIZATION ALGORITHM

by

Ho Joon Heng

A project dissertation submitted to the
Department of Electrical & Electronic Engineering
Universiti Teknologi PETRONAS
in partial fulfilment of the requirement for the
Bachelor of Engineering (Hons)
(Electrical & Electronic Engineering)

Approved:

_____

AP Dr. Irraivan Elamvazuthi
Project Supervisor

UNIVERSITI TEKNOLOGI PETRONAS

TRONOH, PERAK

September 2012

CERTIFICATION OF ORIGINALITY

This is to certify that I am responsible for the work submitted in this project, that the original work is my own except as specified in the references and acknowledgements, and that the original work contained herein have not been undertaken or done by unspecified sources or persons.

_____

HO JOON HENG

ABSTRACT

In oil and gas industry, one of the most important stages in processing petroleum is separation. It can be classified by operating configuration such as vertical, horizontal and spherical or by its function which is 2-phase or 3-phase. In this paper, vertical 3-phase separator will be chosen and researched. 3-phase separator is used to separate water, oil and gas. Gas will be at the top, oil will be the middle layer and water will be at the bottom due to gravitational force and the density of the substance. The objective is to tune the PID controller controlling the level of the water in the separator. Outflow rate of the water from the bottom of the separator will be used to control the water level. Currently there are controlling methods namely PI control using trial and error method, PI control using Butterworth filter design method and IMC method. These methods were having quite high % overshoot and long settling time. So, this paper will introduce Bacterial Foraging Optimization Algorithm (BFOA) in optimizing the parameters for PI control. BFOA mimics the behaviour of the bacteria in searching for highest food concentration which then modified to search the best parameters for the PID controller. BFOA will be able to find the best parameters compared with the conventional methods and show better performance than PI control using trial and error method, PI control using Butterworth filter design method or IMC method. BFOA will be studied and other existing conventional methods as well. Simulation will be done based on the mathematical model of the 3-phase separator.

ACKNOWLEDGEMENT

I would like to thank my supervisor, AP. Dr. Irraivan Elamvazuthi who had been giving me proper guidance and valuable advice in completing this research. I am very sure that it would not be possible without his helping hand. Other than that, this dissertation would not be possible without the constant support from my parents, course mates and friends as well.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

BFOA – Bacterial Foraging Optimization Algorithm

PID – Proportional-Integral-Derivative

GA – Genetic Algorithm

ACO – Ant Colony Optimization

PSO – Particle Swarm Optimization

IMC – Internal Model Control

GUI – Graphic User Interface

CHAPTER 1 -
INTRODUCTION

1.1 Background of Study

Proportional-Integral-Derivative (PID) control is the most commonly used control algorithm in industry such as power plant, oil and gas as well as bio-medical like prosthetics. The controller's performance is decided by the control parameters of K_p, K_i, and K_d of the PID controller. In this project, a 3-phase separator will be focused and discussed. There are a few methods in selecting the proper parameters which are manual method, Zeigler Nichols, Cohen Coon and Ciancone tuning method. Manual method is where experience personnel have to select the optimum parameters by trial and error as well as observing the output response of the system. Zeigler Nichols tuning method requires trial and error to obtain the ultimate gain and period. Then, some minor calculation will be done in selecting the parameters. Cohen Coon and Ciancone tuning method only applies to first order processes with dead time and a series of calculation need to be done in selecting the parameters. In terms of 3-phase separator, the conventional methods used are IMC method, PI controller using Butterworth filter design method and PI controller using trial and error method. All these methods require time as well as cost and the performance of the PID controller might not be satisfying.

Currently, there are intelligent algorithms that can select and optimize the parameters of the plant, for example: Genetic Algorithm (GA), Particle Swarm Optimization (PSO), Ant Colony Optimization (ACO) and Bacterial Foraging Optimization Algorithm (BFOA) [13]. These techniques find and optimize the parameters faster than the conventional tuning methods and very cost effective. This project will explore and study on Bacterial Foraging Optimization Algorithm (BFOA). BFOA applies the behavior a swarm of bacteria searching for nutrients in a manner to maximize energy obtained per unit time.[11]

1.2 Problem Statement

A desired PIC controller will have a low or no percentage overshoot, integral squared error (ISE) and settling time. The conventional tuning method such as manual tuning, IMC method, trial & error and Butterworth filter design can help in selecting parameters but the performance might be not satisfying. Besides, time and cost play an important role and the conventional tuning method might need a lot of time and cost.

Intelligent algorithm could save time and cost in selecting parameters as well as optimizing parameters which could produce desirable results. However, there are different values of parameters and performance for different techniques. The speed of convergence is different as well. From some findings, the performance of Bacterial Foraging Optimization Algorithm (BFOA) is slightly better than GA [4] and PSO [13].

Therefore, Bacteria Foraging Optimization Algorithm (BFOA) will be explored and studied as well as compared with other existing techniques.

1.3 Objective and Scope of Study

The main objectives of this project are:

- To study mathematical model of 3-phase separator
- To study the working principle of Bacterial Foraging Optimization Algorithm(BFOA)
- To study the performance of other existing intelligent algorithms
- To compare the performance of BFOA with other existing techniques

Scope of study will take in consideration of the percentage overshoot, settling time and ISE after implementing BFOA of the PID controller of different orders and applications. Time taken in optimizing the parameters will also be studied.

CHAPTER 2
LITERATURE REVIEW

2.1 Operation of 3-phase Separator

3-phase separator is used to separate water, oil and gas into 3 different phases where the water will be at the bottom, oil in the middle and the gas at the top. Then, those 3 substances will be removed accordingly. Figure 2.1 is a schematic diagram of a horizontal 3-phase separator.

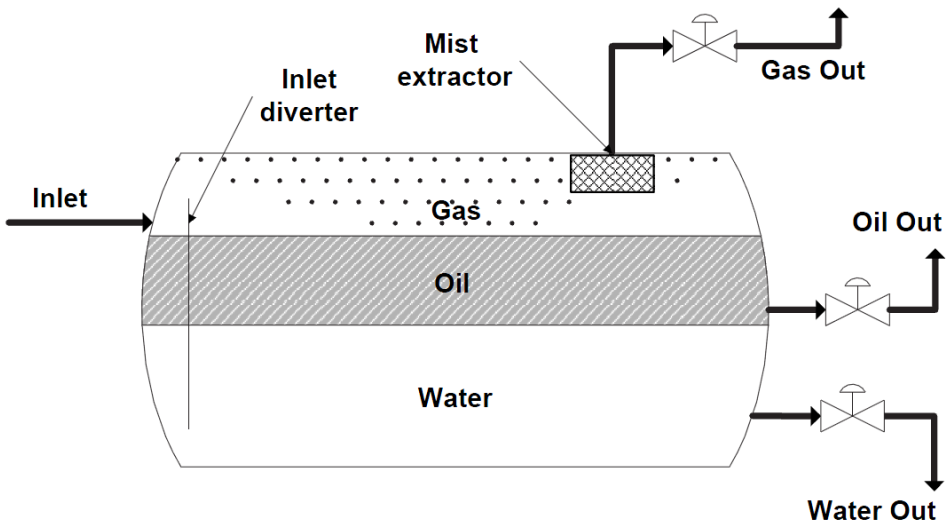


Figure 2.1: Schematic Diagram of a Horizontal 3-phase Separator

The well fluids that were collected from underground would be fed into the 3-phase separator and hit the inlet diverter. There will be a sudden change of momentum of the fluids and causes a gross separation of vapour and liquid. As time goes, the gravitational force will help in pulling the water to the bottom of the separator and pushing the oil on top of the water [19]. This process is called “water-washing”. It will promote the water droplets that trapped in the oil continuous phase to coalescence [19]. As for the gas, some of it flows over the inlet diverter and then horizontally through separator and stays above the liquid. When the gas is flowing

across the inside of the separator, gravitational force will again pulls the small drops of liquid that were flowing with the gas and not separated by the inlet diverter into the liquid [19]. However, there will be an amount of small diameter drops that they are not easily separated using gravity. Thus, a coalescing section or mist extractor is used in order to coalesce and remove them before the gas leaves the vessel [19].

Figure 2 shows the simplified separation process. Oil-well fluid with molar flow *Fin* and gas, oil, and water molar fractions *Zg; Zo; Zw* respectively enters the separator [19].

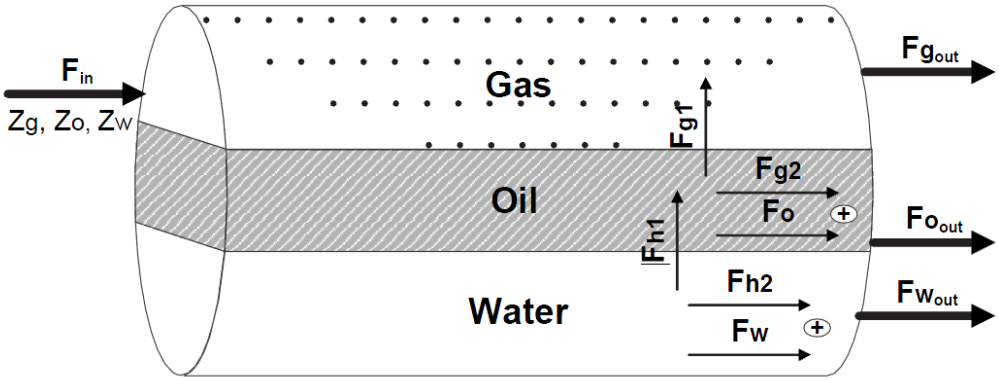


Figure 2.2: Simplified Separation Process

There are 2 main parts for the hydrocarbon which are the first stream *Fh*1 that separated by gravity and enters the oil phase and the second stream *Fh*2 that stayed in the aqueous phase due to incomplete separation [19]. Same goes to the gas component. The first gas stream *Fg*1 flashes out of the oil phase due to the pressure drop in the separator, and the second gas stream *Fg*2 stays dissolved in the oil phase [19]. The oil discharge Fo_{out} from the separator contains the oil component of the separated hydrocarbon *Fo* and the dissolved gas component *Fg*2. The flashed gas Fg_{out} flows out of the separator for further processing [19].

The water and oil levels and the gas pressure inside the separator are controlled by a number of separate control systems [18]. Figure 2 shows the control loop involved in 3-phase separator [18].

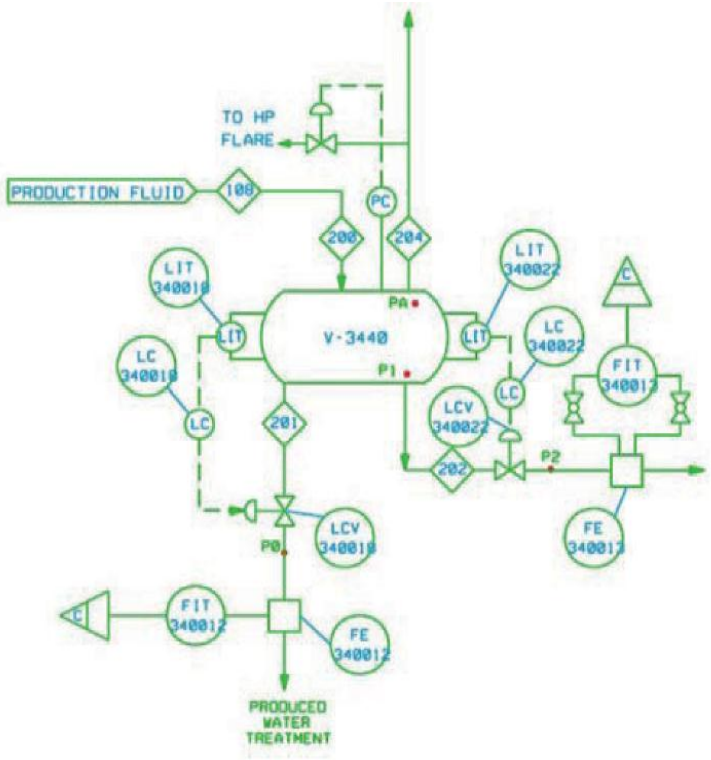


Figure 2.3: P&I Diagram of Relevant Control Loop of 3-phase Separator

It can be observed that a level indicator transmitter (tagged LIT-340018) is employed to measure the water level inside the separator if we focus on the water level control loop. The measured level signal is sent to a level controller, tagged LC-340018. The level controller sends the control signal to a level control valve, tagged LCV-340018. In order to control the water level inside the separator, the LCV-340018 regulates the water outflow. It can be noticed that a flow indicator transmitter, named FIT-340012, is used to measure the water outflow-rate for some other purpose. This measurement is not used by the current level controller [18].

2.2 Behaviour of Bacterial Foraging

For the behavior of the bacteria in searching for nutrients, BFOA mimicked it by focusing in these 4 main sections which are chemotaxis, swarming, reproduction as well as elimination and dispersal.

2.2.1 Chemotaxis

Chemotaxis is the movement of the bacteria in searching for food or nutrients by taking small steps. The bacteria tumble around randomly until the bacteria find higher concentration or gradient of nutrient. Then, the bacteria will run towards that direction until the bacteria reach the highest food concentration.

Chemotaxis simulates a cell to swim and tumble by using flagella. A bacterium can propagate by using 2 different ways biologically. For its entire lifetime, it has 2 modes of operation that kept alternating when necessary which are swimming for a period of time in the same direction or tumbling around randomly.

Let $\theta^i(j+1,k,l)$represents *i*-th bacterium at *j*-th chemotactic, *k*-th reproductive and *l*-th elimination-dispersal step. Besides, let *C* (*i*) equals to the step size of the tumbling in the random direction (run length unit). Then, the mathematical representation of this behaviour is

$$\theta^i(j+1,k,l) = \theta^i(j,k,l) + C(i)\frac{\Delta(i)}{\sqrt{\Delta^T(i)\Delta(i)}} \qquad (1)$$

where Δ represents the vector of random direction which the elements are in [-1, 1] [11].

2.2.2 Swarming

Swarming is the behavior or the tendency of the bacteria to group together. Each bacterium will release signals to other bacteria and to attract or to repel them.

In semisolid nutrient medium, a group of bacteria will show intricate and stable spatio-temporal patterns (swarms). When they are placed in a semisolid matrix with

a nutrient chemo-effecter, they will move up the nutrient gradient by arranging themselves in a travelling ring. When the cells are stimulated by a large amount of *succinate*, they will release an attractant *aspertate* that attracts each other so that they can aggregate into groups as concentric patterns of swarms with high density of bacteria. This behaviour can be represented by using the following mathematical equation.

$$J_{cc}(\theta, P(j,k,l)) = \sum_{i=1}^{S} J_{cc}(\theta, \theta^i(j,k,l)) \quad (2)$$

$$= \sum_{i=1}^{S}\left[-d_{attractant}\exp(-w_{attractant}\sum_{m=1}^{p}(\theta_m - \theta_m^i)^2)\right] + \sum_{i=1}^{S}\left[-h_{repellant}\exp(-w_{repellant}\sum_{m=1}^{p}(\theta_m - \theta_m^i)^2)\right]$$

where $J_{cc}(\theta, P(j,k,l))$ represents the objective function value that later will be added into the actual objective function which is needed to be minimized. This is to form a time varying objective function. Whereas *S* represents the population of the bacteria, *p* represents the dimension or number of variables to be optimized. This p is present in each bacterium and $\theta = [\theta_1, \theta_2, \ldots, \theta_p]^T$. On the other hand, $d_{attractant}, w_{attractant}, h_{repellant}, w_{repellant}$ are different coefficients that should be selected properly [1, 9]. [11]

2.2.3 Reproduction

Reproduction is the behavior of bacteria where weaker half of the group of bacteria will die and the other stronger half will split and reproduce asexually. The number of bacteria of the group will remain unchanged.

For the given *k* and *l*, and for each *i* = 1, 2, ..., *S* , let

$$J_{health}^i = \sum_{j=1}^{N_c+1} J(i,j,k,l) \quad (3)$$

represents the health of the bacterium *i. It is a* measure of the amount of nutrients it had obtained over its lifetime and the ability at avoiding noxious substances. Then, by using the cost J_{health}, the bacteria and chemotactic parameters $C(i)$ are sorted by ascending order. The higher the cost J_{health}, the lower the health.[11]

2.2.4 Elimination and Dispersal

Elimination and dispersal is the probability of a bacterium being eliminated in a group and if elimination is happened, another bacterium will be dispersed to a random location in the optimization domain.

2.2.5 Combination of All 4 Parts

For each bacterium, the cost of its movements had involved the swarming effect from the other bacteria and the chemotaxis effect. The equations had accounted a whole set of chemotatic steps for each swarming effect done. The equation of the swarming effect is represented by

$$J(i,j+1,k,l) = J(i,j,k,l) + J_{cc}\left(\theta^i(j,k,l), P(j,k,l)\right) \qquad (4)$$

Then for each swarming effect, a set of chemotatic steps will be calculated. Previously, a chemotatic step is represented by

$$\theta^i(j+1,k,l) = \theta^i(j,k,l) + C(i)\frac{\Delta(i)}{\sqrt{\Delta^T(i)\Delta(i)}} \qquad (1)$$

and the cost or $J(i,j,k,l)$ can be calculated from that equation. A series of $J(i,j,k,l)$ will be calculated and at the same time the new value will be compared with the previous one. The lower value or cost will be the better one. Then, another bacterium will be selected and the swarming effect as well as the chemotatic steps will be recalculated and the process repeats for every other bacteria.

Besides, this bacteria movement will be repeated for N_c times in random direction to obtain the best results. After N_c times of looping, the $J(i,j,k,l)$ for each bacteria will be summed up to obtain J_{health}. J_{health} will be sorted and the weaker half will die

and the other half will reproduce asexually and the whole process will start over again.

Elimination and Dispersal process will eliminate a bacterium and disperse another one to its domain with a probability of P_{ed} after N_{re} times of reproduction.

2.3 Relation with PID Controller

The bacteria undergo chemotaxis, swarming, reproduction as well as elimination and dispersal in order to survive and maintain the population by considering the optimum concentration of food, tendency of swarming and the health of the bacteria. Hence, the cost needed to find the food and maintain in a group have to be low. In other words bacteria are using the smallest amount of effort to achieve to best result. The formulas or methods mentioned previously from are combined and shows the behavior of the bacteria.

By implementing this survival technique in searching for the best parameters, Integral Squared Error (ISE) which is represented by J_{health}or cost of the lowest will be chosen. BFOA will implement convergence on the parameter values to the optimum level and produces the best result in a short time. So, the optimum parameter of K_p, K_i and K_d will be chosen.

2.4 Related Studies on BFOA

The Table 1 above shows a short analysis of the journals and research papers that had been read.

Table 2.1: Analysis of the Related Journals and Research Papers

	Journal	Author(s)	Date	Pros & Cons	Notes
1	PID Controllers Design for a Power Plant Using Bacteria Foraging Algorithm [1]	Ahmed Bensenouci, Electrical and Computer Engineering Department, King Abdulaziz University, Jeddah, Kingdom of Saudi Arabia	2011	Small % overshoot, short settling time, PSS not required since there is no oscillation	Uses BFOA and PID on Power Plant.

2	A new power system stabilizer design by using Smart Bacteria Foraging Algorithm [6]	E. Daryabeigi, M. Moazzami, A. Khodabakhshian, M.H. Mazidi, Dept. of Electrical Engineering, Islamic Azad University, Iran	2011	Comparison new SBFOA, BFOA and conventional. SBFOA is better than BFOA and BFOA better than conventional. In some cases, SBFOA shows higher % undershoots.	Uses Power System Stabilizer (PSS) as plant.
3	A novel bacterial foraging algorithm for automated tuning of PID controllers of UAVs [7]	John Oyekan and Huosheng Hu School of Computer Science and Electronic Engineering University of Essex, Wivenhoe Park, Colchester CO3 4SQ, United Kingdom	June 10-23, 2010	Smaller % overshoot, shorter settling time, lesser oscillation compared with Zeigler Nichols tuning method	Only use Chemotaxis without swarming, reproduction , elimination & dispersal). Uses PID on UAV.
4	Bacterial foraging oriented by Particle Swarm Optimization strategy for PID tuning [13]	Wael M. Korani, Hassen Taher Dorrah and Hassan M. Emara, Department of Electrical Power and Machines Engineering, Cairo University, Giza, Eqypt	2009	Combination of BFOA and PSO on 6 different plants. BFOA+PSO is better than BFOA and BFOA is better than PSO.	3 plants use PD. Another 3 plants use PID. Plant types are not stated. 4 4^{th} order, 2 3^{rd} order, 1 2^{nd} order plants.
5	A Fast Bacterial Swarming Algorithm For High-dimensional Function Optimization [16]	Ying Chu, Hua Mi, Huilian Liao, Zhen Ji and Q.H. Wu	2008	Combination of BFOA and PSO. BFOA+PSO is faster than PSO and PSO is faster than BFOA. *High- dimensional functions only, for low dimensional functions not so effective.	Compare the speed of convergence by having 13 sets of different constant values for BFOA. Plant type not stated.
6	Optimization Design of PID Controller Parameters Based on Improved *E.Coli* Foraging Optimization Algorithm [10]	Liu Yijian, School of Electrical & Automation Engineering, Nanjing Normal University. Fang Yanjun, Department of Automation, University of Wuhan.	Sept, 2008	Smaller % overshoot, shorter settling time compared with Zeigler Nichols tuning method	All 3 plants use PID. Plant types not stated. 2 1^{st} order and 2 2^{nd} order plants.
7	Improved Bacterial Foraging Strategy for Controller Optimization Applied to Robotic Manipulator System [9]	Leandro dos Santos Coelho and Camila da Costa Silveira, Pontifical Catholic University of Parana, Brazil	Oct 4-6, 2006	Comparison of different values of constant C(i) (step size of bacterium) and Gaussian and Cauchy probability distribution. Best: C(i)=0.05, Gaussian.	Uses PID in motor of robotic manipulator system. Uses BFOA.

8	Optimum Design of PID Controllers Using Only a Germ of Intelligence [4]	Ben Niu,Yunlong Zhu, Xiaoxian He and Xiangping Zeng, Shenyang Institute of Automation, Chinese Academy of Sciences, Liaoning, China	June 21-23, 2006	Smaller % overshoot, shorter settling time, more robust compared with GA	Plant type not stated. Uses PID and BFOA.
9	Bacterial Foraging Optimization Algorithm: Theoretical Foundations, Analysis, and Applications [11]	Swagatam Das, Arijit Biswas, Sambarta DAsgupta and Ajith Abraham, Dept. of Eelctronics and Telecommunication Engg, Jadavpur University, Kolkata, India		Shows the convergence using BFOA, explain BFOA thoroughly.	Explain theories and analysis. No examples of plants or results.

Topics and authors as well as publication date were mentioned in Table 1. The analysis done was the advantages and disadvantages of BFOA or related studies. The performance and the techniques were compared and the types of plants as well as the order of the transfer function used were mentioned to show the coverage of the techniques.

CHAPTER 3
METHODOLOGY

3.1 Research Methodology

3.1.1 Plant with PID Controller

Below is the block diagram for plant where the algorithm will take the input and find the optimum parameters which are K_p, K_i and K_d. Then, PID controller will take the parameters to control the system.

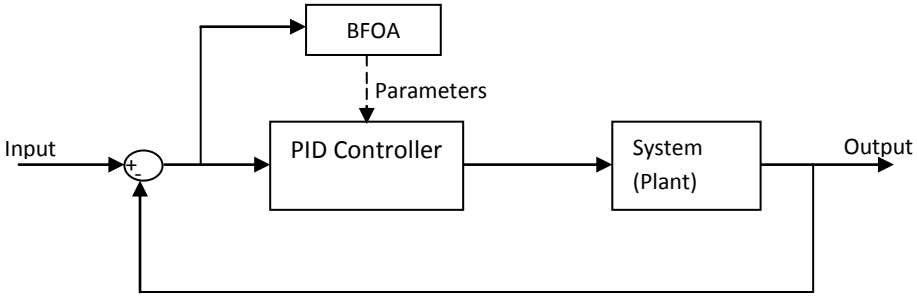
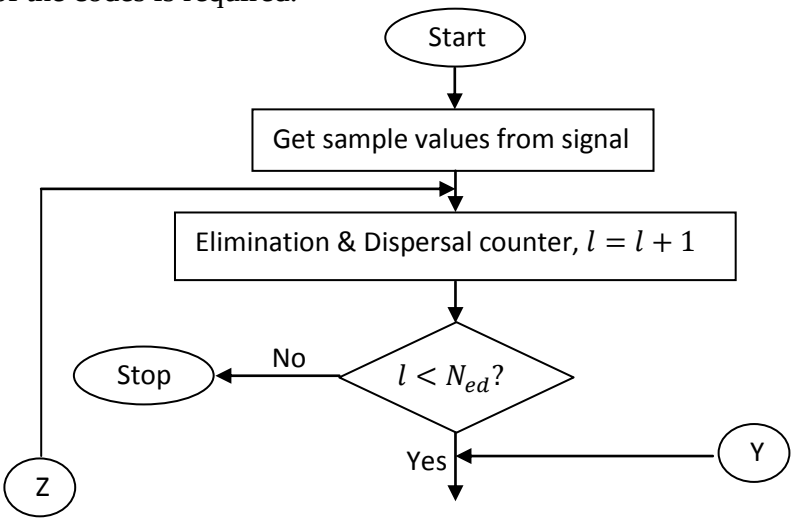


Figure 3.1: Block Diagram of the System with PID Controller and BFOA

3.1.2 Flow of BFOA

In this research, there are codes to be compiled and run in order to optimize and obtain the best parameters. However, before the codes are to be done, a flowchart of the steps of the codes is required.



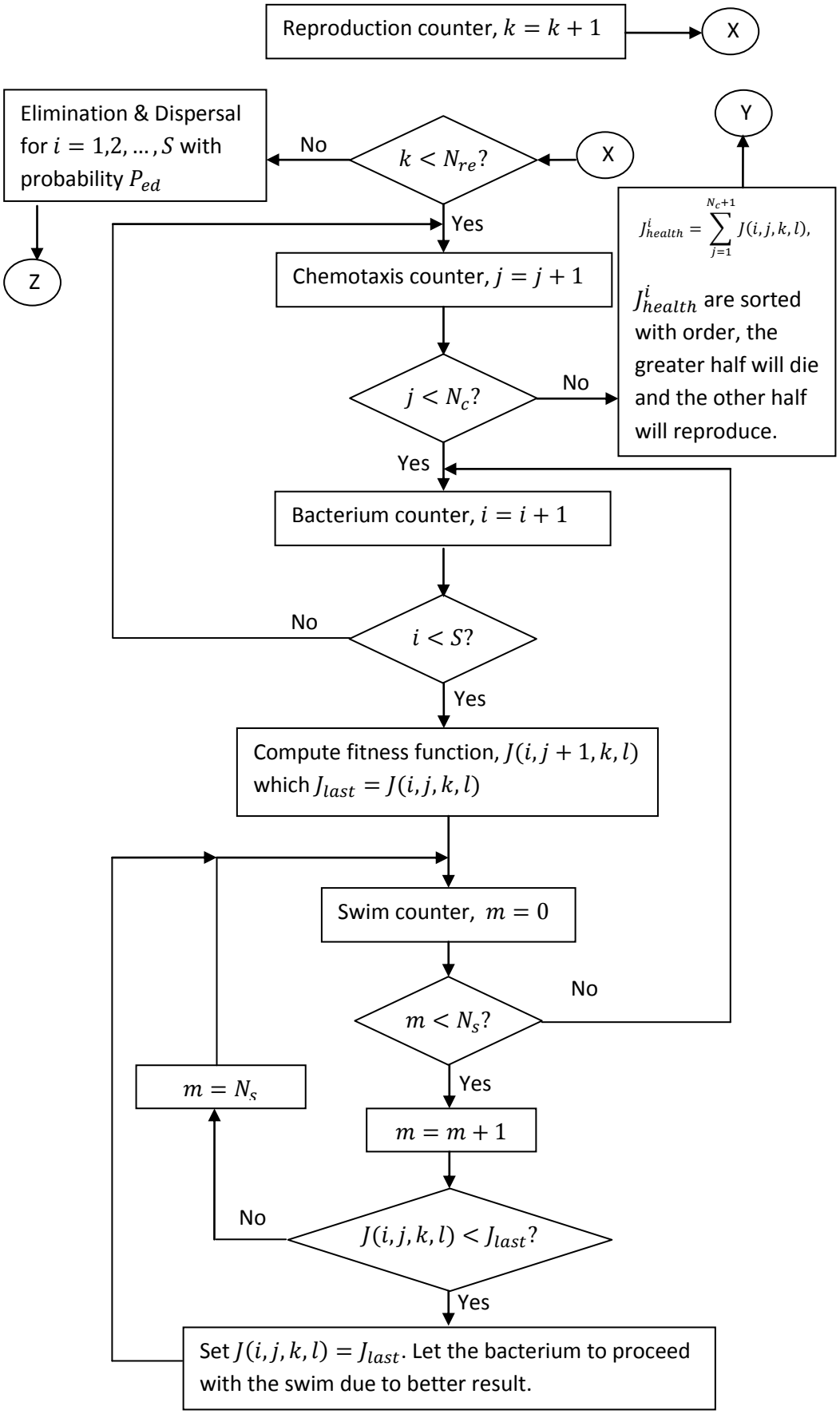


Figure 3.2: Flowchart of the Bacterial Foraging Optimization Algorithm (BFOA)

Figure 5 shows the flow of BFOA by combining all 4 parts which are chemotaxis, swarming, reproduction as well as elimination and dispersal. J_{last} with the smallest value will be produced and chosen since it represents the Integral Squared Error (ISE) or also known as cost.

Please refer to **Appendix A** for **BACTERIA FORAGING OPTIMIZATION ALGORITHM (BFOA) MATLAB CODES.**

3.1.3 Mathematical Model of 3-phase Separator

A 3-phase separator can be modeled and represented by using a transfer function to show its characteristic.

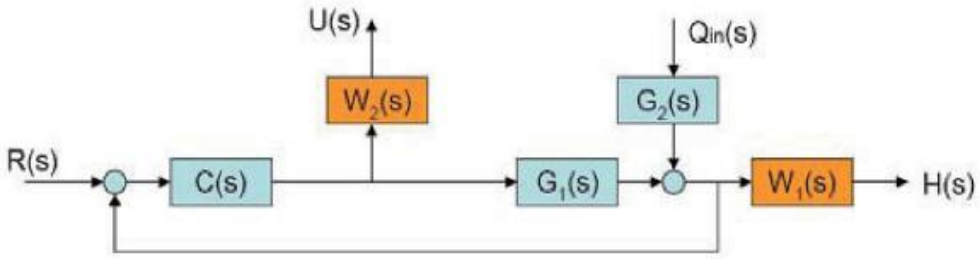


Figure 3.3: Block Diagram of 3-phase Separator with Disturbance, Q_{in} [18]

The figure above shows the block diagram of 3-phase separator with disturbance, Q_{in} included. BFOA is used and implemented as shown in Figure 1 on top of C(s). The input, R(s) and output, H(s) are the desired level and actual level of the water in the separator.

$$G_1(s) \hat{=} \frac{H(s)}{Q_{in}(s)} = \frac{1}{47.55s+1.81}, \quad (5)$$

$$G_2(s) \hat{=} \frac{H(s)}{U(s)} = -\frac{10.82}{47.55s+1.81}. \quad (6)$$

The above equations show the transfer function of the plant and the disturbance [18].

3.2 Project activities

The Figure below shows the guidelines of the steps needed in completing this project. It shows the project activities that should be conducted in this project.

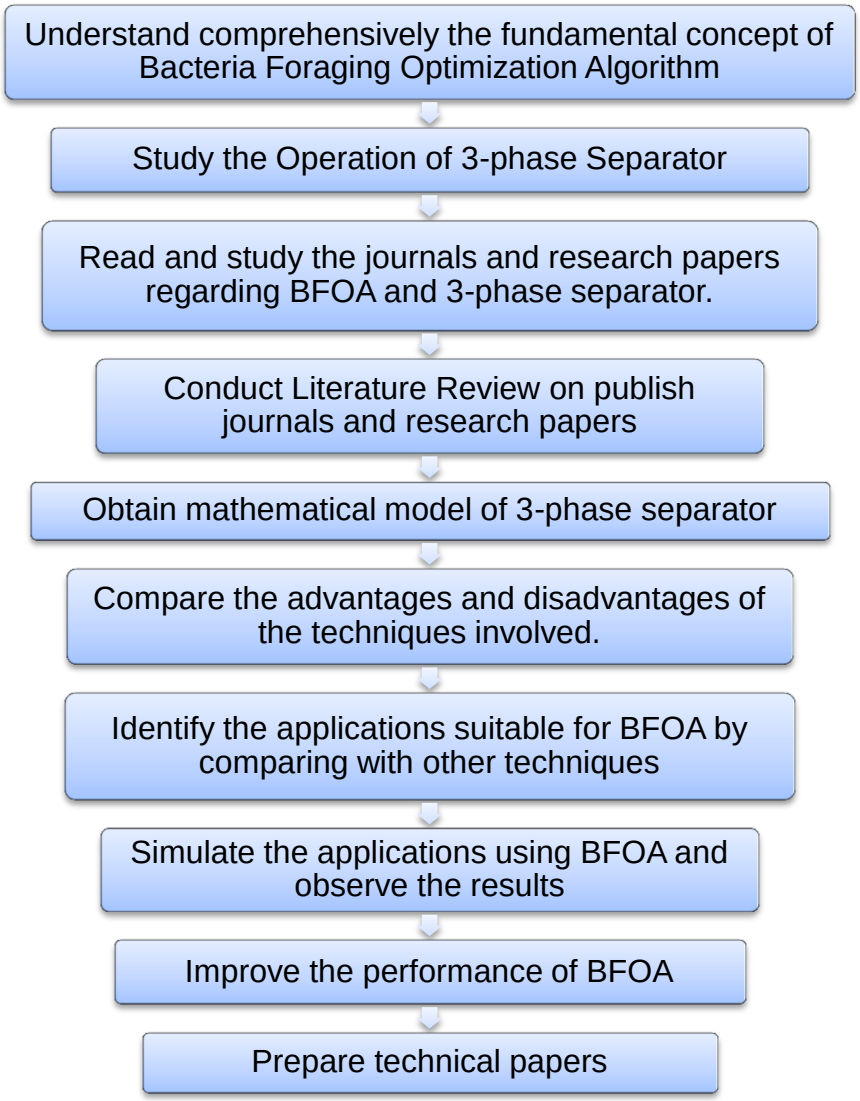
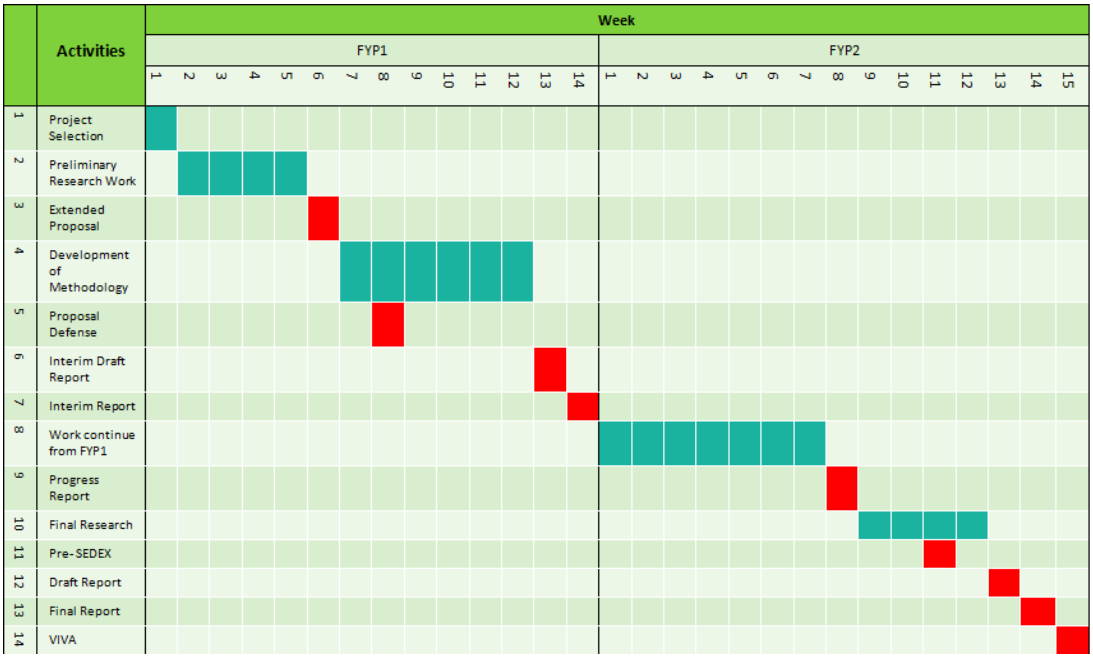


Figure 3.4: The major Project Activities to be done in the Project

3.3 Gantt Chart/Key milestone

The Table 2 below shows the tasks that need to be completed and the dates of completing them. It is a Gantt Chart of the project with the key milestones included to show the progress and schedule of the project.

Table 3.1: Gantt Chart of the Project with Key Milestones



Key Milestone

3.4 Tools

The main software required in this project is:

- MATLAB

CHAPTER 4
RESULTS AND DISCUSSION

4.1 Results

For each of the method, the parameters used were:

Table 4.1: Values of Parameters for each Method

	Kp	Ki
BFOA	-33.898	-0.1065
Trial & Error[18]	-0.7391	-1.582
Butterworth [18]	-1.05	-1.76

And for IMC method [1]:

$$\text{Internal Model Control} = -\frac{47.55s + 1.81}{45.958s} \qquad (3)$$

By using Matlab Simulink, the following system was constructed as shown in Figure 2.1.

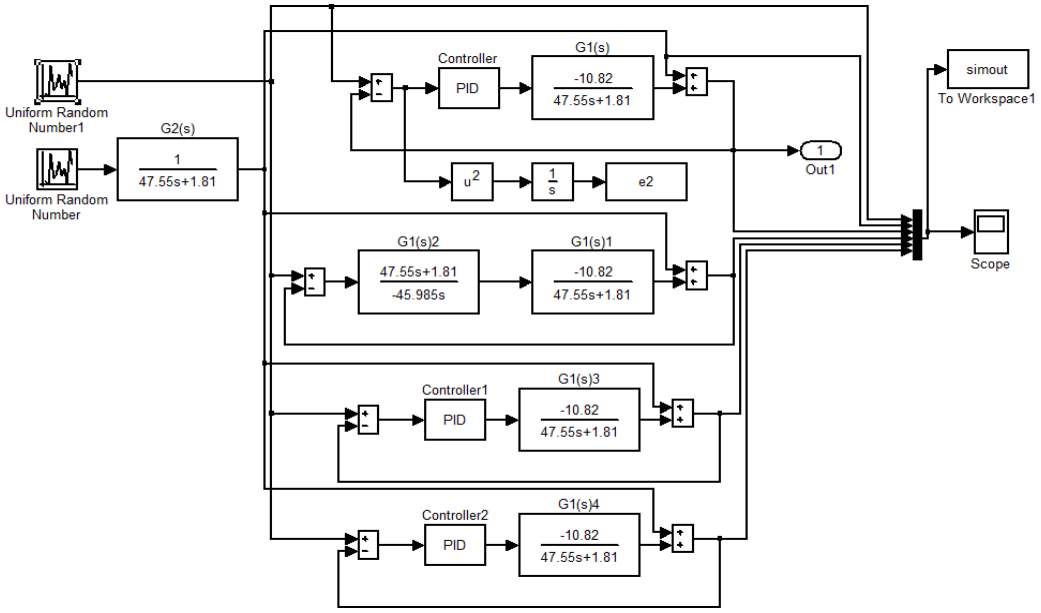


Figure 4.1: Matlab Simulink of Block Diagram Comparing All Methods

From the simulation, the following are the results obtained for each of the methods.

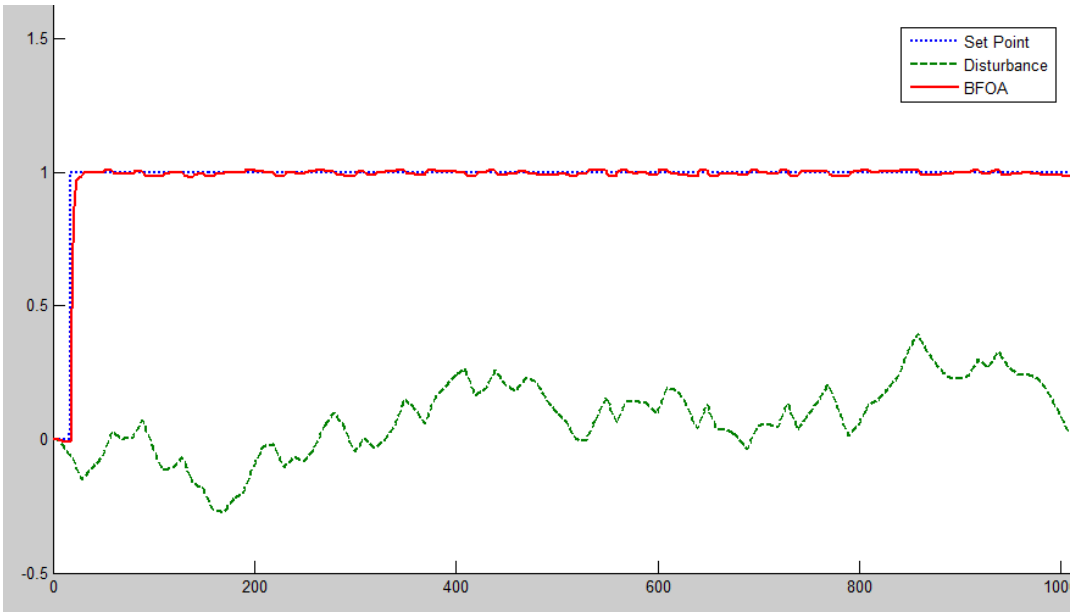


Figure 4.2: Response of Plant using BFOA with Disturbance

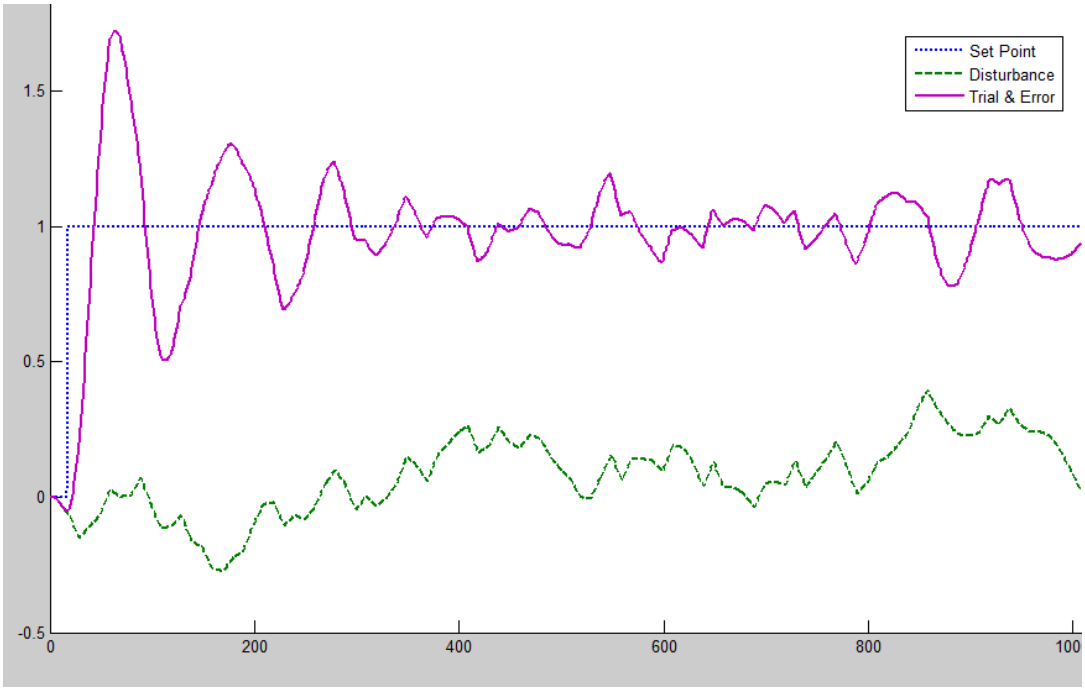


Figure 4.3: Response of Plant using Trial & Error with Disturbance

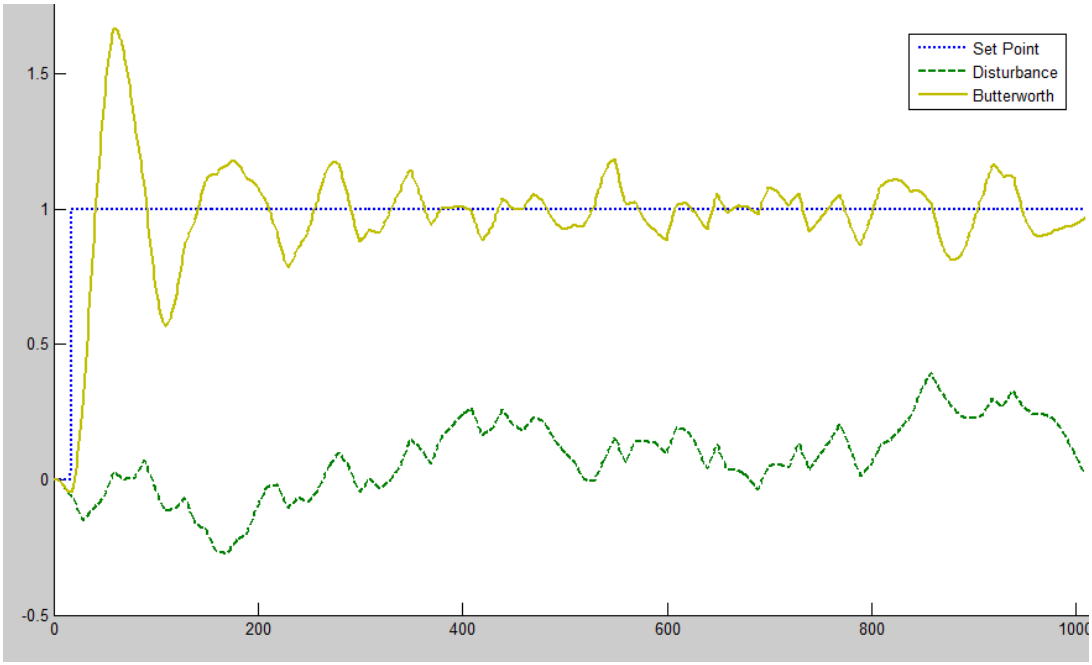


Figure 4.4: Response of Plant using Butterworth Filter Design Method with Disturbance

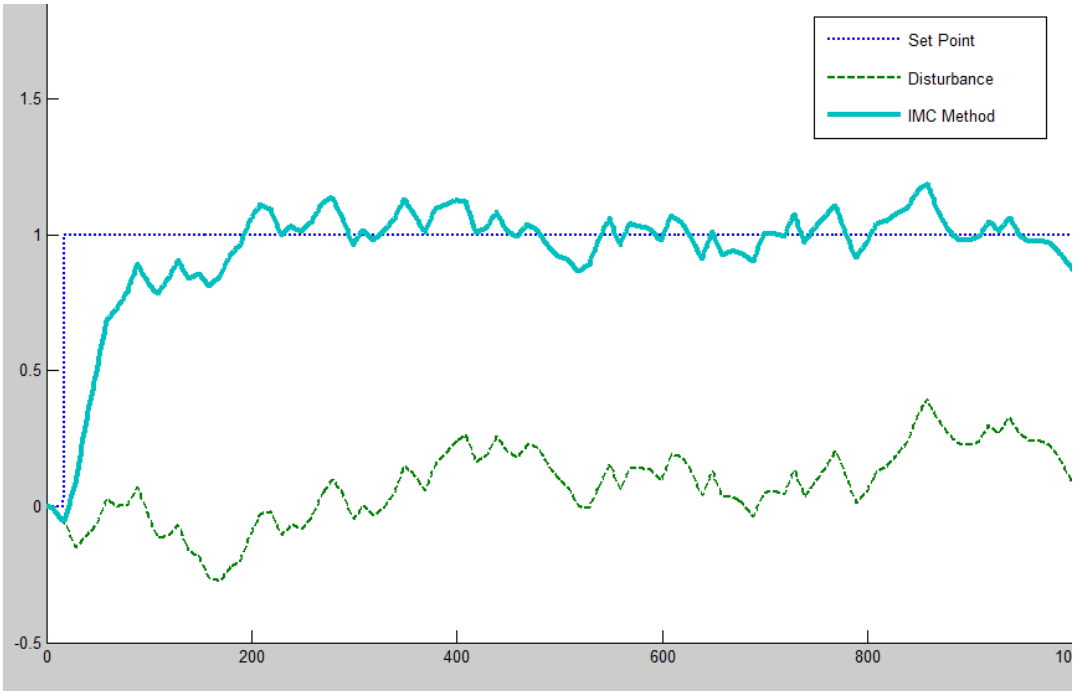


Figure 4.5: Response of Plant using IMC Method with Disturbance

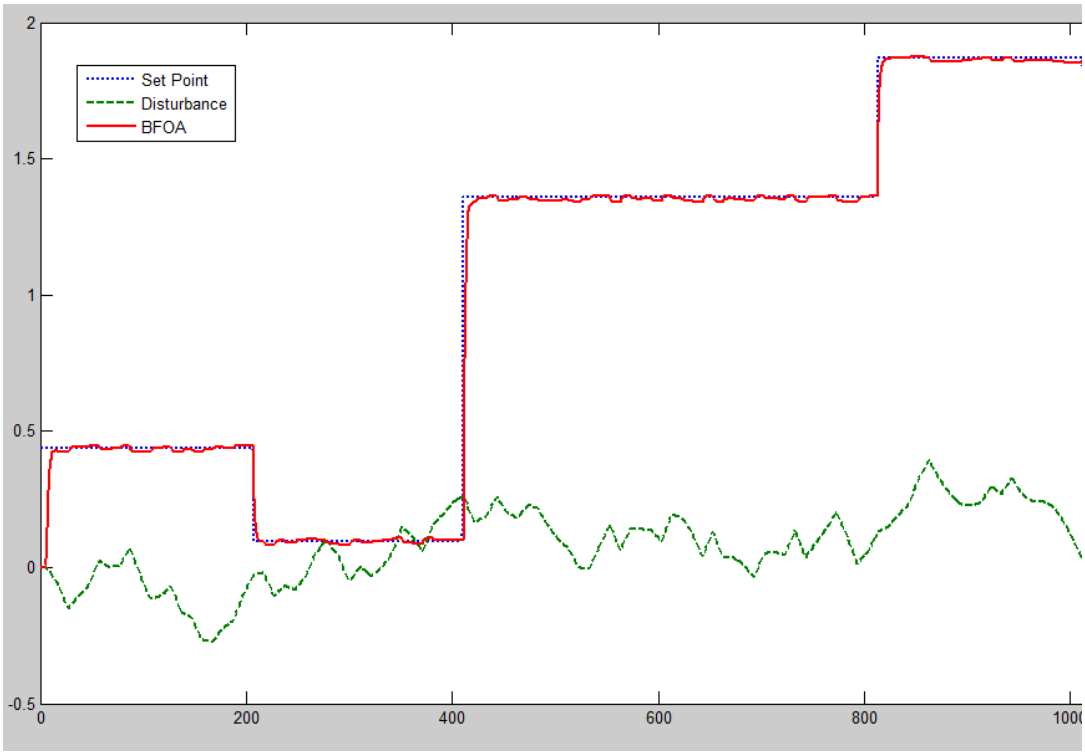


Figure 4.6: Response of Plant using BFOA with Disturbance and Setpoint Changes

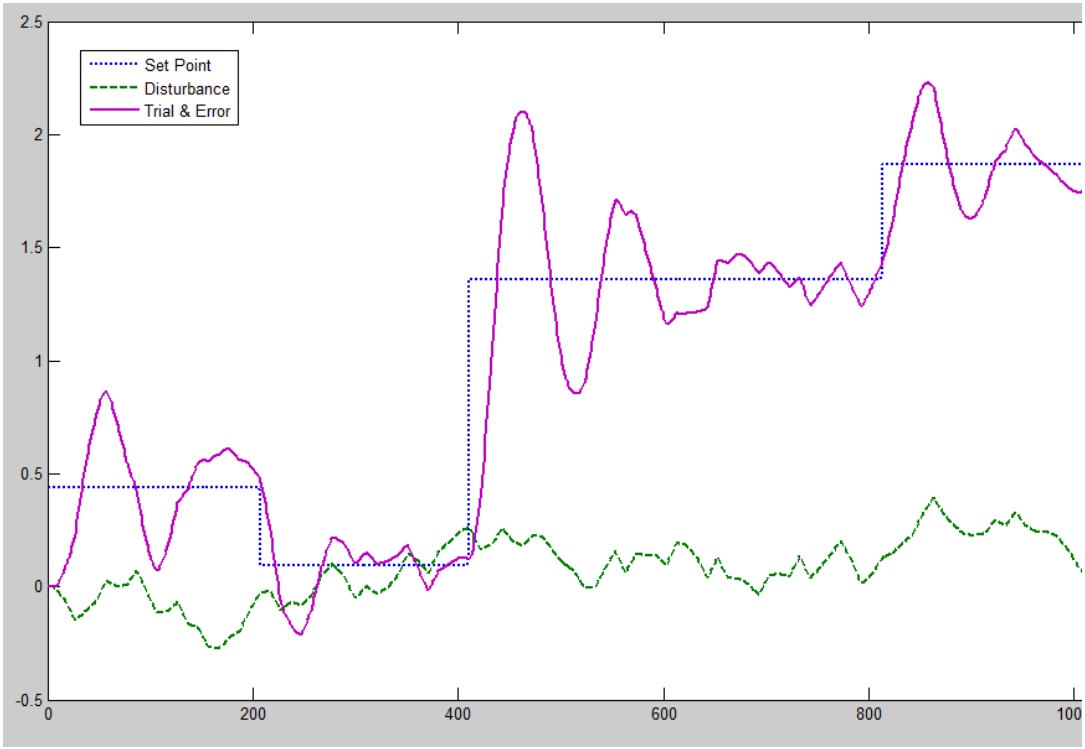


Figure 4.7: Response of Plant using Trial & Error with Disturbance and Setpoint Changes

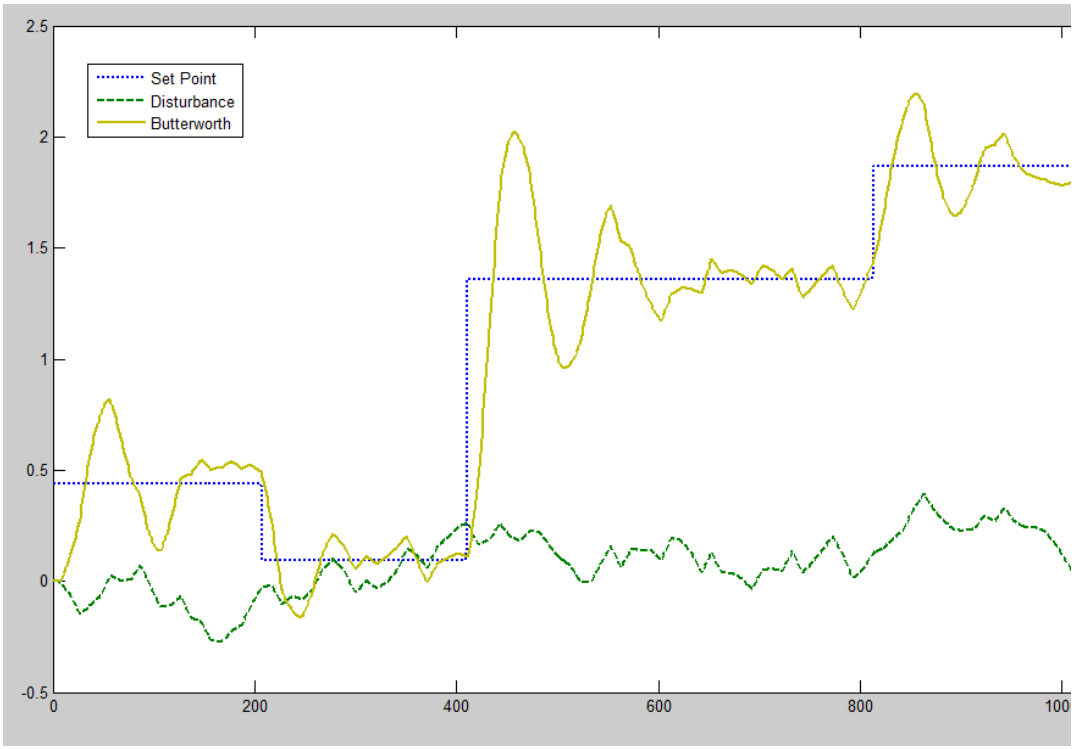


Figure 4.8: Response of Plant using Butterworth Filter Design with Disturbance and Setpoint Changes

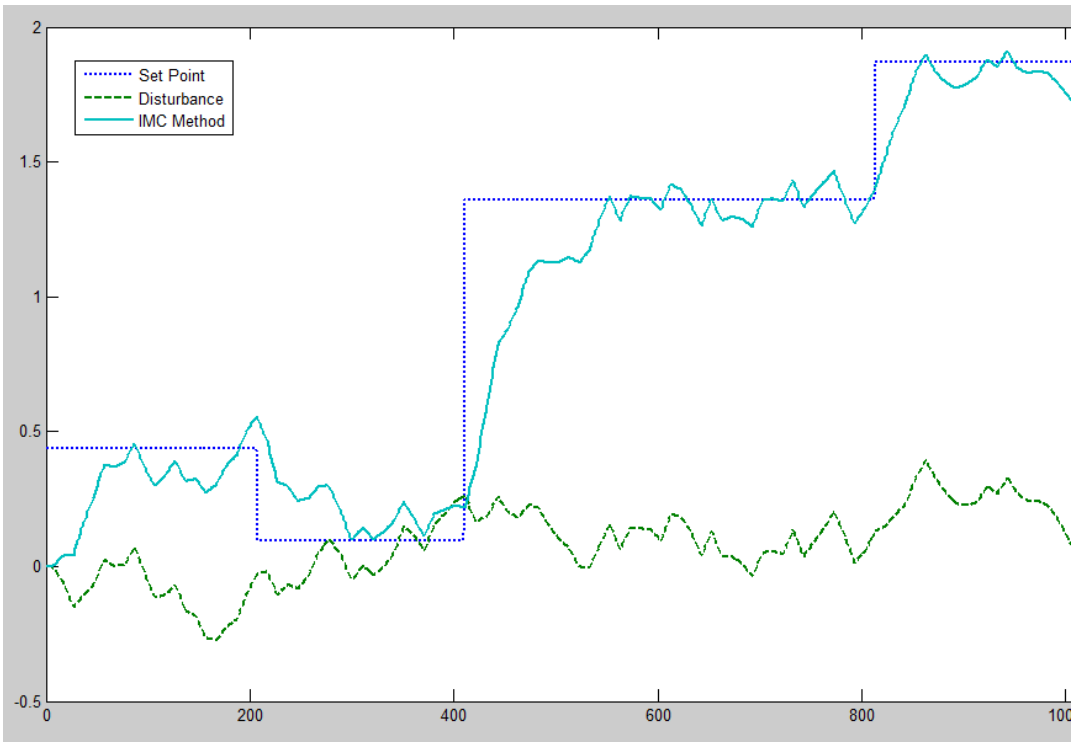


Figure 4.9: Response of Plant using IMC Method with Disturbance and Setpoint Changes

4.2 Discussion and Analysis

Figure 4.2 to Figure 4.5 show the initial set point change then followed by the disturbance due to the input well fluids of the 3-phase separator. Figure 4.6 to Figure 4.9 show the responses which include the input disturbance and the set point changes as well. Both Figure 4.2 and Figure 4.6 show that the responses of using BFOA tuning method produce the best results which are almost the same as the desired results. Trial and error method and Butterworth filter design method produces results that have high overshoots and undershoots. IMC method produces slow responses which have long rise time. IMC method, trial & error and Butterworth filter design method have large settling time or very slow responses compared to BFOA tuning method. The figure below shows the comparison of the conventional methods with the intelligent method.

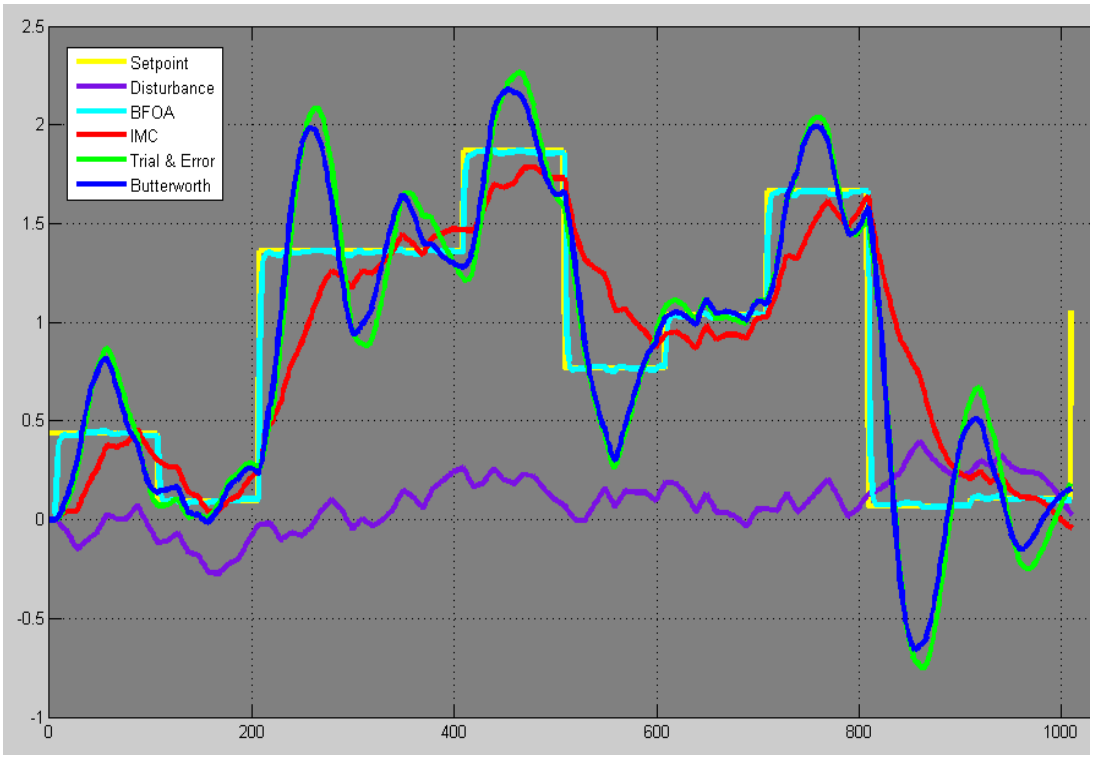


Figure 4.10: Comparison of Response among Conventional methods and BFOA

From the comparison, the response from using BFOA had shown that it copied the value of setpoint very closely while the conventional methods had overshoots and larger settling time. The following figures were obtained by analyzing the responses from various methods and calculated.

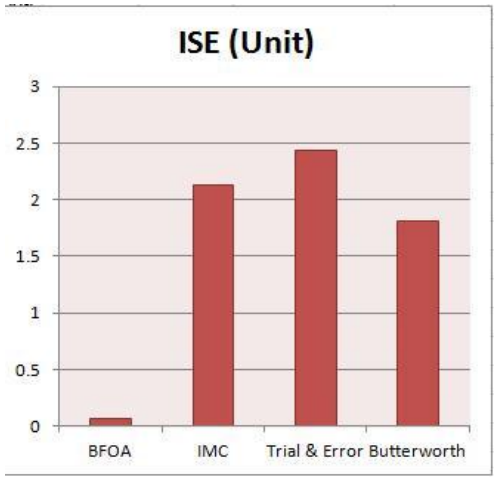


Figure 4.11: Integral Squared Error of Different Methods

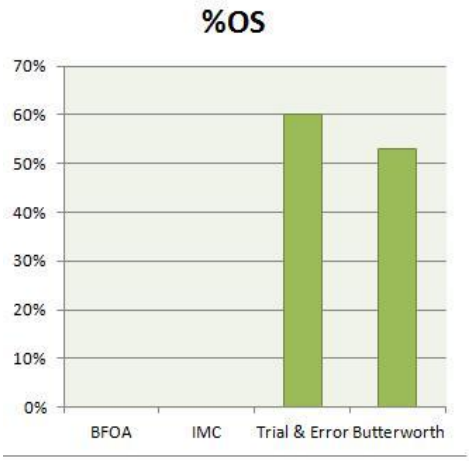


Figure 4.12: Percentage Overshoot of Different Methods

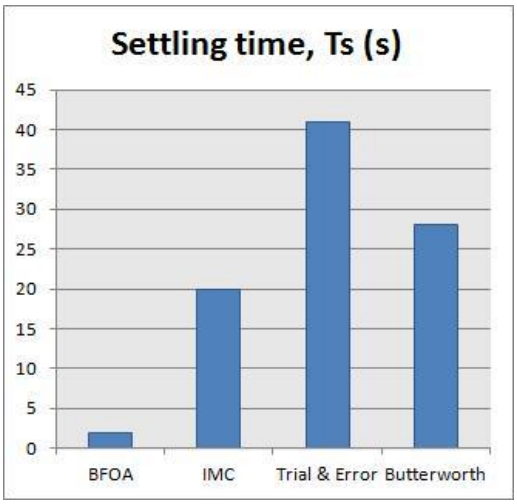


Figure 4.13: Settling Time of Different Methods

Table 4.2: Percentage Improvements of BFOA compared to Conventional Methods

Method	ISE	%OS	Ts (s)	% Improvements
BFOA	0.0663	0%	2	-
IMC	2.125	0%	20	96.88%
Trial and Error	2.434	60%	41	97.27%
Butterworth Filter Design	1.8116	52%	28	96.34%

Figure 4.11, 4.12 and 4.13 had shown the ISE, percentage overshoot and settling time respective for all methods. The lower the values, the better the response is. For these 3 figures, BFOA shows the lowest values. This has proven that BFOA

optimize better parameters compare with these conventional methods. From all of the 4 methods used, BFOA have the most significant improvements by having the least or almost zero % overshoot and the shortest settling time. This had shown that BFOA performs the best of all the methods tested. Table 4.2 was tabulated by calculating the percentage performance improvement based on ISE, percentage overshoot and settling time. It had shown that the overall improvements are more than 90%.

4.3 Graphic User Interface (GUI)

Other than comparing the results of BFOA and the conventional methods, a GUI was developed to ease the user in optimizing the parameters using BFOA. The software developed can optimize parameters for PID, PI or PD controllers with up to 3rd order system.

Please refer to **Appendix B** for **GUI MATLAB CODES**

The following Figure 4.14 is a screen shot of the GUI.

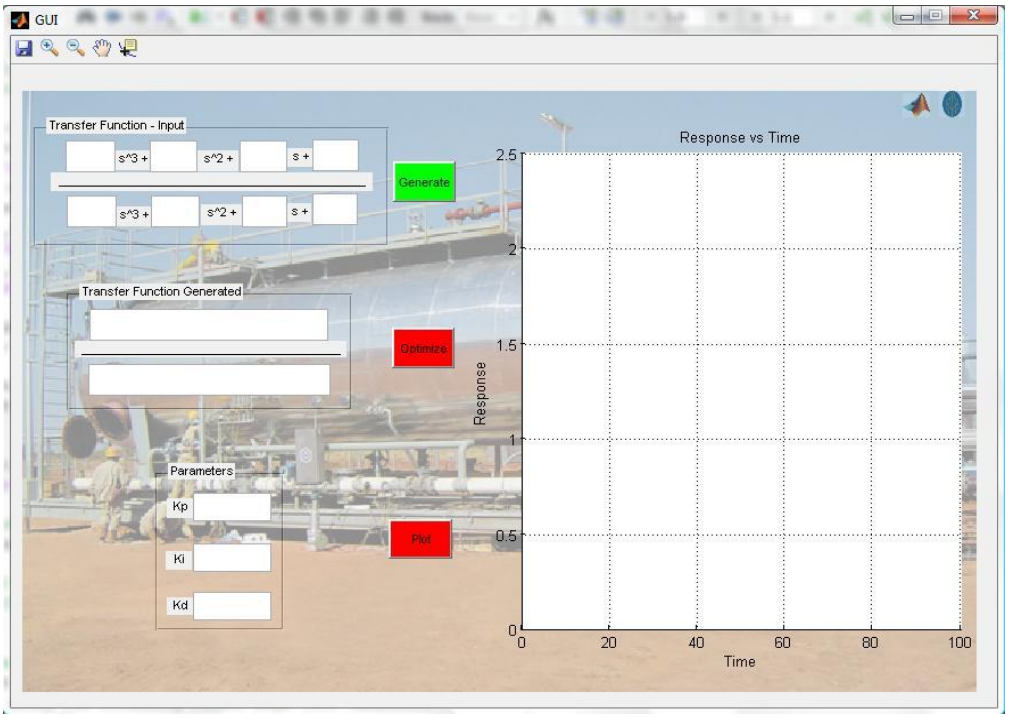


Figure 4.14: Graphic User Interface (GUI) that I had developed in Optimizing Parameters.

Transfer function can be input into the GUI in the first box and then generate the equation to initialize the required parameters. Then, optimization can be done. After that, values of Kp, Ki and Kd will be displayed and a step response of the system can be plotted. There are only 3 push buttons that can be used in the GUI that I had developed. There will be 3 colours for the buttons which are green, red and grey. The push button can only be pushed when it is green in colour. When the button is in grey or red colour, it cannot be pushed. The difference between the grey and red is that if any mistake or wrong data had been keyed into the GUI and the user wanted to change the values, the information with the grey or green button can be changed. Red means that the algorithm is running or the data required is insufficient. The following Figure 4.15 show the transfer function of the system is being input.

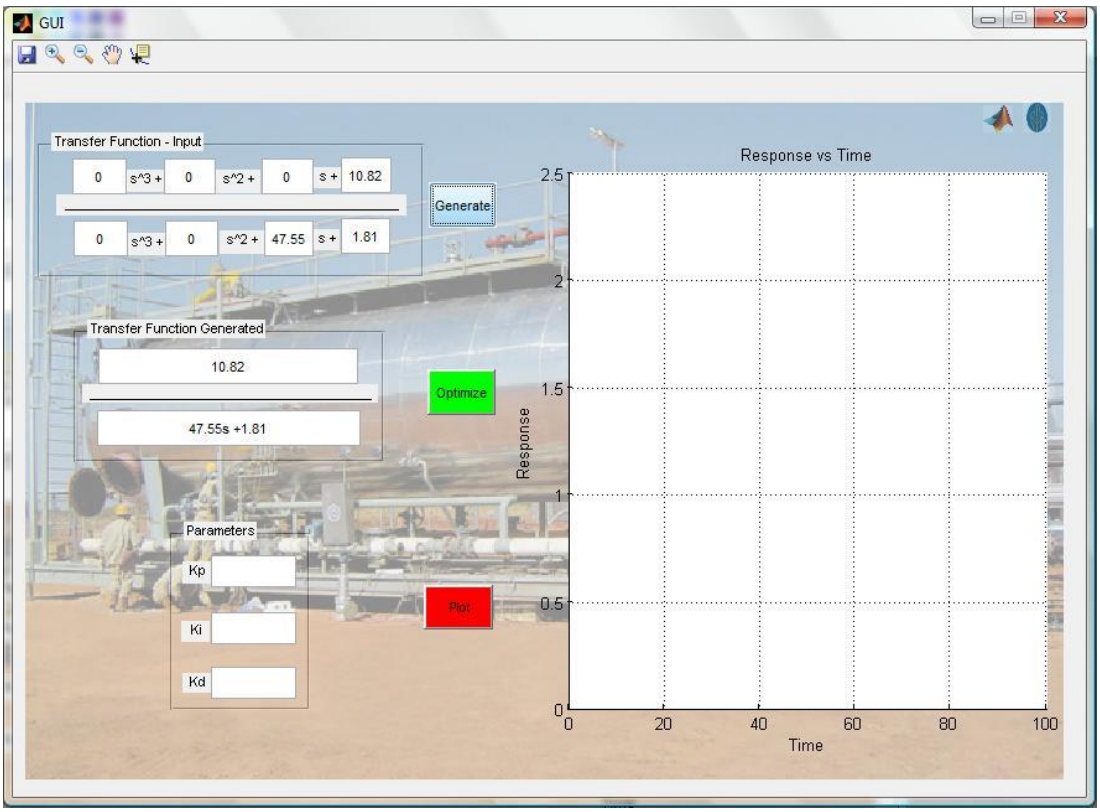


Figure 4.15: GUI – Transfer Function of System being Input and Generated

From Figure 4.15, it can be observed that the push button “Generate” turned from green to grey and push button “Optimize” turned from red to green. It shows that the transfer function is accepted and generated. The transfer function of system can be checked from the “Transfer Function Generated” box. By then, it is ready to

optimize the required parameters. The following Figure 4.16 will show the next process.

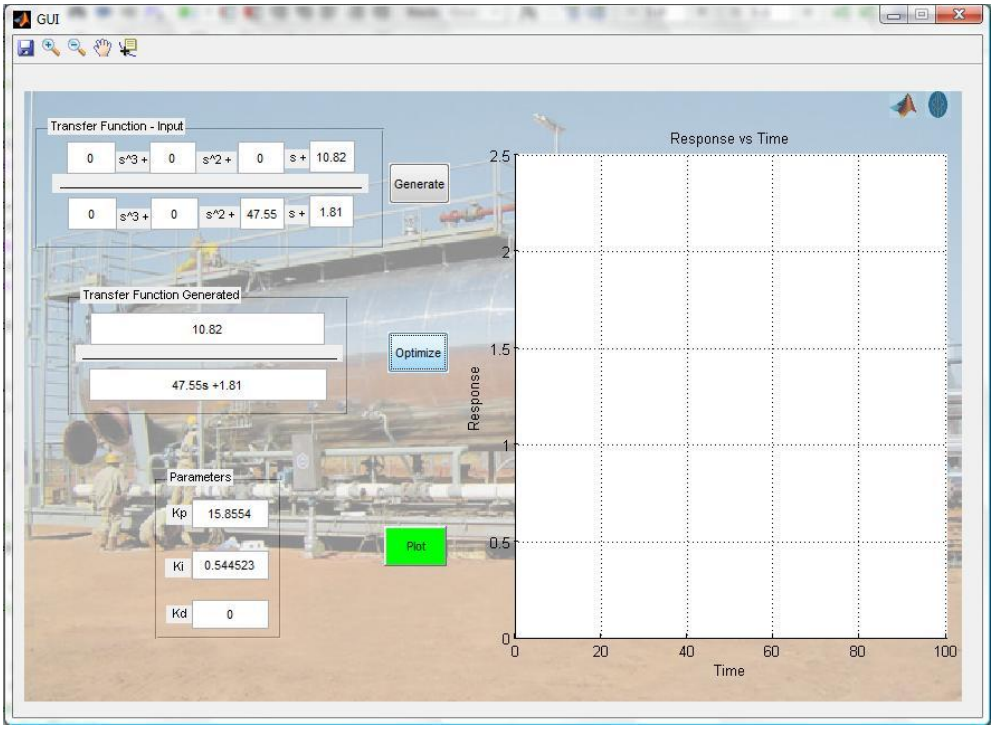


Figure 4.16: GUI – Optimized Parameters Kp, Ki and Kd

The above Figure 4.16 shows that the parameters are being optimized after approximately 20 seconds. Then, the “Optimized” push button will change from green to grey and the “Plot” push button is enabled. After the “Plot” push button is push, Figure 4.17 will show the further results.

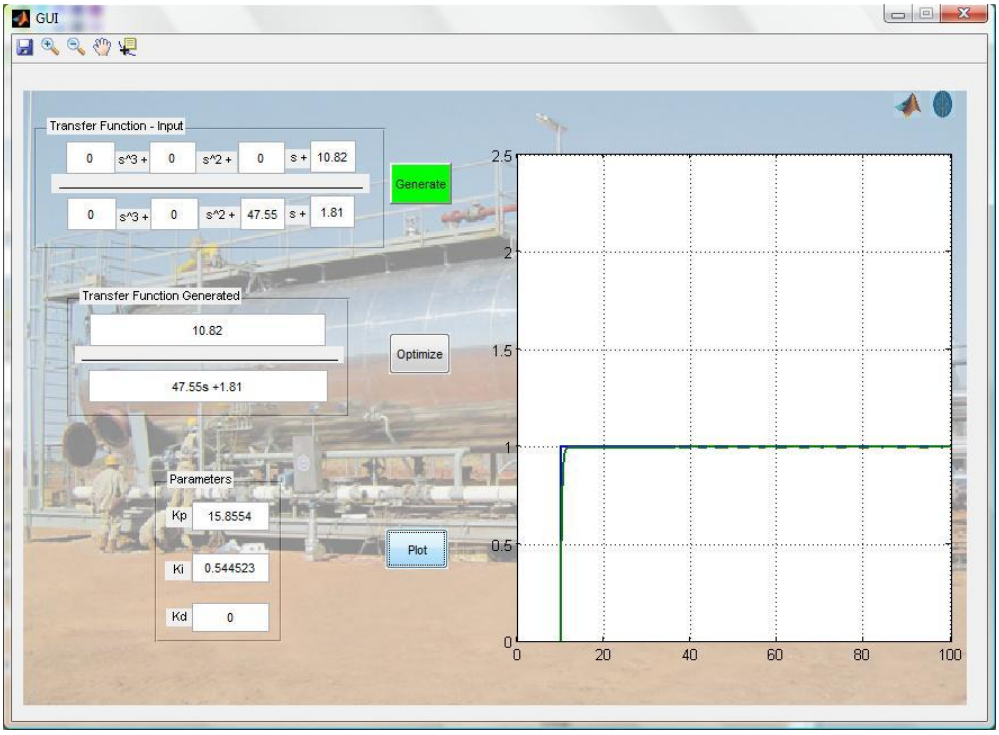


Figure 4.17: GUI – Response of System is generated using Optimized Parameters

The response of the system using the optimized parameters will be plotted after the push button "Plot" is pushed. The user can analyse the error or the performance of the optimized parameters. Other than that, there are a few tools that had been added to help in analysing the plot. This can be observed the Figure 4.18 and 4.19.

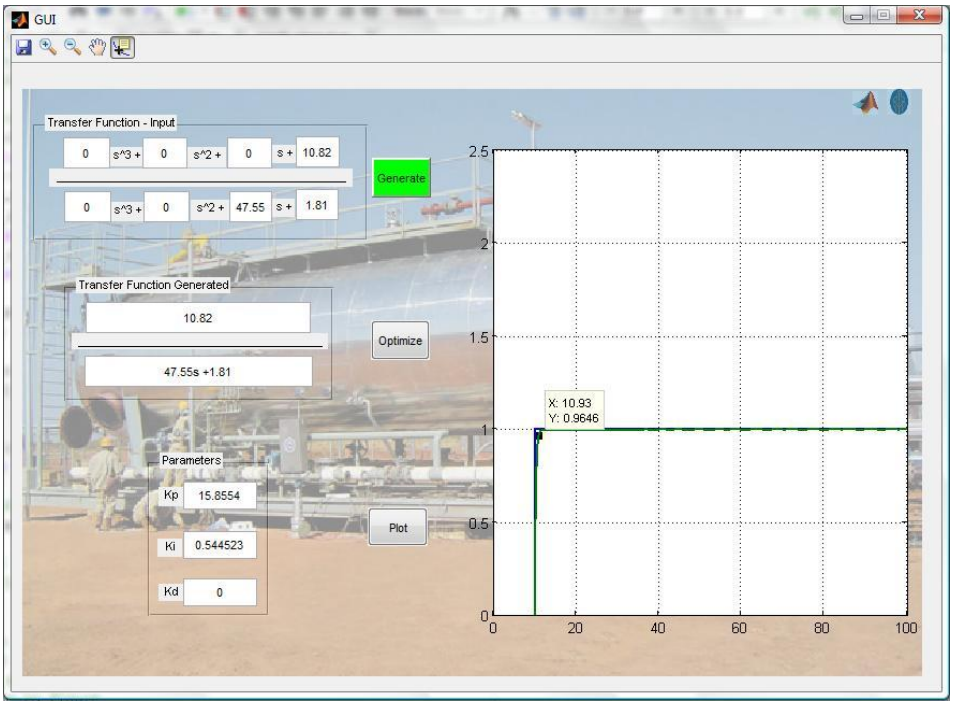


Figure 4.18: GUI – Data Cursor

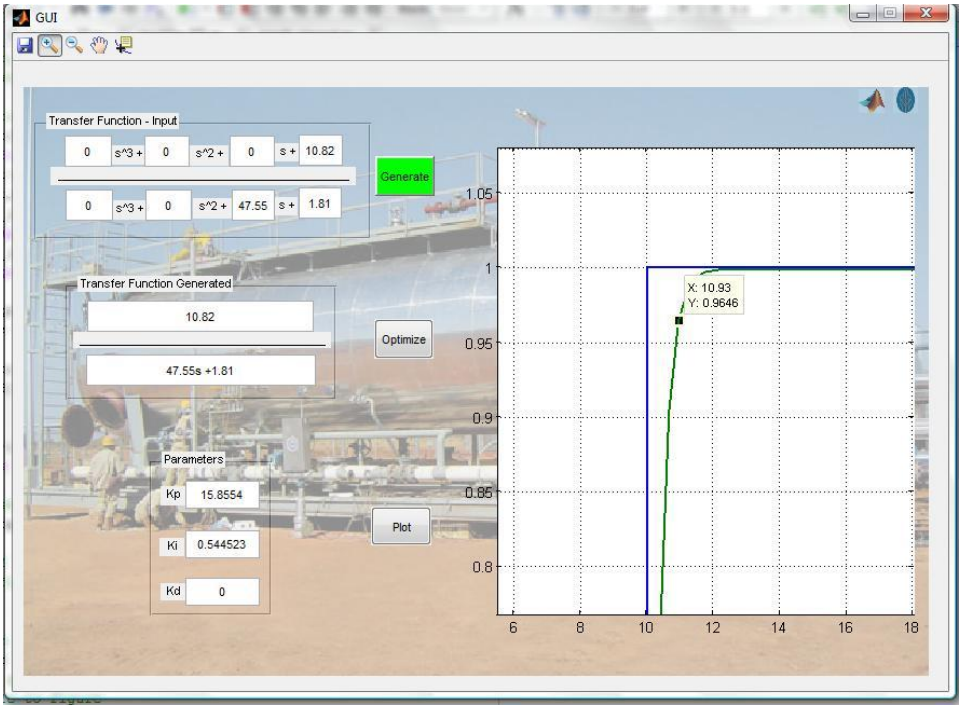


Figure 4.19: GUI – Zoom in/out and Pan

Figure 4.18 had shown that the user can use the Data Cursor tool to obtain the data at certain point on the line. Besides, Figure 4.19 had shown that the user can zoom in and out of the plot to analyse the plot clearly. A Pan tool is added the move the plot around to observe the desired part of the plot.

CHAPTER 5
CONCLUSION AND RECOMMENDATION

A mathematical model of the 3-phase separator was formed. From the mathematical model, trial & error method, Butterworth filter design method, IMC method and BFOA tuning method were tested using Matlab Simulink simulation. A set of parameters were used and the results were obtained. The results have shown that the BFOA method produces the best results and far better that the other 3 conventional methods. BFOA has more than 90% improvements compared to the conventional methods. BFOA can improve the performance of the system thus increases productivity, decrease cost and saves time. A GUI is developed in order to ease the optimization process. This is a promising technique to be used in oil and gas industry other than 3-phase separator. Further studies can be carried out on an actual plant and plants with higher order. Besides, BFOA can be used to compare with other intelligent techniques. Then, more research could be done integrating more than one intelligent technique together to produce even better results.

REFERENCES:

[1] Ahmed Bensenouci, "PID Controllers Design for a Power Plant Using Bacteria Foraging Algorithm", Electrical and Computer Engineering Department, King Abdulaziz University, Jeddah, Kingdom of Saudi Arabia, 2011.

[2] Amir Esmaeili Abharian, Mehdi Alizera, "Hybrid GA-BF Based Intelligent PID active queue management control design for TCP network", Engineeering Department, Islamic Azad University, Adiban-Higher Education Institute, Garmsar, Iran, 2011

[3] Anguluri Rajasekhar, Ravi Kumar Jatoth, Ajith Abraham, Vaclav Snasel, "A Novel Hybrid ABF-PSO Algorithm Based Tuning of Optimal FOPI Speed Controller for PMSM Drive", National Institute of Technology- Warangal, India, Technical University of Ostrava, Czech Republic, 2011

[4] Ben Niu,Yunlong Zhu, Xiaoxian He and Xiangping Zeng, "Optimum Design of PID Controllers Using Only a Germ of Intelligence", Shenyang Institute of Automation, Chinese Academy of Sciences, Liaoning, China, June 21-23, 2006.

[5] Dong Hwa Kim, Ajith Abraham, Jae Hoon Cho, "A hybrid genetic algorithm and bacterial foraging approach for global optimization", Department of Instrumentation and Control Engineering, Hanbat National University, Republic of Korea, 2 April 2007

[6] E. Daryabeigi, M. Moazzami, A. Khodabakhshian, M.H. Mazidi, "A new power system stabilizer design by using Smart Bacteria Foraging Algorithm", Dept. of Electrical Engineering, Islamic Azad University, Iran, 2011.

[7] John Oyekan and Huosheng Hu, "A novel bacterial foraging algorithm for automated tuning of PID controllers of UAVs", School of Computer Science and Electronic Engineering University of Essex, Wivenhoe Park, Colchester CO3 4SQ, United Kingdom, June 10-23, 2010

[8] K.R. Mahmoud, "Design Optimization of a Bow-tie Antenna for 2.45GHz RFID Readers using a Hybrid BSO-NM Algorithm" Electronics & Communications Department, Faculty of Engineering Helwan University, Egypt, 2010

[9] Leandro dos Santos Coelho and Camila da Costa Silveira, "Improved Bacterial Foraging Strategy for Controller Optimization Applied to Robotic Manipulator System", Pontifical Catholic University of Parana, Brazil, Oct 4-6, 2006.

[10] Liu Yijian, Fang Yanjun, "Optimization Design of PID Controller Parameters Based on Improved *E.Coli* Foraging Optimization Algorithm", School of Electrical & Automation Engineering, Nanjing Normal University, Department of Automation, University of Wuhan, Sept 2008.

[11] Swagatam Das, Arijit Biswas, Sambarta DAsgupta and Ajith Abraham, "Bacterial Foraging Optimization Algorithm: Theoretical Foundations, Analysis, and Applications", Dept. of Eelctronics and Telecommunication Engg, Jadavpur University, Kolkata, India

[12] T. Datta, I.S. Misra, B.B. Mangaraj, S.Imtiaj, "Improved Adaptive Bacteria Foraging Algorithm in Optimization of Antenna Array for Faster Convergence", Electronics and Telecommunication Engineering Jadavpur University, University College of Engineering, India, 2008

[13] Wael M. Korani, Hassen Taher Dorrah and Hassan M. Emara, "Bacterial foraging oriented by Particle Swarm Optimization strategy for PID tuning", Department of Electrical Power and Machines Engineering, Cairo University, Giza, Eqypt

[14] W.J. Tang, Q. H. Wu, Senior Member, IEEE and J.R. Saunders, "Bacterial Foraging Algorithm for Dynamic Environments", 2006

[15] XiuJuan Lei, Shuang Wu, Liang GE, Aidong Zhang, "Clustering PPI Data Based on Bacteria Foraging Optimization Algorithm", College of Computer Science, China, Department of Computer Science and Engineering, USA, 2011

[16] Ying Chu, Hua Mi, Huilian Liao, Zhen Ji and Q.H. Wu, "A Fast Bacterial Swarming Algorithm For High-dimensional Function Optimization", 2008

[17] Y. Mishra, S. Mishra and Fangxing Li, "Coordinated Tuning of DFIG-Based Wind Turbines and Batteries Using Bacteria Foraging Technique for Maintaining Constant Grid Power Output", 2011

[18] Zhenyu Yang, Michael Juhl and Bo Lohndorf, "On the Innovation of Level Control of an Offshore Three-Phase Separator" Dept of Electronic Systems, Aalborg University, Ramboll Oil and Gas A/S, Denmark, 2010

[19] Atalla F. Sayda and James H. Taylor, "Modeling and Control of Three-Phase Gravity Separators in Oil Production Facilities"

[20] D.M. Mary Synthia Regis Prabha, Dr. S. Pushpa Kumar and G. Glan Devadhas, "An Optimum Setting of Controller for a dc-dc Converter Using Bacterial Intelligence Technique" Noorul Islam University, Kumaracoil and Heera College of Engg & Tech, Tricandrum, 2011

[21] G.C. Nunes, A.A. Rodrigues Coelho, R. Rodrigues Sumar and R.I. Goytia Mejia, "A Practical Strategy for Controlling Flow Oscillation in Surge

Tanks", Department of Automation and Systems, Federal Universityof Santa Catarina, 2007

[22] Hoang Thanh Nguyen, Bir Bhanu, "Tracking Pedestrians with Bacterial Foraging Optimization Swarms", Center for Research in Intelligent Systems, University of California, Riverside, 2011

[23] IBG Manuaba, M Abdillah, A Soeprijanto, Maruridhi Hery P, "Coordination of PID Based Power System Stabilizer and AVR Using Cobination Bacterial Foraging Technique – Particle Swarm Optimization", Department of Electrical Engineering Institut Teknologi Sepuluh Nopember Surabaya 6011, Department of Electrical Engineering Universitas Udayana Denpasar, Bali, Indonesia.

[24] Nima Amjady, Hamzeh Fatemi and Hamidreza Zareipour, "Solution of Optimal Power Flow Subject to Security Costraints by a New Improved Bacterial Foraging Method", 2012

[25] Sachin Singh, T. Ghose and S. K. Goswami, Optimal Feeder Routing Based on the Bacterial Foraging Technique", Jan 2012

[26] S. S. Patnaik and Prof. A. K. Panda, "Comparative Evaluation of Harmonic Compensation Capability of Active Power Filter with Conventional and Bacterial Foraging Based Control", Department of Electrical Engineering, National Institute of Technology, Rourkela, India

APPENDICES

APPENDIX A

BACTERIA FORAGING OPTIMIZATION ALGORITHM (BFOA) MATLAB CODES

```
%BG.m (MAIN)
% Tunning of PID controller using Bacterial foraging
%
%
% Author: Ho Joon Heng (joonheng89@gmail.com)
% Electrical & Electronics Enginering Dept,
% Universiti Teknologi Petronas, Malaysia
%
% Reference from Wael Mansour (wael192@yahoo.com)
% MSc Student, Electrical Enginering Dept,
% Faculty of Engineering Cairo University, Egypt


%%
%Initialization
%clear all
clc

%Obtain Variables from GUI
Num3 = getappdata(0,'GUI_Num3');
Num2 = getappdata(0,'GUI_Num2');
Num1 = getappdata(0,'GUI_Num1');
Num0 = getappdata(0,'GUI_Num0');
Den3 = getappdata(0,'GUI_Den3');
Den2 = getappdata(0,'GUI_Den2');
Den1 = getappdata(0,'GUI_Den1');
Den0 = getappdata(0,'GUI_Den0');
setappdata(0,'BG_Num3',Num3);
setappdata(0,'BG_Num2',Num2);
setappdata(0,'BG_Num1',Num1);
setappdata(0,'BG_Num0',Num0);
setappdata(0,'BG_Den3',Den3);
setappdata(0,'BG_Den2',Den2);
setappdata(0,'BG_Den1',Den1);
setappdata(0,'BG_Den0',Den0);

p=2;
s=6;
Nc=5;
Ns=4;
Nre=4;
Ned=2;
Sr=s/2;
Ped=0.25;
c(:,1)=0.05*ones(s,1);
for m=1:s
    P(1,:,1,1,1)= 50*rand(s,1)';
    P(2,:,1,1,1)= .2*rand(s,1)';
    %P(3,:,1,1,1)= .2*rand(s,1)';
end

%%
```

```
%Main loop

%Elimination and dispersal loop
for ell=1:Ned


%Reprodution loop


    for K=1:Nre

%  swim/tumble(chemotaxis)loop

        for j=1:Nc

            for i=1:s
                J(i,j,K,ell)=tracklsq_PI(P(:,i,j,K,ell));

% Tumble


                Jlast=J(i,j,K,ell);
                Delta(:,i)=(2*round(rand(p,1))-1).*rand(p,1);


P(:,i,j+1,K,ell)=P(:,i,j,K,ell)+c(i,K)*Delta(:,i)/sqrt(Delta(:,i)'*D
elta(:,i)); % This adds a unit vector in the random direction


                J(i,j+1,K,ell)=tracklsq_PI(P(:,i,j+1,K,ell));
                m=0;         % Initialize counter for swim length
                    while m<Ns
                          m=m+1;
                          if J(i,j+1,K,ell)<Jlast
                             Jlast=J(i,j+1,K,ell);

P(:,i,j+1,K,ell)=P(:,i,j+1,K,ell)+c(i,K)*Delta(:,i)/sqrt(Delta(:,i)'
*Delta(:,i)) ;

J(i,j+1,K,ell)=tracklsq_PI(P(:,i,j+1,K,ell));
                          else
                             m=Ns ;
                          end

                    end
                J(i,j,K,ell)=Jlast;
                sprintf('The value of interation i %3.0f ,j
= %3.0f  , K= %3.0f, ell= %3.0f' , i, j, K ,ell );

            end

        end


%Reprodution
        Jhealth=sum(J(:,:,K,ell),2);
        [Jhealth,sortind]=sort(Jhealth);
        P(:,:,1,K+1,ell)=P(:,sortind,Nc+1,K,ell);
        c(:,K+1)=c(sortind,K);
```

```
%Split the bacteria (reproduction)
            for i=1:Sr
                %Sr2=2*Sr;
                %Sr3=2*Sr;
                P(:,i+Sr,1,K+1,ell)=P(:,i,1,K+1,ell); % The least
fit do not reproduce, the most fit ones split into two identical
copies
                %P(:,i+Sr2,1,K+1,ell)=P(:,i,1,K+1,ell);
                %P(:,i+Sr3,1,K+1,ell)=P(:,i,1,K+1,ell);
                c(i+Sr,K+1)=c(i,K+1);
                %c(i+Sr2,K+1)=c(i,K+1);
                %c(i+Sr3,K+1)=c(i,K+1);
            end
        end %  Go to next reproduction


%Eliminatoin and dispersal
        for m=1:s
            if  Ped>rand % % Generate random number
                P(1,:,1,1,1)= 50*rand(s,1)';
                P(2,:,1,1,1)= .2*rand(s,1)';
                %P(3,:,1,1,1)= .2*rand(s,1)';
            else
                P(:,m,1,1,ell+1)=P(:,m,1,Nre+1,ell);
            end
        end
    end

%Report
           reproduction = J(:,1:Nc,Nre,Ned);
           [jlastreproduction,O] = min(reproduction,[],2);  % min
cost function for each bacterial
           [Y,I] = min(jlastreproduction);
           pbest=P(:,I,O(I,:),K,ell);
           Kp= abs(pbest(1,:))
           Ki= abs(pbest(2,:))
           Kd= 0%abs(pbest(3,:))
           assignin('base','Kp',Kp);
           assignin('base','Ki',Ki);
           assignin('base','Kd',Kd);
           setappdata(0,'BG_Opt_Kp',Kp);
           setappdata(0,'BG_Opt_Ki',Ki);
           setappdata(0,'BG_Opt_Kd',Kd);
```

%tracklsq (function)

```
function F = tracklsq(pid)

         Num3 = getappdata(0,'BG_Num3');
         Num2 = getappdata(0,'BG_Num2');
         Num1 = getappdata(0,'BG_Num1');
         Num0 = getappdata(0,'BG_Num0');
         Den3 = getappdata(0,'BG_Den3');
         Den2 = getappdata(0,'BG_Den2');
         Den1 = getappdata(0,'BG_Den1');
         Den0 = getappdata(0,'BG_Den0');
         Kp = pid(1);
         Ki = pid(2);
         Kd = pid(3);

         sprintf('The value of interation Kp= %3.0f, Ki= %3.0f,
Kd= %3.0f', pid(1), pid(2), pid(3));

         %Compute function value
         simopt =
simset('solver','ode5','SrcWorkspace','Current','DstWorkspace','Curr
ent');  % Initialize sim options
         [tout,xout,yout] = sim('optsim2',[0 50],simopt);

         %Compute the error
         e=yout(1)-1 ;

         %Compute the overshoot
         sys_overshoot = max(yout(1))-1;
         sys_shoot = abs(sys_overshoot);

         alpha = 10;
         beta = 10;
         F = e2*beta+sys_shoot*alpha;

    end
```

APPENDIX B

GUI MATLAB CODES

```
% GUI (MAIN)
function varargout = GUI(varargin)
% GUI M-file for GUI.fig
%      GUI, by itself, creates a new GUI or raises the existing
%      singleton*.
%
%      H = GUI returns the handle to a new GUI or the handle to
%      the existing singleton*.
%
%      GUI('CALLBACK',hObject,eventData,handles,...) calls the local
%      function named CALLBACK in GUI.M with the given input
arguments.
%
%      GUI('Property','Value',...) creates a new GUI or raises the
%      existing singleton*.  Starting from the left, property value
pairs are
%      applied to the GUI before GUI_OpeningFcn gets called.  An
%      unrecognized property name or invalid value makes property
application
%      stop.  All inputs are passed to GUI_OpeningFcn via varargin.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows
only one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% View the above text to modify the response to help GUI

% Last Modified by GUIDE v2.5 26-Jul-2012 16:55:39

% Begin initialization code - DO NOT VIEW
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @GUI_OpeningFcn, ...
                   'gui_OutputFcn',  @GUI_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT VIEW


% --- Executes just before GUI is made visible.
function GUI_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
```

```
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to GUI (see VARARGIN)

% Choose default command line output for GUI
handles.output = hObject;
set(handles.Gen,'BackgroundColor','green');
set(handles.pushbutton_Opt,'BackgroundColor','red');
set(handles.pushbutton_Plot,'BackgroundColor','red');
axes(handles.logo_in);
imshow('UTP_logo.jpg');
axes(handles.logo_in2);
imshow('matlab_logo3.jpg')
opengl software
axes(handles.Background);
imshow('3PS.jpg');
alpha (0.4);
axes(handles.graph);
axis ([0 100 0 2.5]);
grid on;
xlabel('Time');
ylabel('Response');
title('Response vs Time');

% Update handles structure
guidata(hObject, handles);

% UIWAIT makes GUI wait for user response (see UIRESUME)
% uiwait(handles.figure1);


% --- Outputs from this function are returned to the command line.
function varargout = GUI_OutputFcn(hObject, eventdata, handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;



function edit1_Callback(hObject, eventdata, handles)
% hObject    handle to edit1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit1 as text
%        str2double(get(hObject,'String')) returns contents of edit1
as a double


% --- Executes during object creation, after setting all properties.
function edit1_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called
```

```
% Hint: view controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function Num3_Callback(hObject, eventdata, handles)
% hObject    handle to Num3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
handles.Num3_value = str2double(get(hObject,'String'));
guidata(hObject, handles);

% Hints: get(hObject,'String') returns contents of Num3 as text
%        str2double(get(hObject,'String')) returns contents of Num3
as a
%        double


% --- Executes during object creation, after setting all properties.
function Num3_CreateFcn(hObject, eventdata, handles)
% hObject    handle to Num3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: view controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function edit5_Callback(hObject, eventdata, handles)
% hObject    handle to edit5 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit5 as text
%        str2double(get(hObject,'String')) returns contents of edit5
as a double


% --- Executes during object creation, after setting all properties.
function edit5_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit5 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: view controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
```

```
function Num2_Callback(hObject, eventdata, handles)
% hObject    handle to Num2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
handles.Num2_value = str2double(get(hObject,'String'));
guidata(hObject, handles);

% Hints: get(hObject,'String') returns contents of Num2 as text
%        str2double(get(hObject,'String')) returns contents of Num2
as a double


% --- Executes during object creation, after setting all properties.
function Num2_CreateFcn(hObject, eventdata, handles)
% hObject    handle to Num2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: view controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function Num1_Callback(hObject, eventdata, handles)
% hObject    handle to Num1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
handles.Num1_value = str2double(get(hObject,'String'));
guidata(hObject, handles);

% Hints: get(hObject,'String') returns contents of Num1 as text
%        str2double(get(hObject,'String')) returns contents of Num1
as a double


% --- Executes during object creation, after setting all properties.
function Num1_CreateFcn(hObject, eventdata, handles)
% hObject    handle to Num1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: view controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


% --- If Enable == 'on', executes on mouse press in 5 pixel border.
% --- Otherwise, executes on mouse press in 5 pixel border or over
edit5.
```

```
function edit5_ButtonDownFcn(hObject, eventdata, handles)
% hObject    handle to edit5 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)


% --------------------------------------------------------------------
--
function Menu_Callback(hObject, eventdata, handles)
% hObject    handle to File (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)


% --- Executes on button press in pushbutton_Opt.
function pushbutton_Opt_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton_Opt (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
BG;
BG_Opt_Kp = getappdata(0,'BG_Opt_Kp');
BG_Opt_Ki = getappdata(0,'BG_Opt_Ki');
BG_Opt_Kd = getappdata(0,'BG_Opt_Kd');
set(handles.Opt_Kp,'String',BG_Opt_Kp);
set(handles.Opt_Ki,'String',BG_Opt_Ki);
set(handles.Opt_Kd,'String',BG_Opt_Kd);
setappdata(0,'BG_Opt_Kp',BG_Opt_Kp);
setappdata(0,'BG_Opt_Ki',BG_Opt_Ki);
setappdata(0,'BG_Opt_Kd',BG_Opt_Kd);
set(handles.Gen,'BackgroundColor','default');
set(handles.pushbutton_Opt,'BackgroundColor','default');
set(handles.pushbutton_Plot,'BackgroundColor','green');
guidata(hObject,handles);

% --- Executes on button press in pushbutton_Plot.
function pushbutton_Plot_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton_Plot (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
BG_Opt_Kp = getappdata(0,'BG_Opt_Kp');
BG_Opt_Ki = getappdata(0,'BG_Opt_Ki');
BG_Opt_Kd = getappdata(0,'BG_Opt_Kd');
graph(1) = handles.Num3_value;
graph(2) = handles.Num2_value;
graph(3) = handles.Num1_value;
graph(4) = handles.Num0_value;
graph(5) = handles.Den3_value;
graph(6) = handles.Den2_value;
graph(7) = handles.Den1_value;
graph(8) = handles.Den0_value;
graph(9) = BG_Opt_Kp;
graph(10) = BG_Opt_Ki;
graph(11) = BG_Opt_Kd;
[tout yout]= graph_simout(graph);
axes(handles.graph);
plot(tout,yout(:,2),tout,yout(:,1),'LineWidth',(1.5));
axis([0 100 0 2.5]);
grid on;
%xlabel('Time');
%ylabel('Response');
%title('Response vs Time');
```

```
set(handles.Gen,'BackgroundColor','green');
set(handles.pushbutton_Opt,'BackgroundColor','default');
set(handles.pushbutton_Plot,'BackgroundColor','default');
guidata(hObject,handles);



function Num0_Callback(hObject, eventdata, handles)
% hObject    handle to Num0 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
handles.Num0_value = str2double(get(hObject,'String'));
guidata(hObject, handles);

% Hints: get(hObject,'String') returns contents of Num0 as text
%        str2double(get(hObject,'String')) returns contents of Num0
as a double


% --- Executes during object creation, after setting all properties.
function Num0_CreateFcn(hObject, eventdata, handles)
% hObject    handle to Num0 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: view controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function Den3_Callback(hObject, eventdata, handles)
% hObject    handle to Den3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
handles.Den3_value = str2double(get(hObject,'String'));
guidata(hObject, handles);

% Hints: get(hObject,'String') returns contents of Den3 as text
%        str2double(get(hObject,'String')) returns contents of Den3
as a double


% --- Executes during object creation, after setting all properties.
function Den3_CreateFcn(hObject, eventdata, handles)
% hObject    handle to Den3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: view controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
```

```
function Den2_Callback(hObject, eventdata, handles)
% hObject    handle to Den2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
handles.Den2_value = str2double(get(hObject,'String'));
guidata(hObject, handles);

% Hints: get(hObject,'String') returns contents of Den2 as text
%        str2double(get(hObject,'String')) returns contents of Den2
as a double


% --- Executes during object creation, after setting all properties.
function Den2_CreateFcn(hObject, eventdata, handles)
% hObject    handle to Den2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: view controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function Den1_Callback(hObject, eventdata, handles)
% hObject    handle to Den1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
handles.Den1_value = str2double(get(hObject,'String'));
guidata(hObject, handles);

% Hints: get(hObject,'String') returns contents of Den1 as text
%        str2double(get(hObject,'String')) returns contents of Den1
as a double


% --- Executes during object creation, after setting all properties.
function Den1_CreateFcn(hObject, eventdata, handles)
% hObject    handle to Den1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: view controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function Den0_Callback(hObject, eventdata, handles)
% hObject    handle to Den0 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
```

```
% handles    structure with handles and user data (see GUIDATA)
handles.Den0_value = str2double(get(hObject,'String'));
guidata(hObject, handles);

% Hints: get(hObject,'String') returns contents of Den0 as text
%        str2double(get(hObject,'String')) returns contents of Den0
as a double


% --- Executes during object creation, after setting all properties.
function Den0_CreateFcn(hObject, eventdata, handles)
% hObject    handle to Den0 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: view controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function Opt_Kp_Callback(hObject, eventdata, handles)
% hObject    handle to Opt_Kp (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of Opt_Kp as text
%        str2double(get(hObject,'String')) returns contents of
Opt_Kp as a double


% --- Executes during object creation, after setting all properties.
function Opt_Kp_CreateFcn(hObject, eventdata, handles)
% hObject    handle to Opt_Kp (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: view controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function Opt_Ki_Callback(hObject, eventdata, handles)
% hObject    handle to Opt_Ki (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of Opt_Ki as text
%        str2double(get(hObject,'String')) returns contents of
Opt_Ki as a double
```

```
% --- Executes during object creation, after setting all properties.
function Opt_Ki_CreateFcn(hObject, eventdata, handles)
% hObject    handle to Opt_Ki (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: view controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function Opt_Kd_Callback(hObject, eventdata, handles)
% hObject    handle to Opt_Kd (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of Opt_Kd as text
%        str2double(get(hObject,'String')) returns contents of
Opt_Kd as a double


% --- Executes during object creation, after setting all properties.
function Opt_Kd_CreateFcn(hObject, eventdata, handles)
% hObject    handle to Opt_Kd (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: view controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


% --- Executes on button press in Gen.
function Gen_Callback(hObject, eventdata, handles)
% hObject    handle to Gen (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
Trans_Num3 = handles.Num3_value;
Trans_Num2 = handles.Num2_value;
Trans_Num1 = handles.Num1_value;
Trans_Num0 = handles.Num0_value;
Trans_Den3 = handles.Den3_value;
Trans_Den2 = handles.Den2_value;
Trans_Den1 = handles.Den1_value;
Trans_Den0 = handles.Den0_value;
assignin('base','Num3',Trans_Num3);
assignin('base','Num2',Trans_Num2);
assignin('base','Num1',Trans_Num1);
assignin('base','Num0',Trans_Num0);
assignin('base','Den3',Trans_Den3);
assignin('base','Den2',Trans_Den2);
assignin('base','Den1',Trans_Den1);
```

```
assignin('base','Den0',Trans_Den0);
setappdata(0,'GUI_Num3',Trans_Num3);
setappdata(0,'GUI_Num2',Trans_Num2);
setappdata(0,'GUI_Num1',Trans_Num1);
setappdata(0,'GUI_Num0',Trans_Num0);
setappdata(0,'GUI_Den3',Trans_Den3);
setappdata(0,'GUI_Den2',Trans_Den2);
setappdata(0,'GUI_Den1',Trans_Den1);
setappdata(0,'GUI_Den0',Trans_Den0);

% To display generated transfer function - Numerator
if Trans_Num3== 0
    if Trans_Num2== 0
        if Trans_Num1== 0
            if Trans_Num0== 0
                display_Num = 'Error!';
            else
                display_Num = num2str(Trans_Num0);
            end
        else
            if Trans_Num0== 0
                display_Num = [num2str(Trans_Num1),'s'];
            elseif Trans_Num0 < 0
                display_Num = [num2str(Trans_Num1),'s
',num2str(Trans_Num0)];
            else
                display_Num = [num2str(Trans_Num1),'s
+',num2str(Trans_Num0)];
            end
        end
    else
        if Trans_Num1== 0
            if Trans_Num0== 0
                display_Num = [num2str(Trans_Num2),'s^2'];
            elseif Trans_Num0 < 0
                display_Num = [num2str(Trans_Num2),'s^2
',num2str(Trans_Num0)];
            else
                display_Num = [num2str(Trans_Num2),'s^2
+',num2str(Trans_Num0)];
            end
        elseif Trans_Num1 < 0
            if Trans_Num0== 0
                display_Num = [num2str(Trans_Num2),'s^2
',num2str(Trans_Num1),'s'];
            elseif Trans_Num0 < 0
                display_Num = [num2str(Trans_Num2),'s^2
',num2str(Trans_Num1),'s ',num2str(Trans_Num0)];
            else
                display_Num = [num2str(Trans_Num2),'s^2
',num2str(Trans_Num1),'s +',num2str(Trans_Num0)];
            end
        else
            if Trans_Num0== 0
                display_Num = [num2str(Trans_Num2),'s^2
+',num2str(Trans_Num1),'s'];
            elseif Trans_Num0 < 0
                display_Num = [num2str(Trans_Num2),'s^2
+',num2str(Trans_Num1),'s ',num2str(Trans_Num0)];
            else
```

```
                display_Num = [num2str(Trans_Num2),'s^2
+',num2str(Trans_Num1),'s +',num2str(Trans_Num0)];
            end
        end
    end
else
    if Trans_Num2== 0
        if Trans_Num1== 0
            if Trans_Num0== 0
                display_Num = [num2str(Trans_Num3),'s^3 +'];
            elseif Trans_Num0 < 0
                display_Num = [num2str(Trans_Num3),'s^3
',num2str(Trans_Num0)];
            else
                display_Num = [num2str(Trans_Num3),'s^3
+',num2str(Trans_Num0)];
            end
        elseif Trans_Num1 < 0
            if Trans_Num0== 0
                display_Num = [num2str(Trans_Num3),'s^3
',num2str(Trans_Num1),'s'];
            elseif Trans_Num0 < 0
                display_Num = [num2str(Trans_Num3),'s^3
',num2str(Trans_Num1),'s ',num2str(Trans_Num0)];
            else
                display_Num = [num2str(Trans_Num3),'s^3
',num2str(Trans_Num1),'s +',num2str(Trans_Num0)];
            end
        else
            if Trans_Num0== 0
                display_Num = [num2str(Trans_Num3),'s^3
+',num2str(Trans_Num1),'s'];
            elseif Trans_Num0 < 0
                display_Num = [num2str(Trans_Num3),'s^3
+',num2str(Trans_Num1),'s ',num2str(Trans_Num0)];
            else
                display_Num = [num2str(Trans_Num3),'s^3
+',num2str(Trans_Num1),'s +',num2str(Trans_Num0)];
            end
        end
    elseif Trans_Num2 < 0
        if Trans_Num1== 0
            if Trans_Num0== 0
                display_Num = [num2str(Trans_Num3),'s^3
',num2str(Trans_Num2),'s^2'];
            elseif Trans_Num0 < 0
                display_Num = [num2str(Trans_Num3),'s^3
',num2str(Trans_Num2),'s^2 ',num2str(Trans_Num0)];
            else
                display_Num = [num2str(Trans_Num3),'s^3
',num2str(Trans_Num2),'s^2 +',num2str(Trans_Num0)];
            end
        elseif Trans_Num1 < 0
            if Trans_Num0== 0
                display_Num = [num2str(Trans_Num3),'s^3
',num2str(Trans_Num2),'s^2 ',num2str(Trans_Num1),'s'];
            elseif Trans_Num0 < 0
                display_Num = [num2str(Trans_Num3),'s^3
',num2str(Trans_Num2),'s^2 ',num2str(Trans_Num1),'s
',num2str(Trans_Num0)];
            else
```

```
                display_Num = [num2str(Trans_Num3),'s^3
',num2str(Trans_Num2),'s^2 ',num2str(Trans_Num1),'s
+',num2str(Trans_Num0)];
            end
        else
            if Trans_Num0== 0
                display_Num = [num2str(Trans_Num3),'s^3
',num2str(Trans_Num2),'s^2 +',num2str(Trans_Num1),'s'];
            elseif Trans_Num0 < 0
                display_Num = [num2str(Trans_Num3),'s^3
',num2str(Trans_Num2),'s^2 +',num2str(Trans_Num1),'s
',num2str(Trans_Num0)];
            else
                display_Num = [num2str(Trans_Num3),'s^3
',num2str(Trans_Num2),'s^2 +',num2str(Trans_Num1),'s
+',num2str(Trans_Num0)];
            end
        end
    else
        if Trans_Num1== 0
            if Trans_Num0== 0
                display_Num = [num2str(Trans_Num3),'s^3
+',num2str(Trans_Num2),'s^2'];
            elseif Trans_Num0 < 0
                display_Num = [num2str(Trans_Num3),'s^3
+',num2str(Trans_Num2),'s^2 ',num2str(Trans_Num0)];
            else
                display_Num = [num2str(Trans_Num3),'s^3
+',num2str(Trans_Num2),'s^2 +',num2str(Trans_Num0)];
            end
        elseif Trans_Num1 < 0
            if Trans_Num0== 0
                display_Num = [num2str(Trans_Num3),'s^3
+',num2str(Trans_Num2),'s^2 ',num2str(Trans_Num1),'s'];
            elseif Trans_Num0 < 0
                display_Num = [num2str(Trans_Num3),'s^3
+',num2str(Trans_Num2),'s^2 ',num2str(Trans_Num1),'s
',num2str(Trans_Num0)];
            else
                display_Num = [num2str(Trans_Num3),'s^3
+',num2str(Trans_Num2),'s^2 ',num2str(Trans_Num1),'s
+',num2str(Trans_Num0)];
            end
        else
            if Trans_Num0== 0
                display_Num = [num2str(Trans_Num3),'s^3
+',num2str(Trans_Num2),'s^2 +',num2str(Trans_Num1),'s'];
            elseif Trans_Num0 < 0
                display_Num = [num2str(Trans_Num3),'s^3
+',num2str(Trans_Num2),'s^2 +',num2str(Trans_Num1),'s
',num2str(Trans_Num0)];
            else
                display_Num = [num2str(Trans_Num3),'s^3
+',num2str(Trans_Num2),'s^2 +',num2str(Trans_Num1),'s
+',num2str(Trans_Num0)];
            end
        end
    end
end

%To display generated transfer function - Denominator
```

```
if Trans_Den3== 0
    if Trans_Den2== 0
        if Trans_Den1== 0
            if Trans_Den0== 0
                display_Den = 'Error!';
            else
                display_Den = num2str(Trans_Den0);
            end
        else
            if Trans_Den0== 0
                display_Den = [num2str(Trans_Den1),'s'];
            elseif Trans_Den0 < 0
                display_Den = [num2str(Trans_Den1),'s
',num2str(Trans_Den0)];
            else
                display_Den = [num2str(Trans_Den1),'s
+',num2str(Trans_Den0)];
            end
        end
    else
        if Trans_Den1== 0
            if Trans_Den0== 0
                display_Den = [num2str(Trans_Den2),'s^2'];
            elseif Trans_Den0 < 0
                display_Den = [num2str(Trans_Den2),'s^2
',num2str(Trans_Den0)];
            else
                display_Den = [num2str(Trans_Den2),'s^2
+',num2str(Trans_Den0)];
            end
        elseif Trans_Den1 < 0
            if Trans_Den0== 0
                display_Den = [num2str(Trans_Den2),'s^2
',num2str(Trans_Den1),'s'];
            elseif Trans_Den0 < 0
                display_Den = [num2str(Trans_Den2),'s^2
',num2str(Trans_Den1),'s ',num2str(Trans_Den0)];
            else
                display_Den = [num2str(Trans_Den2),'s^2
',num2str(Trans_Den1),'s +',num2str(Trans_Den0)];
            end
        else
            if Trans_Den0== 0
                display_Den = [num2str(Trans_Den2),'s^2
+',num2str(Trans_Den1),'s'];
            elseif Trans_Den0 < 0
                display_Den = [num2str(Trans_Den2),'s^2
+',num2str(Trans_Den1),'s ',num2str(Trans_Den0)];
            else
                display_Den = [num2str(Trans_Den2),'s^2
+',num2str(Trans_Den1),'s +',num2str(Trans_Den0)];
            end
        end
    end
else
    if Trans_Den2== 0
        if Trans_Den1== 0
            if Trans_Den0== 0
                display_Den = [num2str(Trans_Den3),'s^3 +'];
            elseif Trans_Den0 < 0
```

```
                display_Den = [num2str(Trans_Den3),'s^3
',num2str(Trans_Den0)];
            else
                display_Den = [num2str(Trans_Den3),'s^3
+',num2str(Trans_Den0)];
            end
        elseif Trans_Den1 < 0
            if Trans_Den0== 0
                display_Den = [num2str(Trans_Den3),'s^3
',num2str(Trans_Den1),'s'];
            elseif Trans_Den0 < 0
                display_Den = [num2str(Trans_Den3),'s^3
',num2str(Trans_Den1),'s ',num2str(Trans_Den0)];
            else
                display_Den = [num2str(Trans_Den3),'s^3
',num2str(Trans_Den1),'s +',num2str(Trans_Den0)];
            end
        else
            if Trans_Den0== 0
                display_Den = [num2str(Trans_Den3),'s^3
+',num2str(Trans_Den1),'s'];
            elseif Trans_Den0 < 0
                display_Den = [num2str(Trans_Den3),'s^3
+',num2str(Trans_Den1),'s ',num2str(Trans_Den0)];
            else
                display_Den = [num2str(Trans_Den3),'s^3
+',num2str(Trans_Den1),'s +',num2str(Trans_Den0)];
            end
        end
    elseif Trans_Den2 < 0
        if Trans_Den1== 0
            if Trans_Den0== 0
                display_Den = [num2str(Trans_Den3),'s^3
',num2str(Trans_Den2),'s^2'];
            elseif Trans_Den0 < 0
                display_Den = [num2str(Trans_Den3),'s^3
',num2str(Trans_Den2),'s^2 ',num2str(Trans_Den0)];
            else
                display_Den = [num2str(Trans_Den3),'s^3
',num2str(Trans_Den2),'s^2 +',num2str(Trans_Den0)];
            end
        elseif Trans_Den1 < 0
            if Trans_Den0== 0
                display_Den = [num2str(Trans_Den3),'s^3
',num2str(Trans_Den2),'s^2 ',num2str(Trans_Den1),'s'];
            elseif Trans_Den0 < 0
                display_Den = [num2str(Trans_Den3),'s^3
',num2str(Trans_Den2),'s^2 ',num2str(Trans_Den1),'s
',num2str(Trans_Den0)];
            else
                display_Den = [num2str(Trans_Den3),'s^3
',num2str(Trans_Den2),'s^2 ',num2str(Trans_Den1),'s
+',num2str(Trans_Den0)];
            end
        else
            if Trans_Den0== 0
                display_Den = [num2str(Trans_Den3),'s^3
',num2str(Trans_Den2),'s^2 +',num2str(Trans_Den1),'s'];
            elseif Trans_Den0 < 0
```

```
                display_Den = [num2str(Trans_Den3),'s^3
',num2str(Trans_Den2),'s^2 +',num2str(Trans_Den1),'s
',num2str(Trans_Den0)];
            else
                display_Den = [num2str(Trans_Den3),'s^3
',num2str(Trans_Den2),'s^2 +',num2str(Trans_Den1),'s
+',num2str(Trans_Den0)];
            end
        end
    else
        if Trans_Den1== 0
            if Trans_Den0== 0
                display_Den = [num2str(Trans_Den3),'s^3
+',num2str(Trans_Den2),'s^2'];
            elseif Trans_Den0 < 0
                display_Den = [num2str(Trans_Den3),'s^3
+',num2str(Trans_Den2),'s^2 ',num2str(Trans_Den0)];
            else
                display_Den = [num2str(Trans_Den3),'s^3
+',num2str(Trans_Den2),'s^2 +',num2str(Trans_Den0)];
            end
        elseif Trans_Den1 < 0
            if Trans_Den0== 0
                display_Den = [num2str(Trans_Den3),'s^3
+',num2str(Trans_Den2),'s^2 ',num2str(Trans_Den1),'s'];
            elseif Trans_Den0 < 0
                display_Den = [num2str(Trans_Den3),'s^3
+',num2str(Trans_Den2),'s^2 ',num2str(Trans_Den1),'s
',num2str(Trans_Den0)];
            else
                display_Den = [num2str(Trans_Den3),'s^3
+',num2str(Trans_Den2),'s^2 ',num2str(Trans_Den1),'s
+',num2str(Trans_Den0)];
            end
        else
            if Trans_Den0== 0
                display_Den = [num2str(Trans_Den3),'s^3
+',num2str(Trans_Den2),'s^2 +',num2str(Trans_Den1),'s'];
            elseif Trans_Den0 < 0
                display_Den = [num2str(Trans_Den3),'s^3
+',num2str(Trans_Den2),'s^2 +',num2str(Trans_Den1),'s
',num2str(Trans_Den0)];
            else
                display_Den = [num2str(Trans_Den3),'s^3
+',num2str(Trans_Den2),'s^2 +',num2str(Trans_Den1),'s
+',num2str(Trans_Den0)];
            end
        end
    end
end
axes(handles.graph);
cla;
set(handles.Gen_Num,'String',display_Num);
set(handles.Gen_Den,'String',display_Den);
set(handles.Gen,'BackgroundColor','default');
set(handles.pushbutton_Opt,'BackgroundColor','green');
set(handles.pushbutton_Plot,'BackgroundColor','red');
guidata(hObject,handles);

function Gen_Num_Callback(hObject, eventdata, handles)
% hObject    handle to Gen_Num (see GCBO)
```

```
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of Gen_Num as text
%        str2double(get(hObject,'String')) returns contents of
Gen_Num as a double


% --- Executes during object creation, after setting all properties.
function Gen_Num_CreateFcn(hObject, eventdata, handles)
% hObject    handle to Gen_Num (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: view controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


function Gen_Den_Callback(hObject, eventdata, handles)
% hObject    handle to Gen_Den (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of Gen_Den as text
%        str2double(get(hObject,'String')) returns contents of
Gen_Den as a double


% --- Executes during object creation, after setting all properties.
function Gen_Den_CreateFcn(hObject, eventdata, handles)
% hObject    handle to Gen_Den (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: view controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


% ------------------------------------------------------------------
--
function File_Callback(hObject, eventdata, handles)
% hObject    handle to File (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% ------------------------------------------------------------------
--
function View_Callback(hObject, eventdata, handles)
% hObject    handle to View (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
```

```
% --------------------------------------------------------------------
--
function View_Zoomin_Callback(hObject, eventdata, handles)
% hObject    handle to View_Zoomin (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)


% --------------------------------------------------------------------
--
function View_Zoomout_Callback(hObject, eventdata, handles)
% hObject    handle to View_Zoomout (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)


% --------------------------------------------------------------------
--
function View_Pan_Callback(hObject, eventdata, handles)
% hObject    handle to View_Pan (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)


% --------------------------------------------------------------------
--
function View_Cursor_Callback(hObject, eventdata, handles)
% hObject    handle to View_Cursor (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)


% --------------------------------------------------------------------
--
function File_Save_Callback(hObject, eventdata, handles)
% hObject    handle to File_Save (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)


% --------------------------------------------------------------------
--
function File_Saveas_Callback(hObject, eventdata, handles)
% hObject    handle to File_Saveas (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)


% --------------------------------------------------------------------
--
function File_Close_Callback(hObject, eventdata, handles)
% hObject    handle to File_Close (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
```

```
% graph_simout (function)
function [F G] = graph_simout(const_graph)

        % Variables
        Num3 = const_graph(1);
        Num2 = const_graph(2);
        Num1 = const_graph(3);
        Num0 = const_graph(4);
        Den3 = const_graph(5);
        Den2 = const_graph(6);
        Den1 = const_graph(7);
        Den0 = const_graph(8);
        Kp = const_graph(9);
        Ki = const_graph(10);
        Kd = const_graph(11);

        %sprintf('The value of interation Kp= %3.0f, Ki= %3.0f,
Kd= %3.0f', const_graph(9), const_graph(10), const_graph(11));

        %Compute function value
        simopt3 =
simset('SrcWorkspace','Current','DstWorkspace','Current');  %
Initialize sim options
        [tout3,xout3,yout3] = sim('optsim2',[0 100],simopt3);

        F = tout3;
        G = yout3;

    end
```